

Formularios Web. La clase Page

Indice

Nº- 6 Formularios Web. La clase Page.....	1
1. Formularios Web.....	1
1.1 Tipos de ficheros en ASP.NET.....	2
1.2 Directorio de una aplicación .NET.....	2
1.3 Controles de servidor.....	3
2. Formularios HTML.....	4
2.1 La etiqueta <form>.....	4
2.2 Ejemplo 1.....	12
2.3 Ejemplo 2.....	18
3. Clases de los controles HTML.....	23
3.1 Eventos avanzados con el control HtmlInputImage.....	24
3.2 La clase HtmlControls.....	28
3.3 La clase HtmlContainerControl.....	29
3.4 La clase HtmlInputControl.....	29
4. La clase Page.....	30
4.1 Objeto Request.....	30
4.2 Objeto Response.....	37
4.3 Objeto Server.....	42
4.4 Almacenar estados y las aplicaciones con ASP.NET.....	43
4.5 Estado "Application".....	46
4.6 El fichero GLOBAL.ASAX.....	47
4.6 Estado "Session".....	51
5. Configuración de ASP.NET.....	52
5.1 Configuraciones anidadas.....	54
5.2 Almacenar información en el fichero web.config.....	55
6. Administración del sitio web.....	58
Ejercicios.....	62
Ejercicio 1.....	62
Ejercicio 2.....	63
Ejercicio 3.....	63

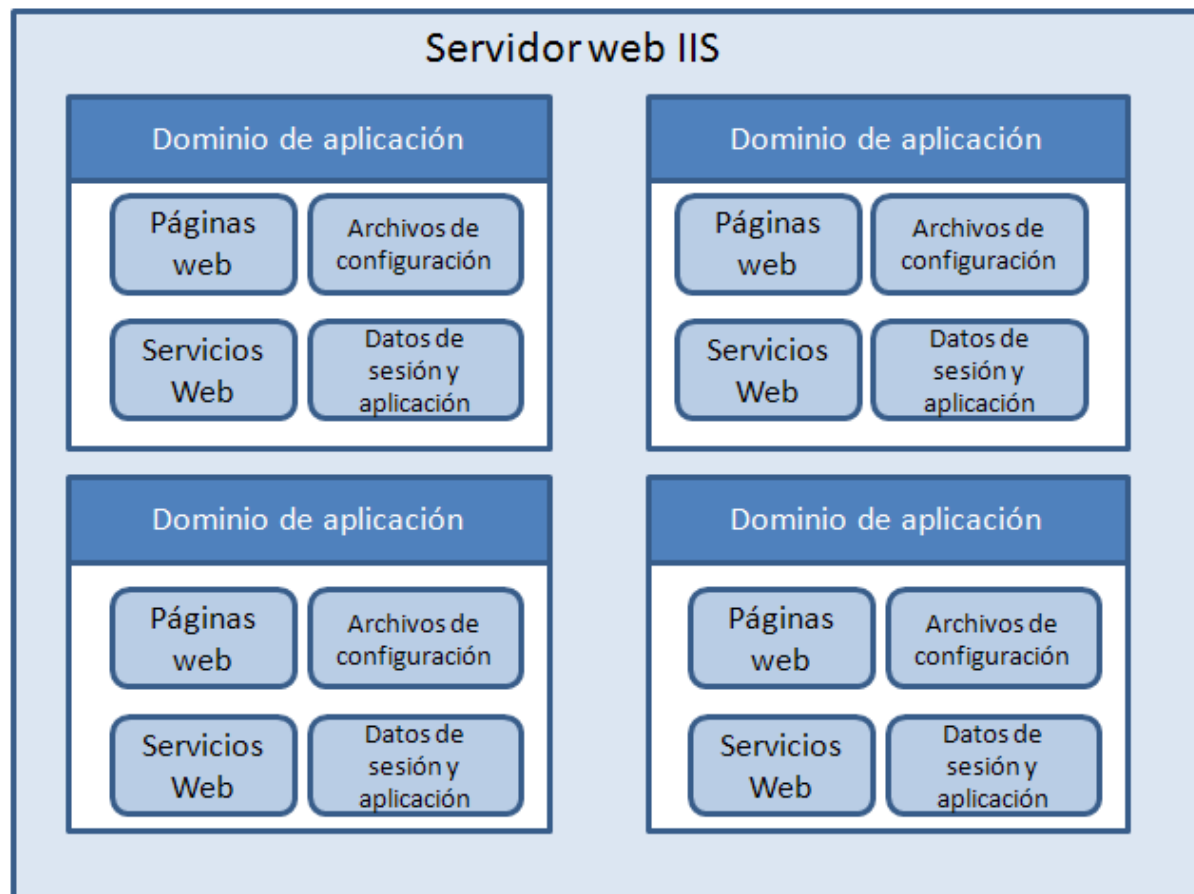
Nº- 6 Formularios Web. La clase Page

1. Formularios Web

Comenzamos ya a conocer las partes de una aplicación en ASP.NET, que en su mayoría serán formularios web pero de alguna forma deberán estar conectados en una gran aplicación que será nuestro proyecto web.

Nuestro proyecto web va a comportarse como una gran aplicación donde los elementos serán las páginas HTML, las páginas ASP.NET y el resto de archivos que compartirán una serie de recursos y valores de configuración. Cada aplicación ASP.NET es independiente, esto es, si tenemos distintos directorios virtuales con distintos proyectos en ASP.NET cada uno tendrá sus particularidades, recursos y configuraciones. A esta separación lógica la llamaremos dominio de aplicación y está aislado del resto de los dominios de aplicación, gracias esto al ejecutarse cada una en sus propios espacios de memoria no interferirán entre ellas y un fallo en una de ellas no provocará fallos en otras.

Una aplicación ASP.NET se compone pues de archivos, páginas, controladores de eventos, módulos y otros ficheros que estarán todos en un directorio virtual y configurados así como una sola aplicación. Así que dentro de nuestro Internet Information Server tendremos:



1.1 Tipos de ficheros en ASP.NET

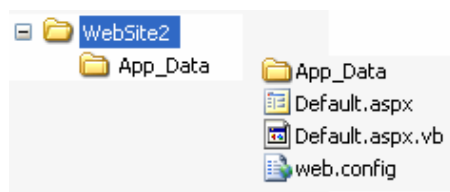
Veamos que tipos de ficheros nos podemos encontrar en nuestros proyectos web:

- .aspx. Son los ficheros que contienen código de las páginas. Recuerda que si se crea con la peculiaridad del "código detrás" se compondrá de dos páginas, una aspx con los controles de usuario y el HTML para darle forma y colores y otra con extensión aspx.vb que tendrá el código VB de la página
- .ascx. Son controles de usuario, son similares a las páginas web pero no se pueden llamar directamente sino que deben incluirse dentro de una página. Digamos que creamos un control que haga un proceso que nosotros queramos, lo grabamos en una página de este tipo y luego en las páginas que queramos utilizarlos "importaremos" este archivo, y por tanto, la definición del control.
- .asmx. Son servicios Web que son colecciones de métodos que pueden llamarse a través de Internet. Veremos ejemplos mas adelante.
- web.config. Este también nos suena, es un fichero de texto en formato XML que contiene la configuración de nuestra aplicación ASP.NET.
- Global.asax. Es un archivo de alcance global, es decir , podremos definir en él variables que van a ser globales para todo nuestro sitio web. fíjate qué fácil es pasar valores entre páginas si resulta que tenemos variables globales que podemos leer y modificar en cualquier momento.
- .vb Son las página de "código detrás" que hemos comentado antes, complementan a las aspx cuando queremos que el código VB esté en una página independiente.

Estos digamos que son los ficheros del mundo ASP.NET, además de estos tendremos hojas de estilos, ficheros HTML, imágenes, ...

1.2 Directorio de una aplicación .NET

En nuestros sencillos ejemplos todavía no tenemos mas una carpeta además de las páginas, fíjate:



Si te fijas en el gráfico de la derecha tenemos nuestra página predeterminada default.aspx, su asociada para el código VB: default.aspx.vb y el fichero de configuración de la aplicación web: web.config. Además una carpeta "App_Data" vacía, los archivos ya los conocemos, veamos ahora que carpetas puede crear nuestro entorno para almacenar partes de nuestro web:

- **Bin.** Contiene componentes compilados (DLL) que utilizará nuestra aplicación web. Por ejemplo si adquirimos un componente para dibujar gráficos de gestión avanzados, el entorno detectará que se va a utilizar ese ensamblado y lo colocará allí, así estará disponible cuando movamos el web completo al servidor de explotación.
- **App_Code.** Contiene código fuente que se compila en cada ejecución. De momento poco lo veremos.
- **App_GlobalResources.** Almacena recursos globales que serán accesibles por todas las páginas de la aplicación web.
- **App_LocalResources.** Igual que la anterior pero los recursos sólo están accesibles para una página.
- **App-WebReferences.** Almacena referencias a servicios web.

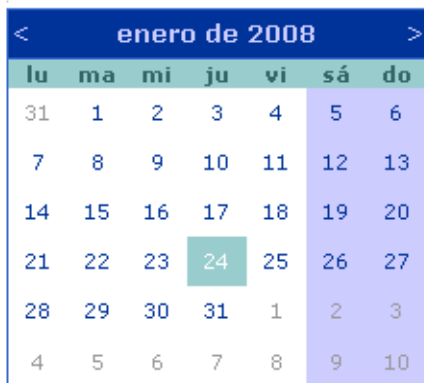
- **App_Data.** Almacena datos como ficheros XML, o bases de datos en SQL Server
- **App_Themes.** Guarda los distintos temas utilizados en el web. También lo veremos mas adelante.

1.3 Controles de servidor

Ya hemos comentado en otros capítulos, y de hecho ya hemos practicado algo, con los controles de servidor, que son los componentes que componen nuestra páginas web y sobre los que aplicaremos el código VB para que realicen las acciones que queramos. Estos controles generan el código HTML para que sea correctamente visible por los navegadores. Tenemos dos grupos de controles:

- **Controles HTML de servidor.** Son los controles equivalentes a los elementos estándar HTML. Son los idóneos para los programadores web que prefieran trabajar con los objetos de siempre, es decir con los controles HTML básicos. Son perfecto para cuando queramos migrar páginas hechas con formularios HTML a formularios ASP.NET
- **Controles Web.** Son similares a los anteriores pero proporcionan un modelo de objetos mucho mas amplio y versátil. Muchos mas métodos y eventos para poder controlar mucho mejor nuestras aplicaciones web y además, muchos de ellos se parecen a los controles de Windows.

Por ejemplo un control de calendario de ASP.NET:



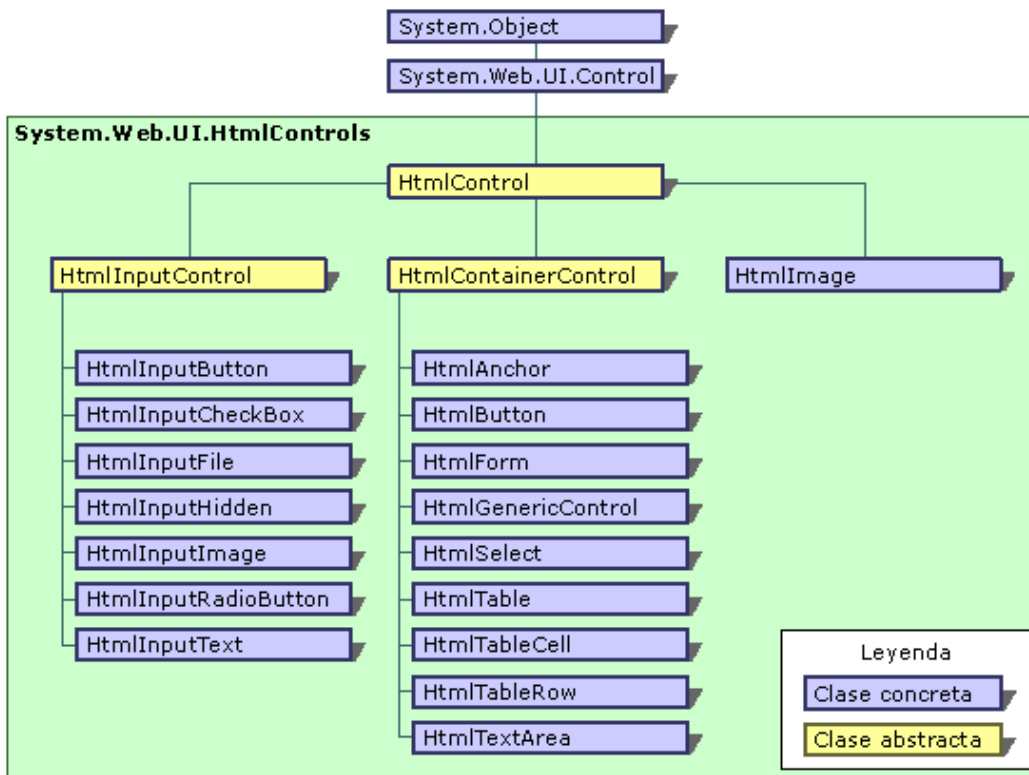
< enero de 2008 >						
lu	ma	mi	ju	vi	sá	do
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Es muy parecido a los controles que habitualmente utilizamos en los programas Windows. Es un control Web de ASP.NET, los controles estándar HTML son mucho mas simples. El crear esto con HTML sería un trabajo enorme, en cambio gracias a ASP.NET incorporar un calendario para que el usuario seleccione una fecha es algo totalmente trivial.

Controles HTML de servidor

Los controles de servidor HTML están definidos en el espacio de nombres "System.UI.HtmlControls", fíjate en este gráfico y los objetos que contiene:

Formularios Web. La clase Page



2. Formularios HTML

Veamos con detalle los formularios HTML. Como hemos comentado antes son perfectamente válidos en el mundo ASP.NET pero el nuevo entorno incorpora los mismos pero mas avanzados, de momento veamos cómo son los formularios HTML estándar. Veamos ahora los principales controles

2.1 La etiqueta <form>

Los formularios están definidos por la etiqueta <form> dentro del código HTML, por ejemplo este es el código de un formulario:

```
<form method="POST" runat="server" action="destino.aspx">
  <input type="text" name="T1" size="20">
  <input type="submit" value="Enviar" name="B1">
  <input type="reset" value="Restablecer" name="B2">
</form>
```

La etiqueta <form> es una etiqueta contenedora porque delimita a un grupo de controles de formulario. Es decir tenemos varios controles de texto y un botón de enviar, todos esos controles están contenidos del formulario. Así que todos los datos que se escriban dentro de los controles de formulario se enviarán a la página destino cuando le pulsemos para enviar los datos. Ojo, si ponemos algún control fuera de las etiquetas

`<form>` y `</form>` ese dato no irá y podría producirnos problemas porque no podríamos recoger esa información.

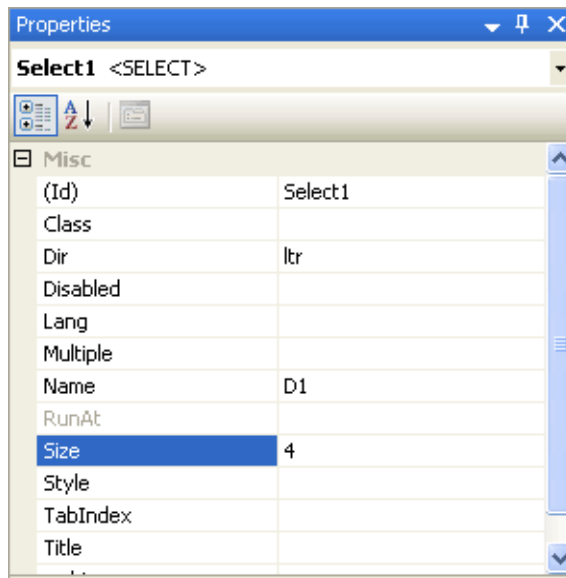
Hay dos indicadores o atributos importantes en la etiqueta del formulario:

- **Action**, indica la página destino, donde se enviarán los datos del formulario. Luego tendremos otras instrucciones para recoger estos datos.
- **Method**. Especifica la forma de enviar la información que hay dentro del formulario. Hay dos formas que veremos mas adelante.

Y los elementos que podemos utilizar en estos formularios

Elemento	Descripción	Propiedades mas importantes	Código HTML
Cuadro de texto	Casilla para introducción de textos	Size, Value, Name	<code><input type="text"></code>
Área de texto	Cuadro de varias líneas para introducir textos	Rows, Cols, Name	<code><textarea rows="3" name="S1" cols="32"></textarea></code>
Casilla de verificación	Casilla para marcar un estado	name, value, checked	<code><input type="checkbox" name="C1" value="ON"></code>
Botón de opción	Selecciona una opción entre varias	name, value, checked	<code><input type="radio" value="V1" name="R1"></code>
Cuadro de Lista	Muestra una lista o un cuadro desplegable	name, size, value, selected, multiple	<code><select size="1" name="D1"></select></code>
Botones	Establecen la acción enviar, restablecer u otra acción	type, value, name	<code><input type="submit" value="Enviar" name="B1"></code>
img	Inserta una imagen	name, size	<code><input type="img" name="F1" size="20"></code>
Invisible	Campos no visibles ni editables	name, value	<code><input type="hidden" name="F1" value="20"></code>

Todas las propiedades de estos controles las tenemos a la derecha abajo, en la ventana de propiedades. Así podemos ponerlas ahí o escribirlas directamente en el código HTML:



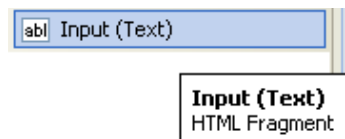
Cuadro de texto

Nombre:

Es el cuadro tradicional de introducción de textos. En HTML se denomina:

```
<input type="text" name="T1" size="32">
```

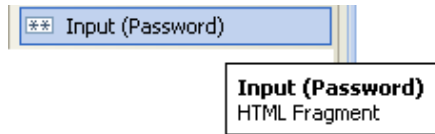
Es decir un input de tipo "text" como propiedades mas importantes como vemos en el ejemplo son el nombre y su tamaño. El tamaño hace referencia a su visualización en pantalla ya que puede almacenar mas caracteres al hacer un "scroll" lateral a medida que se escribe. Es uno de los controles de "input" o introducción de datos de nuestra barra de controles HTML:



El nombre "T1" identifica a esta entrada de texto. Es imprescindible que le indiquemos un nombre único a cada elemento del formulario sino no podremos acceder a su valor. Cuando el usuario escribe algún dato éste se almacena en la propiedad "value", ésta se lo podemos indicar inicialmente en "Valor Inicial" así como el "Ancho de caracteres".

Contraseña

En realidad es una variante del anterior pero lo tenemos en el cuadro de los controles como uno independiente:



Si escribes en este cuadro verás la diferencia respecto al otro::

Contraseña:

Es una variante hace que los caracteres escritos no se muestren en pantalla, obviamente utilizado para escribir contraseñas. En este caso el funcionamiento es igual pero la descripción html es:

```
<input type="password" name="T1" size="32" value="secreta">
```

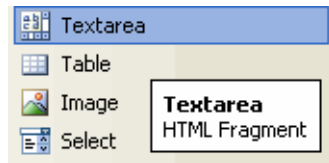
Área de texto

Observaciones:

Es un cuadro para introducción de textos pero en este caso de múltiples líneas. En HTML se denomina:

```
<textarea rows="3" name="S1" cols="32">Contenido del cuadro de texto</textarea>
```

Como propiedades tenemos las que indican su tamaño: "rows" para filas y "cols" para columnas.



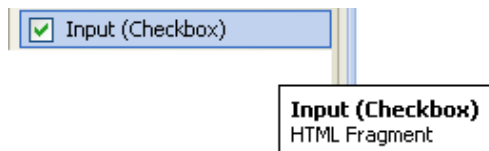
Igual que en el caso anterior tenemos su nombre obligatorio que identificará de forma única a este elemento. En la parte inferior tenemos la posibilidad de indicar un valor inicial y debajo los valores para el número de filas y de columnas. Cuando el usuario escribe algún dato éste se posiciona entre las etiquetas HTML `<textarea>` y `</textarea>`

Casilla de verificación

Casado

Es una casilla que puede almacenar dos estados, marcado o no. HTML se denomina:

```
<input type="checkbox" name="C1" value="ON">
```

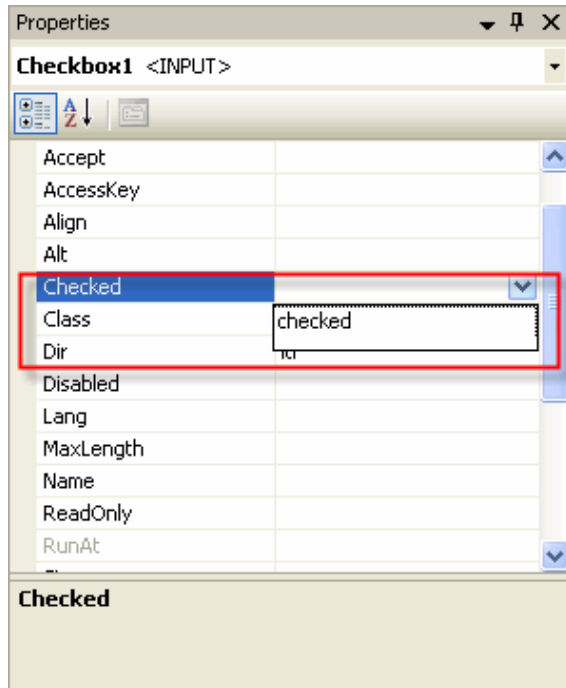


Este elemento admite dos estados, estos dos estados lo indica la presencia de una propiedad denominada "checked". El ejemplo anterior pero con la casilla activada sería de esta forma:

Formularios Web. La clase Page

```
<input type="checkbox" name="C1" value="ON" checked>
```

Para asignar sus propiedades utilizaremos la ventana de propiedades, por ejemplo para que esté activado:



La propiedad "name" como siempre obligatoria. A continuación el valor de devolución "ON" y si estará activada la casilla o no. Este control y la propiedad value funciona de una forma particular. Si el usuario activa esta casilla el valor que devuelve este elemento es la cadena de caracteres que le indicamos en la propiedad "value". En el ejemplo:

```
<input type="checkbox" name="C1" value="Si" checked>
```

Tendríamos que evaluarlo de esta forma en el código:

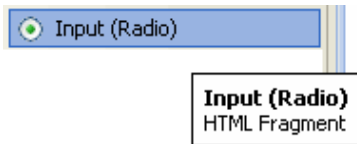
```
if c1="Si" then
    'Entonces está casado
else
    'No está casado
end if
```

Botón de opción

Hombre Mujer

Selecciona una opción entre varias presentadas. Es excluyente, es decir, cuando seleccionamos una de ellas se desactiva la anterior. Por lo tanto sólo una permanece activa. En HTML se denomina:

```
<input type="radio" value="V1" name="R1" checked>
```



Igual que en el caso anterior además de la propiedad "name" tenemos la propiedad "checked" que indica que esta opción está activa. Para crear un grupo de botones de opción que se excluyan entre sí basta con asignarles a todos el mismo valor en la propiedad "name", es decir que tengan el mismo nombre. Puesto que no dispone de texto visible en pantalla debemos escribirlo a continuación del elemento como si fuera texto normal. Como se puede deducir en el código el valor que asignará al "name" será el valor de "value" seleccionado es decir en este ejemplo:

```
<input type="radio" value="Azul" name="R1">Azul<br>
<input type="radio" value="Rojo" checked name="R1">Rojo<br>
<input type="radio" value="Verde" name="R1">Verde</p>
```

Vemos que hay tres botones de opción agrupados porque se llaman de la misma forma "R1" y que está seleccionado el segundo que tiene el literal "Rojo" a continuación del elemento y en la propiedad "value" Por lo gráficamente será:

Azul
Rojo
Verde

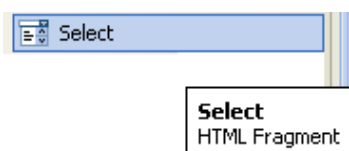
Y el valor asignado a "R1" será "Verde"

Cuadro de lista

Es el elemento que mas posibilidades presenta aunque sigue manteniendo la sencillez de los anteriores.

```
<select size="1" name="D1">
<option value="Enero">Enero</option>
<option value="Febrero">Febrero</option>
<option value="Marzo">Marzo</option>
<option value="Diciembre">Diciembre</option>
</select>
```

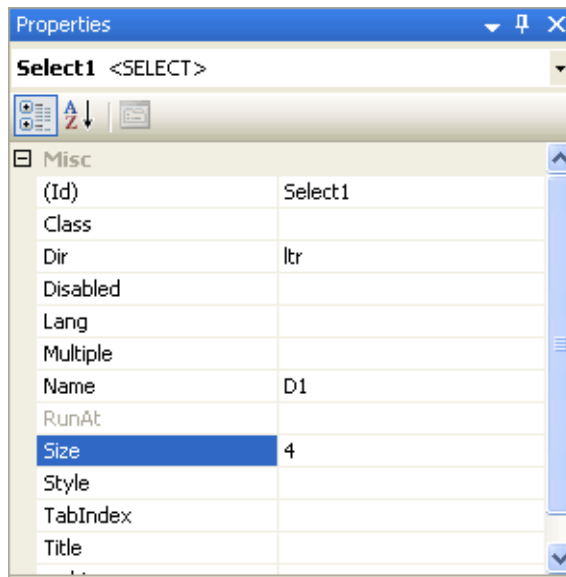
Corresponde al control:



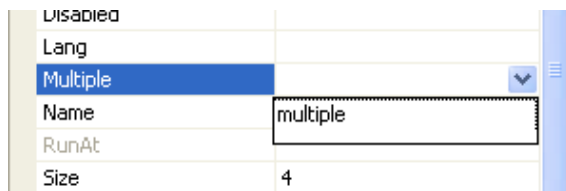
Formularios Web. La clase Page

Este elemento se compone de dos partes, por un lado la que delimita el control y su aspecto `<select name="D1" size="1"></select>` y dentro de estas etiquetas la sucesión de elementos que componen la lista `<option>elemento</option>`.

En la primera parte tenemos una propiedad muy importante y es la del tamaño "size" que nos va a indicar su aspecto en pantalla. Si su valor es 1 se comportará como el ejemplo de la izquierda, es decir como un cuadro desplegable, similar al cuadro combinado de Visual Basic. Si ponemos un número mayor que 1 nos dibujará una lista con tantos elementos como le indiquemos en ese valor. En el ejemplo de la derecha se le ha indicado un `value="4"`. Se muestran en pantalla sólo cuatro líneas aunque tenga mas elementos. Para modificar estos valores haremos doble clic sobre el control para seleccionarlo y luego modificaremos sus propiedades en la parte inferior derecha en la ventana de propiedades:



Un valor de 1 hace que sea un cuadro desplegable y un número superior una lista con esa altura. Por último podemos indicar que "permita selecciones múltiples". Esto hace que incluya la propiedad "múltiple" en la definición del campo.



La segunda parte es la introducción de los elementos, que de momento haremos en la vista del código. Ahí introduciremos uno a uno los elementos de la lista. Escribimos el literal que queramos presentar en la casilla "Opción". Si queremos especificar un valor, es decir que cuando se seleccione nos asigne ya un valor debemos activar la casilla "Especificar valor" e indicarlo. Finalmente la propiedad "Seleccionada" dejará el elemento indicado como activo. Por ejemplo:

```
<select size="4" name="D1">

    <option value="1">Enero</option>

    <option value="2">Febrero</option>

    <option value="3" selected>Marzo</option>
```

Formularios Web. La clase Page

```
<option value="4">Diciembre</option>

</select>
```

Esto creará una lista con cuatro líneas de altura "size=4" y cuatro elementos en su interior. Además el valor "Marzo" aparecerá seleccionado porque tiene la propiedad "selected". Finalmente cuando se seleccione cada línea se asignará a "D1" el valor 1, 2 3 ó 4 según se seleccione Enero, Febrero, Marzo ó Diciembre ya que así lo indica la propiedad value de cada línea.

Esta característica de presentar un valor e internamente devolver otro será tremendamente útil a la hora de trabajar con bases de datos ya que podemos presentar una lista de personas pero el valor interno puede ser su DNI, número de empleado u otro código, con lo que será mas fácil de tratar.

Botones

Que son los controles:



Existen tres tipos de botones: normales, para enviar datos y para restablecer formularios. Se distinguen por la propiedad de "Tipo de botón en su página de propiedades:

Veamos los tres tipos:

- **Botón Enviar.** Es un botón especial que va a provocar que se envíen los datos del formulario a la página indicada en la definición del formulario. El código HTML es:

```
<input type="submit" value="Enviar" name="B1">
```

Sólo necesita un "value" que es el nombre que le presentará al usuario en pantalla además del obligatorio "name"

- **Botón restablecer.** Es otro botón especial que borra el contenido de los elementos del formulario. Digamos que hace un reset de los datos introducidos hasta el momento, similar a si cargamos la página otra vez (tiene poca a ninguna utilidad):

```
<input type="reset" value="Restablecer" name="B2">
```

- **Botón estándar.** No tiene una acción especial asignada. Se puede utilizar para incluir programación, es decir debemos escribir el código que queremos que nos ejecute.:

```
<input type="button" value="Botón" name="B3">
```

En una página compleja podemos tener varios botones con el mismo nombre y en el código los podremos diferenciar por la propiedad "value", por ejemplo supongamos que en la parte inferior de un formulario tenemos:

Los tres se llaman B1, en el código deberíamos distinguirlos de la forma:

Formularios Web. La clase Page

```
select case B1  
  
    case "Insertar":  
        ' Introducir el código necesario  
  
    case "Eliminar":  
        ' Introducir el código necesario  
  
    case "Modificar":  
        ' Introducir el código necesario  
  
end select
```

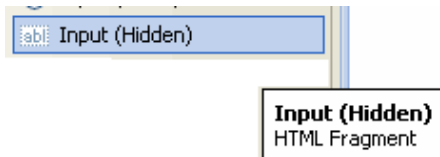
De esta forma no tenemos que poner nombres diferentes y tener que evaluar cual de ellos es el que se ha pulsado

Campos invisibles

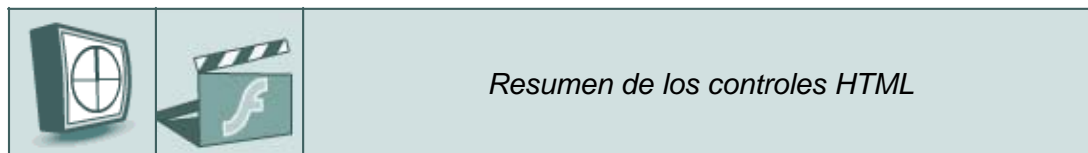
Existe un campo especial que no es visible para el usuario y que no permite la edición de datos: los campos ocultos. Pero... ¿entonces para que sirven?. Su código html es:

```
<input type="hidden" name="indice" size="20">
```

Para insertarlos seleccionamos "Input (Hidden):



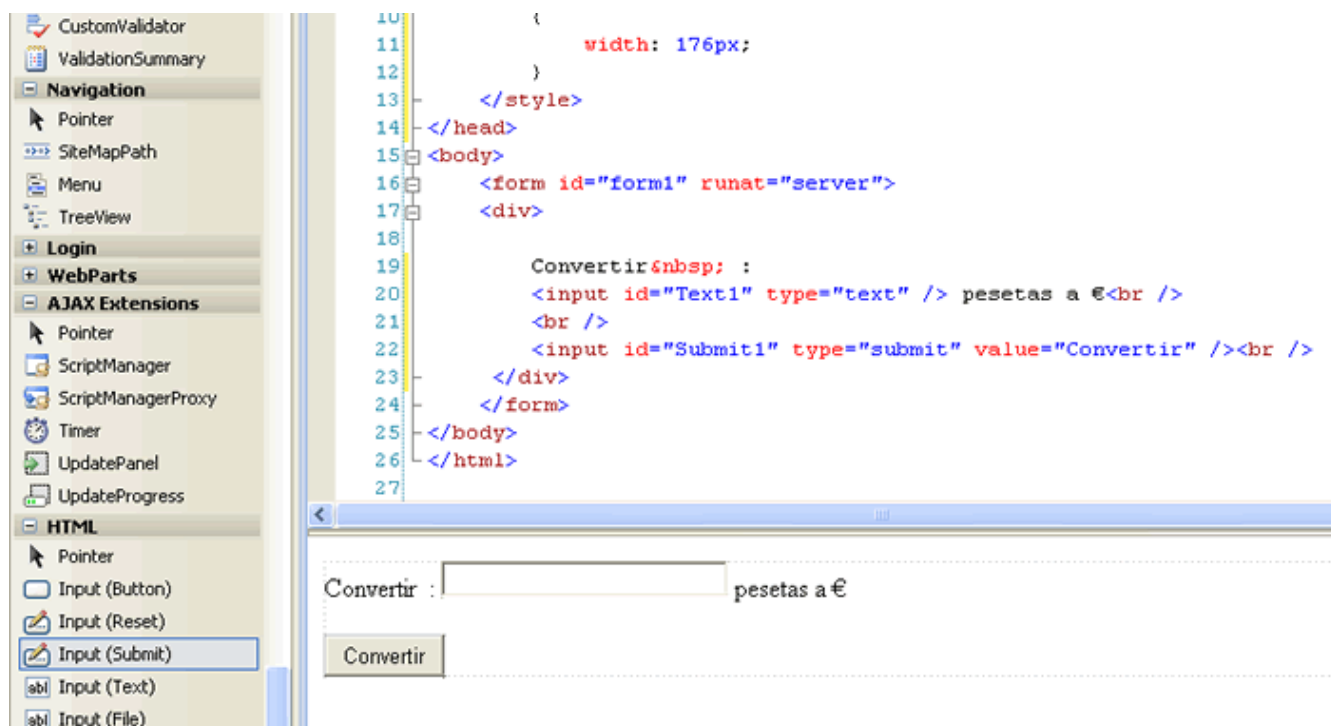
Supongamos que necesitamos varias variables que se cargan en una consulta interna en el formulario. Estas variables son datos que no necesitamos mostrar al usuario y que no debe modificar. Si los incluimos como de tipo "Hidden" conseguimos que al pulsar sobre un botón de tipo "Submit" podamos disponer de estos valores en el formulario destino, pero... ya veremos su uso mas adelante.



2.2 Ejemplo 1

Vamos a realizar un ejemplo completo con una página que nos va a convertir de € a pesetas. Para esto vamos a crear un formulario HTML, así que creamos una página nueva en blanco y ponemos estos controles:

Formularios Web. La clase Page



Los controles los escogeremos la sección HTML ya que son estos, los estándar HTML, los que quiero utilizar. Fíjate en la parte de abajo, simplemente dentro de la sección del formulario he arrastrado un control de tipo "Input (Text)" para introducir el valor que queremos convertir y luego un botón de tipo "Input(Submit)" que provocará que la página se envíe al servidor para que se ejecute. Como ves en el código HTML tenemos:

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

<div>

Convertir:

<input id="Text1" type="text" /> pesetas a €<br />

<br />

<input id="Submit1" type="submit" value="Convertir" /><br />

</div>

</form>
```

Al margen de las etiquetas que solo son visuales o no son código como
(salto de línea) o <div> (agrupa secciones de HTML) tenemos:

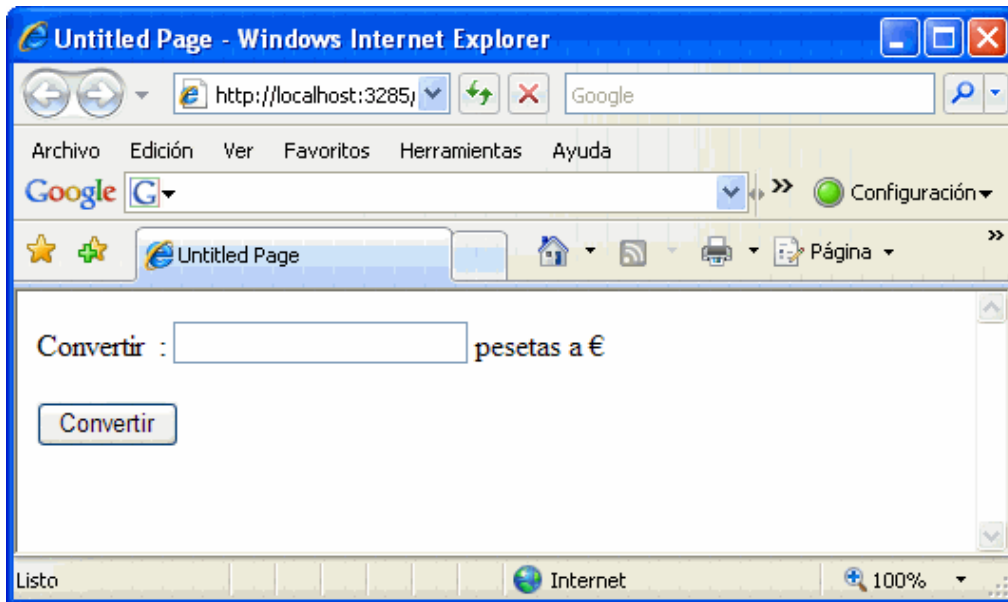
- El formulario definido con el nombre de "form1" y que se ejecutará en el servidor: runat="server"
- Un cuadro de texto llamado "text1"
- Un botón de tipo "submit"

Nos faltan todavía un detalle para que este código nos funcione correctamente. Tenemos que ponerles a los controles una etiqueta para indicar que se deben ejecutar en el servidor "runat=server", por tanto quedará:

Formularios Web. La clase Page

```
<form id="form1" runat="server">
<div>
    Convertir&nbsp;:
    <input id="Text1" runat="server" type="text" /> pesetas a €<br />
    <br />
    <input id="Submit1" runat="server" type="submit" value="Convertir" /><br />
</div>
</form>
```

Vamos a ejecutar esta página pulsando F5. El resultado si no hemos tenido errores será como este:



Pero vamos a lo importante, vete al internet explorer y en la opción de menú "Ver" selecciona "Código fuente", vamos a analizar el código que ha enviado al navegador:

```
<form name="form1" method="post" action="Default.aspx" id="form1">
<div>
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKLTg2NjIxOTM2OWRkcSpLj+XgCCYtJ3et
</div>

<div>
    Convertir&nbsp;:
    <input name="Text1" type="text" id="Text1" /> pesetas a €<br />
    <br />
    <input name="Submit1" type="submit" id="Submit1" value="Convertir" /><br />
</div>
</div>

<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/wEVAwKWtMPfagLz1KGw
```

Lo he hecho con el navegador Firefox, por eso te saldrá algo distinto en pantalla pero el código es el mismo. Lo primero que vemos es que nuestra página pasada por la ejecución del servidor ha cambiado ligeramente. Para empezar han desaparecido las etiquetas de `runat="server"`, ya que son propias de nuestro entorno ASP.NET y el navegador no sabría interpretarlas. Y el cambio mas importante vemos que hay dos campos ocultos (de tipo "hidden") con unos valores ilegibles. Estos campos ocultos almacenan información, aunque no lo parezca, sobre el estado de cada control en la página.

Formularios Web. La clase Page

Por ejemplo, nos habrá pasado muchas veces en un formulario que lo rellenamos y nos hemos equivocado en algo y al darle a enviar nos dice que nos falta seleccionar un dato. Resulta que le damos hacia atrás para rellenar el dato y se han borrado todos los campos que hemos rellenado y tenemos que volver a escribirlos. Estos campos ocultos almacenan entre otras cosas esta información, que son las propiedades de los cuadros de texto, entre otros. Así que podremos tener información persistente entre distintas páginas gracias a que ASP.NET nos va a mantener de forma automática estas variables de estado ocultas.

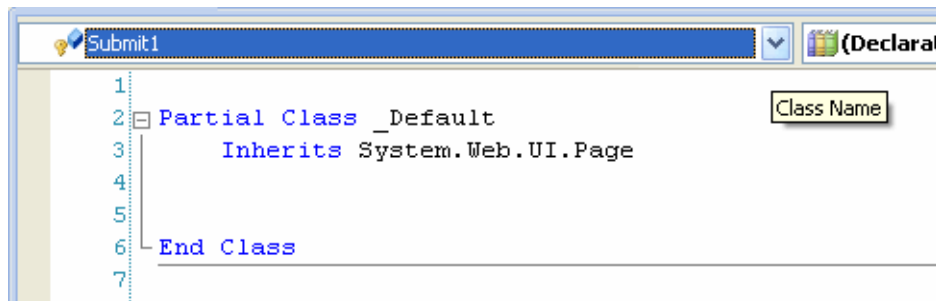
Sigamos con nuestro ejemplo. Ahora debemos escribir el código necesario para hacer la conversión. ¿Dónde pondremos ese código? Exacto: en el evento click del botón. Espero que te hayas acordado. De todas formas ten en cuenta que estamos haciendo todo "a mano" luego lo haremos todo seguido con el entorno de desarrollo y, sobre todo, con los controles ASP que son mucho mas avanzados y cómodos de utilizar.

Antes de continuar un detalle muy importante, nos falta una variable o "algo" donde podamos escribirle el resultado al usuario. Está claro, recogemos el valor en pesetas lo traducimos a euros y lo escribimos al usuario, pero claro, nos hace falta un campo para escribirlo así que debajo del botón de enviar escribiremos esto:

```
<form id="form1" runat="server">
<div>
    Convertir    :
    <input id="Text1" runat="server" type="text" /> pesetas a €<br />
    <br />
    <input id="Submit1" runat="server" type="submit" value="Convertir" /><br />
</div>
<div style="font-weight: bold" ID="Resultado" runat="server"></div>
</form>
```

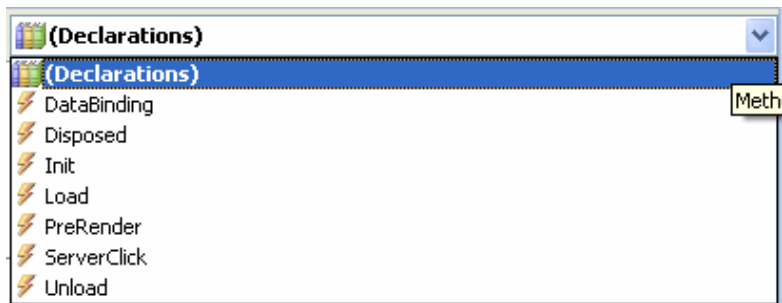
Recuerda que <div> es una etiqueta multipropósito que sirve para agrupar secciones o, para en este caso, asignarle un texto que será el resultado de nuestra operación.

Ahora debemos meter el código pues dentro del evento click del botón. Para esto nos vamos a ir a nuestra página de "código detrás" que en mi caso es la página "default.aspx.vb" y seleccionamos de arriba a la izquierda el botón de enviar que aparece con el nombre que tiene en la propiedad ID. Por eso a partir de ahora le pondremos nombres mas descriptivos.



Hemos seleccionado el objeto a la izquierda, y como novedad, vemos a la derecha la lista de eventos a los que puede responder este control:

Formularios Web. La clase Page



Uno de ellos es el que nos interesa, así que seleccionamos "ServerClick" para poder meter nuestro código. Te recuerdo que todo esto es una prueba y no te preocupes si no entiendes mucho... nos aparecerá el controlador del evento clic del botón:

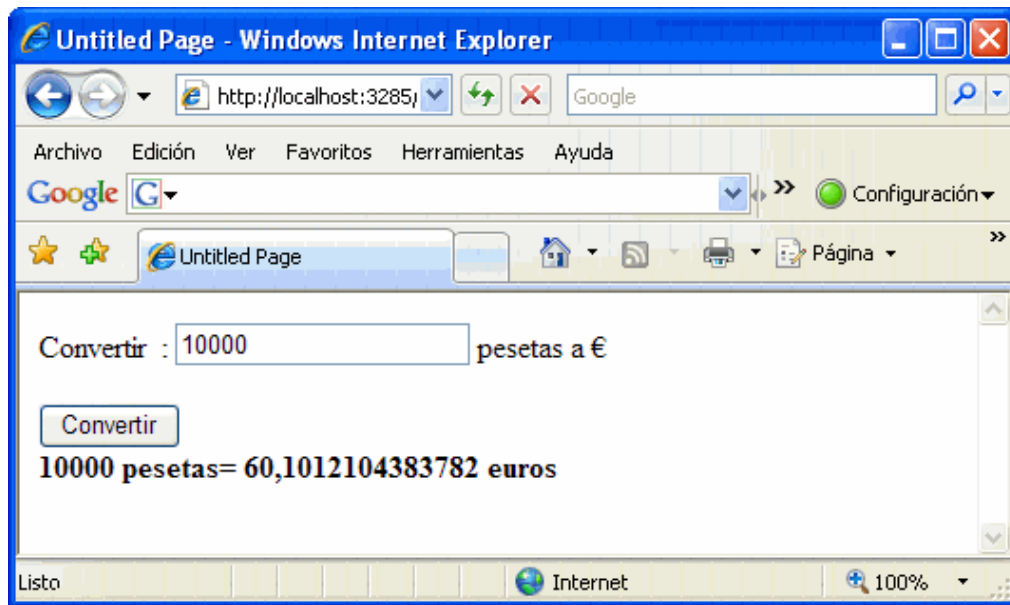
```
2 Partial Class _Default
3     Inherits System.Web.UI.Page
4
5
6     Protected Sub Submit1_ServerClick(ByVal sender As Object, ByVal e As System.EventArgs)
7
8     End Sub
9 End Class
10
```

Y metemos este código para el controlador del evento clic del botón:

```
Protected Sub Submit1_ServerClick(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim pesetas As Double = Val(Text1.Value)
    Dim euros As Double = pesetas / 166.386

    Resultado.InnerHtml = pesetas.ToString & " pesetas= "
    Resultado.InnerHtml &= euros.ToString & " euros"
End Sub
```

Vamos a ejecutar primero la página y luego la explicamos:



¡Ha funcionado! Menos mal, después de tanto código.... Bien, lo que hemos hecho de una forma tan laboriosa ha sido aprender como funcionan los formularios. Ponemos una serie de controles que le indicamos se ejecutarán en el servidor. y posteriormente escribimos el código que queremos que se ejecute en algún evento. En este caso queríamos que al producirse el evento clic en el botón de convertir ejecutase una serie de instrucciones y luego escribiese ese valor en la página así que hemos hecho ese código.

De las instrucciones de visual basic vemos que se han declarado dos variables una "pesetas" en la que asignamos el contenido del cuadro de texto convertido a numérico con la función "Val". Luego hemos definido otra variable "euros" que le hemos asignado directamente la división de las pesetas entre 166.386 para calcular lo euros.

Luego hemos escrito el contenido en la etiqueta "resultado" de tipo "div" que hemos creado antes. Un detalle fíjate que la segunda línea pone:

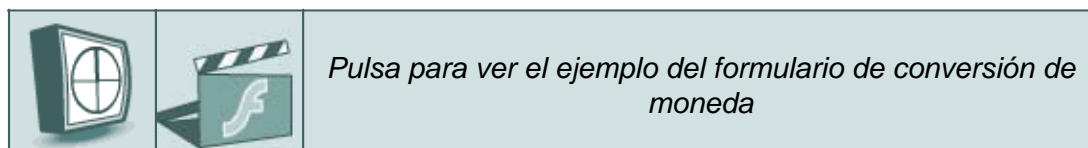
```
Resultado.InnerHtml &= euros.ToString & " euros
```

Te acuerdas de cómo podíamos simplificar las operaciones? por ejemplo $i += 1$ significa $i = i + 1$. Luego esa instrucción equivale a:

```
Resultado.InnerHtml = Resultado.InnerHtml & euros.ToString & " euros"
```

Es decir, añade (concatena) a lo que tiene ese valor el siguiente "&" pero lo podemos simplificar simplemente poniendo "&="

Finalmente hemos utilizado los métodos "tostring" que si recuerdas dijimos que lo tienen la mayoría de los objetos para convertir a "string" los valores "double". No es estrictamente necesaria esta conversión pero como queremos hacer las cosas bien pues lo convertimos antes de escribirlos.



Secuencia de ejecución

Bien, sigamos comprendiendo que porqué se ha ejecutado la página. El primer paso lógicamente es que le llega al servidor una petición de esta página, bien desde el navegador, bien desde nuestro entorno de desarrollo. El servidor web ve que tiene de extensión "aspx" así que sabe que debe analizar el contenido para ejecutar el código antes de devolverla al navegador del usuario que ha solicitado la página.

Al solicitar una páginas ASP.NET se pone en marcha todo el motor de este complejo entorno. Para empezar crea un dominio de aplicación y luego "compila" la página para preparar su ejecución. La página se convierte así en un miniprograma y lanza la página traduciendo el resultado a HTML estándar que es lo que envía finalmente al navegador.

Hay dos fases fundamentales, una es cuando solicitamos por el navegador una página, ahí es IIS quien recoger la solicitud y se prepara a devolverla. Y lo haría si fuera por ejemplo, una página HTML normal, pero al encontrarse que tiene de extensión ".aspx" sabe que primero debe procesarla así que IIS le manda la página al motor "ASP.NET" para que la procese antes de enviarla.

2.3 Ejemplo 2

Mas que un nuevo ejemplo vamos a mejora el que hemos visto antes. Lo que nos va a permitir este ejemplo es seguir aprendiendo instrucciones de código ASP.NET, que aunque luego se pondrán de forma casi automática nos viene muy bien para seguir aprendiendo.

La primera mejora que vamos a incluir es la posibilidad de seleccionar distintas monedas para que nos traduzca, esto nos hace mas completo el código porque tendremos que realizar una operación u otra dependiendo del tipo de conversión que seleccione el usuario.

Para permitir al usuario seleccionar el tipo de conversión pondremos un control HTML de servidor de una lista de elementos:

```
<select name="lista_monedas" style="width: 205px" size="4">  
  <option selected="">Dólar</option>  
  <option>Euro</option>  
  <option>Yen japones</option>  
</select>
```

A la izquierda vemos el control, y a la derecha su código HTML: <select> para las listas.

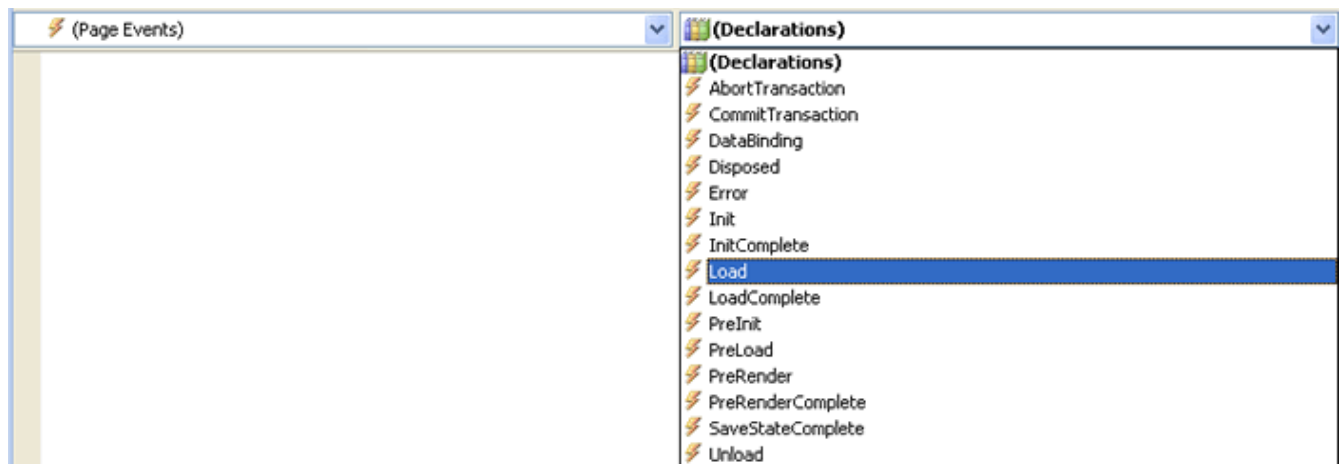
Vamos a poner esta lista pero vacía en nuestro formulario, pondremos una lista vacía, como ya sabemos el código HTML podemos escribirla a mano o con el editor gráfico, el resultado en el formulario sería:

Formularios Web. La clase Page

```
<form id="form1" runat="server">
<div>
    Convertir&nbsp;  :
    <input id="Text1" runat="server" type="text" /> pesetas a €<br />
    <select id="lista_monedas" runat="server" />
    <input id="Submit1" runat="server" type="submit" value="Convertir" /><br />
</div>
<div style="font-weight: bold" ID="Resultado" runat="server"></div>
</form>
```

Podemos rellenar esta lista manualmente desde el editor o, mucho mejor, podemos escribir código VB para añadir elementos a una lista. Para realizar esto debemos rellenar esta lista justo cuando se cargue la página... ¿hay un evento que se ejecute al iniciarse la página? Pues si, y lo utilizaremos a partir de ahora mucho, podremos ejecutar una serie de instrucciones en el momento que se cargue la página así la llenaremos con los datos necesarios antes de mostrársela al usuario. Imagina una consulta a una base de datos, esa página tendrá que hacer uso de este evento para que cuando se cargue, realice la consulta a la base de datos y los escriba dentro de una tabla.

Busquemos este evento, vamos a la vista del código vb (default.aspx.vb), seleccionamos a la izquierda la página y desplegamos los eventos disponible en la página:



Así que seleccionamos a la izquierda el objeto, que en este caso es la páginas y a la derecha seleccionamos el evento en el que queremos que se ejecute el código. Hacemos clic en "Load" y habremos creado un controlador para el evento "load" de la página, es decir ahí pondremos las líneas que queremos que se ejecuten cuando se cargue la página.

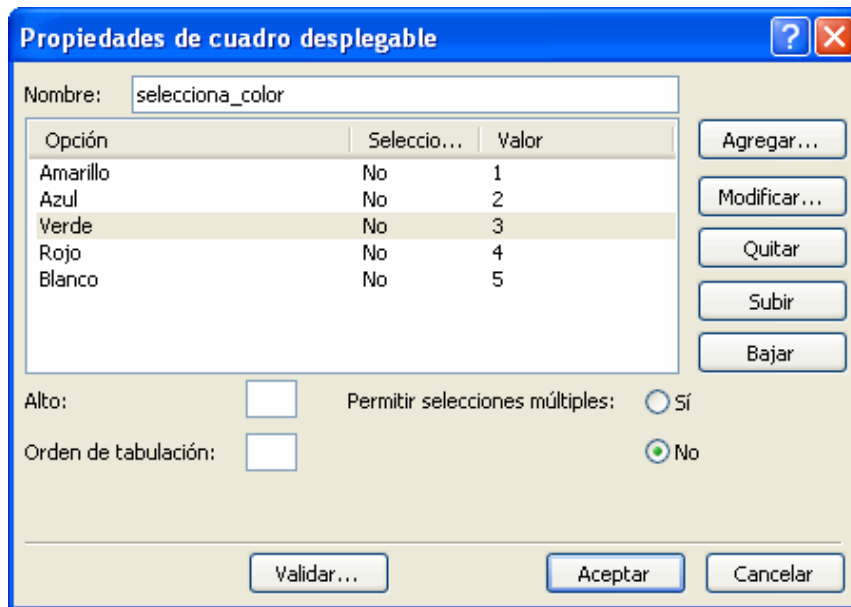
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Me.IsPostBack = False Then
        lista_monedas.Items.Add("Euro")
        lista_monedas.Items.Add("Dólar")
        lista_monedas.Items.Add("Yen japones")
    End If
End Sub
```

Aquí volvemos a aprender varias cosas nuevas muy importantes y que serán habituales en nuestras páginas. Por un lado hemos utilizado la propiedad "items" para añadir elementos a la lista, cuando ASP.NET ejecute este código lo que hará será traducirlo a HTML con <option>...</option>.

Formularios Web. La clase Page

Si al cargarse la página se añaden esos elementos en la lista probablemente no de problemas porque si la volvemos a cargar con otra consulta se volverá a disparar el evento load y volverá a cargar la página, con lo que nos repetiría continuamente la inserción de esas líneas. Para solucionar esto vamos a utilizar si la propiedad "IsPostBack" está activada o no. Se activa cuando se pide esta página con una solicitud del botón, por tanto cada vez que le demos al botón para que nos haga la operación "isPostBack" será "true" porque se está solicitando la página. Así que si le decimos que "carga estos elementos cuando no haya habido una solicitud de página", que traducido sería al cargarse por primera vez porque las siguientes serían porque el usuario ha pulsado el botón de enviar para que le calcule una moneda.

Esto ya va tomando forma, pero vamos a realizar otra mejora. Las listas pueden almacenar un valor, así cuando se selecciona devuelve ese valor en lugar del texto seleccionado. Me explico, fíjate como pongo los valores en esta lista y cómo lo asocio a unos valores:



Esa pantalla nos muestra cómo creamos una lista en Frontpage, por ejemplo, y le indicamos unos elementos y el valor que nos devolverá cuando se seleccione un elemento. Por ejemplo, si el usuario selecciona el color Verde, devolverá el valor 3.

Nos vamos a ayudar de esta propiedad para que cuando el usuario seleccione una moneda devolvamos ya el tipo de cambio. El código ahora será un poco mas complejo porque necesitaremos crear un objeto "listitem" en cada elemento:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Me.IsPostBack = False Then
        'Nota: los valores de cambio son de ejemplo
        lista_monedas.Items.Add(New ListItem ("Euro", "166.386"))
        lista_monedas.Items.Add(New ListItem ("Dólar", "120"))
        lista_monedas.Items.Add(New ListItem( "Yen japones", "0.1"))
    End If
```

Lo que hemos añadido ahora con "items.add" son elementos "listitem" que nos permite identificar tanto el literal a mostrar como el valor que va a devolver.

Formularios Web. La clase Page

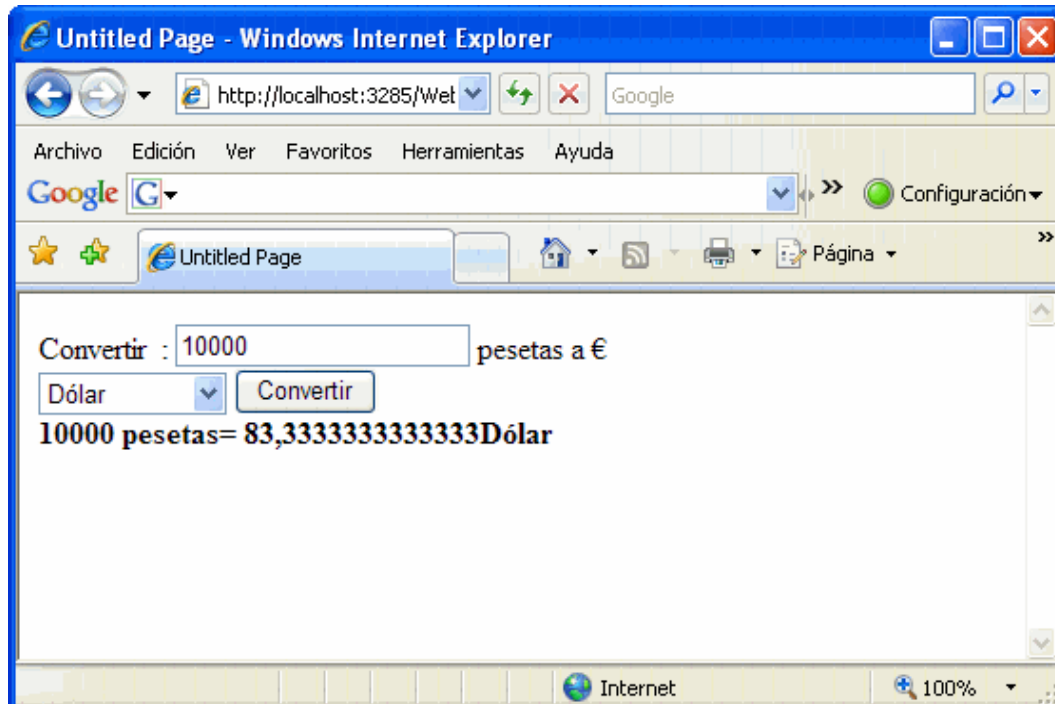
Fíjate que ahora la operación va a ser muy sencilla, porque el resultado va a ser el valor que metamos en el cuadro de texto del formulario y luego multiplicarlo por el valor que devuelve la lista ¿no?. Me explico, si el usuario introduce el valor 10000 en el cuadro de texto y selecciona la moneda "euro" lo que le proporcionaremos al evento load de la página va a ser por un lado en el cuadro de texto el valor 10000 y por otro la lista devolverá el valor "166.386" que es el correspondiente al valor que ha seleccionado el usuario. Hagamos esa multiplicación y veamos el resultado:

```
Protected Sub Submit1_ServerClick(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim cantidad As Double = Val(Text1.Value)

    Dim seleccion As ListItem = lista_monedas.Items(lista_monedas.SelectedIndex)

    Dim cambio As Double = cantidad / Val(seleccion.Value)
    Resultado.InnerHtml = cantidad.ToString & " pesetas= "
    Resultado.InnerHtml &= cambio.ToString & seleccion.Text
End Sub
```

Una vez introducido, ejecutamos la página...



Bien, esto funciona. Según seleccionamos un elemento de la lista nos hace la operación con él.

[Pulsa aquí para descargar esta página de ejemplo.](#)

Veamos ahora la explicación. La gran diferencia respecto al caso anterior es que tendremos que extraer ese valor que se ha seleccionado de la lista. Pero antes vamos a ver que ha hecho ASP.NET al ejecutar este código:

```
lista_monedas.Items.Add(New ListItem("Euro", "166.386"))
lista_monedas.Items.Add(New ListItem("Dólar", "120"))
lista_monedas.Items.Add(New ListItem("Yen japones", "0.1"))
```

Formularios Web. La clase Page

Al cargarse la página por primera vez ha debido traducir ese código a HTML, vamos a abrir la página en el internet explorer y luego vemos el código de la página:

```
<select name="lista_monedas" id="lista_monedas">
  <option value="166.386">Euro</option>
  <option value="120">Dólar</option>
  <option value="0.1">Yen japonés</option>
</select>
```

Como ves lo ha traducido creando las etiquetas HTML estándar para representar una lista con "<select></select>". Además ves que muestra un texto y devolverá el valor del elemento seleccionado.

Y ahora veamos la diferencia con el cálculo del anterior ejemplo. El cálculo como hemos visto antes tiene ahora:

```
Protected Sub Submit1_ServerClick(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim cantidad As Double = Val(Text1.Value)

    Dim seleccion As ListItem = lista_monedas.Items(lista_monedas.SelectedIndex)

    Dim cambio As Double = cantidad / Val(seleccion.Value)
    Resultado.InnerHtml = cantidad.ToString & " pesetas= "
    Resultado.InnerHtml &= cambio.ToString & seleccion.Text
End Sub
```

Empecemos declarando una variable "cantidad" donde metemos el contenido del cuadro de texto del formulario convertido a numérico: val (text1.value). Luego declaramos un elemento de lista para recoger el valor seleccionado por el usuario: nombre y contenido. ¿cómo sabemos qué elemento está seleccionado? Muy fácil, con "SelectedIndex" por tanto Lista_monedas.Items (lista_monedas.SelectedIndex) significa devuélveme el elemento que está en la posición x: Lista_monedas.Items (x), x debe ser uno de la lista, que en ese caso será el seleccionado por el usuario que es "lista_monedas.SelectedIndex". Hemos conseguido por tanto asignar a "seleccion" el elemento seleccionado por el usuario.

Ahora ya solo nos falta realizar el cálculo. La variable "seleccion" que es de tipo "ListItem" tiene dos propiedades fundamentales: "Value" que me devuelve el valor asignado y "text" que es el texto que el usuario selecciona de la lista. Por tanto en:

```
Dim cambio As Double = cantidad / Val(seleccion.Value)
```

Hemos asignado a cambio la división de "cantidad" y el valor de la selección del usuario val (seleccion.value). Finalmente escribimos el resultado

```
Resultado.InnerHtml = cantidad.ToString & " pesetas= "
```

```
Resultado.InnerHtml &= cambio.ToString & seleccion.Text
```

Pero escribimos "seleccion.text" al final para indicarle el tipo de cambio que ha elegido:

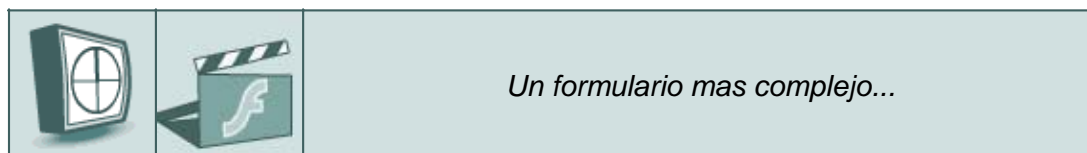
Convertir : 3000 pesetas a €

Dólar ▼ Convertir

3000 pesetas= 25Dólar

Formularios Web. La clase Page

Y ya está que no es poco. Como ves, es laborioso pero es la mejor forma de ir aprendiendo. ASP.NET es muy extenso y es mejor escribir mucho código para aprender la sintaxis, los objetos...



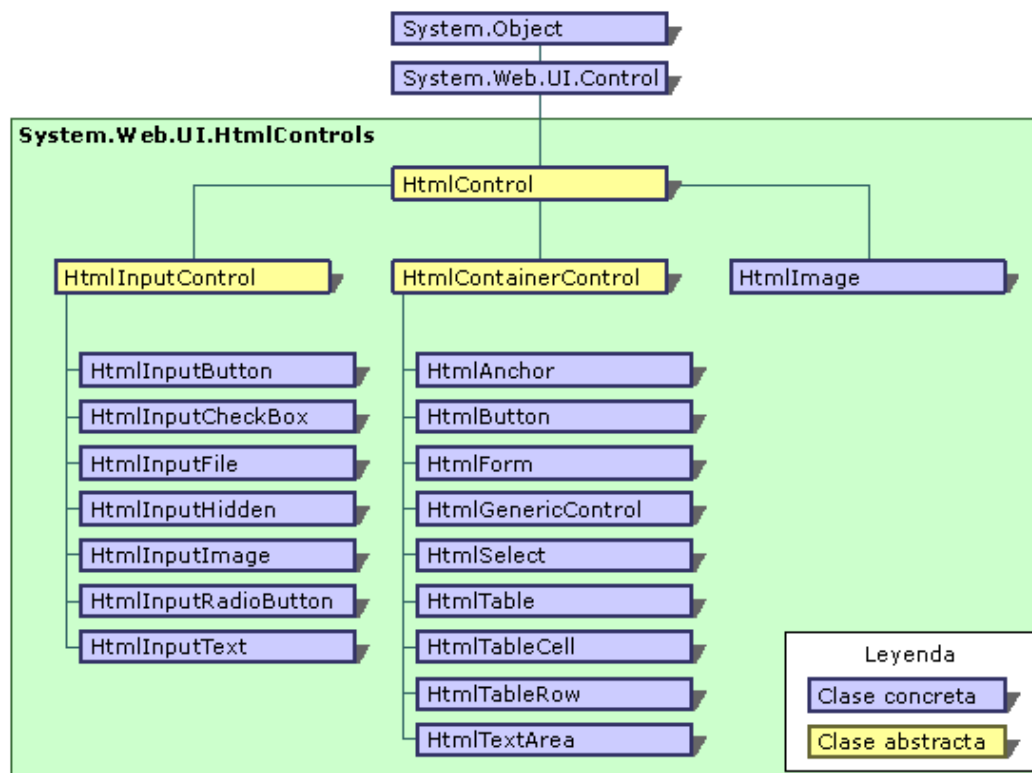
Estilos

No hemos hablado nada de los estilos, mas que nada por los veremos mas adelante pero ya que estamos escribiendo código podríamos mandarle el resultado con algún estilo para mejorar la apariencia.

3. Clases de los controles HTML

Los controles HTML son digamos "los de siempre", los que nacieron con la Web y son por tanto, **limitados**, ya que se diseñaron para cumplir unas necesidades muy sencillas. Entonces si ASP.NET tiene sus propios controles mas potentes **¿porqué ver los antiguos?** Pues porque hay que conocerlos, es necesario saber cómo funcionan los sencillos para luego pasar de pleno al mundo ASP.NET. Además, si alguno de vosotros quiere pasar a esta mundo con páginas ya construidas, le será mas fácil ya que utilizará los mismos controles, aunque con la capacidad añadida de la programación que tienen al pasar al mundo .NET

Veamos un aspecto mas técnico de las clases HTML. Para empezar veamos su modelo de objetos:



Los controles de la primera columna son los de introducción de datos: cuadros de texto, casilla de verificación, ... y se heredan de la clase "HtmlInputControls" y a su vez de "HtmlControl" del espacio de

nombres "System.Web.UI.Control". La segunda columna tiene los controles mas genéricos, aunque hay alguno de introducción de datos también están otros como las tablas, los enlaces (HtmlAnchor), ... y fiñemente y por libre las imágenes.

Y ahora que hemos visto parte de sus controles trataremos sus eventos. Los controles HTML proporcionan principalmente dos eventos: "ServerClick" y "ServerChange". El primero de ellos como habrás podido deducir es simplemente cuando el usuario hace clic en el control y se procesa en el servidor, por ejemplo, un clic en un botón.

Un ejemplo lo podemos ver con el objeto "HtmlAnchor" que no es mas que la etiqueta: `hola` que representa un hipervínculo. Tenemos dos formas de tratar este control, una de ellas es poner la propiedad "href" del control que apunte una URL, en cuyo caso el control se comportará exactamente que su equivalente en HTML ya que únicamente le ponemos la propiedad del enlace a la página que apunta. La ventaja es que como ponemos el enlace en nuestro código éste es dinámico y podremos poner el que nosotros queramos y no uno fijo como se pondría en un editor. La otra forma de tratar un control como este es crear un controlador para el evento "HtmlAnchor.ServerClick", en este caso cuando se haga clic se producirá un "postback" y pasaremos a tener el control en nuestro código.

El evento "ServerChange" responde cuando se ha hecho un cambio o una selección en el control. Pero como desventaja no podremos tratarlos hasta que no se haya hecho un "submit" o un envío de la página.

3.1 Eventos avanzados con el control HtmlInputImage

Hemos hecho controladores de eventos en ejemplos anteriores pero no hemos reparado apenas en la declaración de los procedimientos de estos controladores, veamos algunos detalles de ellos con un control HtmlInputImage

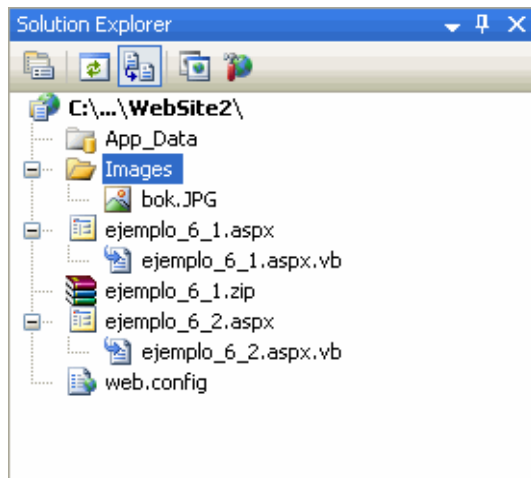
Cada evento que queramos controlar necesita dos tipos de información: El primer parámetro indica el objeto que produce el evento, el segundo parámetro es un objeto especial que proporciona información adicional sobre el evento del objeto, de ahí que se llame de tipo "eventargs", es decir argumentos o parámetros del evento. Recordemos el controlador del evento clic de un botón:

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

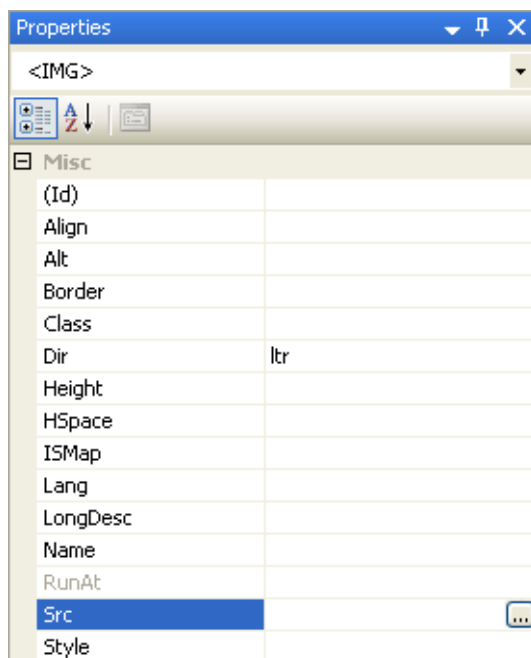
```
End Sub
```

Primero vemos que se llama igual que el nombre (ID) del botón y seguido del evento: Button1_Click. Y seguido vemos los dos parámetros comunes en los controladores de eventos: por un lado el objeto o control que causa el evento: sender as Object y por otro lado una variable llamada "e" que es de tipo "System.EventArgs" que incorpora información adicional sobre el objeto que ha producido el evento. Finalmente "Handles Button1.Click" nos indica que este procedimiento se va a ejecutar "handles" cuando se produzca un evento clic en el botón: "Button1.clic"

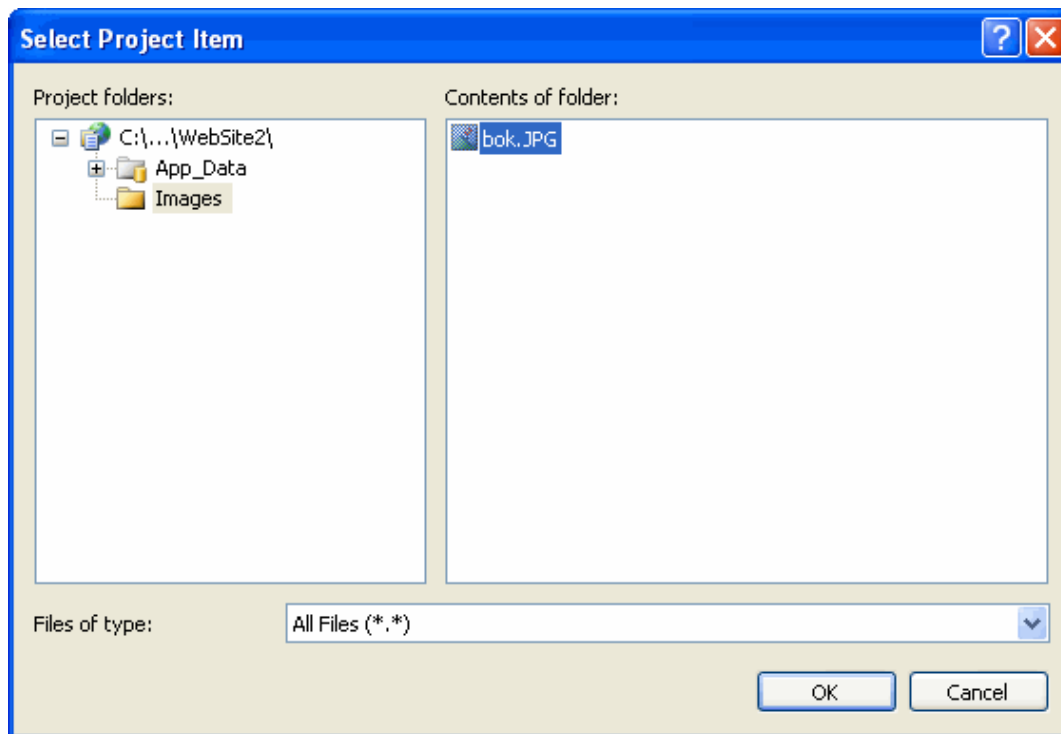
Vemos un ejemplo de cómo se envía información adicional en el segundo parámetro. Crea una página que contenga una imagen dentro del área del formulario. Antes de insertar la imagen podemos ir organizando nuestro web de ejemplos. Crearemos una carpeta que se llama "images" para meter ahí todos los gráficos que vayamos incorporando a nuestro web:



Una vez creada cogeremos la imagen que queremos incorporar y mediante el cortar-pegar la introducimos en esa carpeta. Así cuando pongamos un control HTML de tipo imagen tendremos ya la imagen a mano. Recuerda que vamos a introducir un control HTML, así que debes cogerlo de la sección de controles de abajo del todo en el grupo "HTML" y no del grupo "estándar". Lo ponemos en nuestra página y vemos sus propiedades a la derecha en la ventana de propiedades:



"Src" es la propiedad para asignarle un gráfico así que se lo asignaremos explorando nuestro web que, como ya tenemos la imagen, será inmediato:

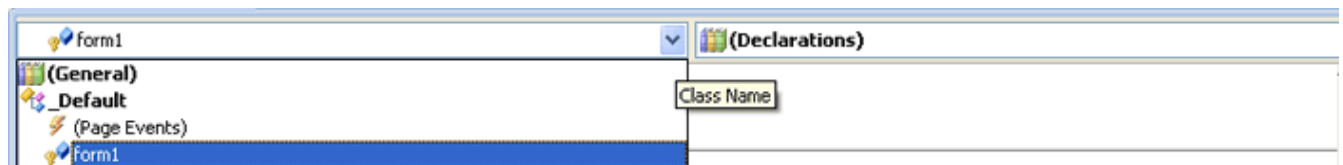


Veamos ahora el código de este control en nuestra página:

```

```

Este control en principio es "pasivo" es decir no es de tipo "input" como un botón un cuadro de texto porque no envía información ni nada parecido. De hecho si nos vamos a la página del código veremos que no tiene siquiera el evento de "ServerClick". Recuerda, en la página de código "ejemplo.aspx.vb" selecciona arriba a la izquierda el objeto y a la derecha tendremos los eventos:

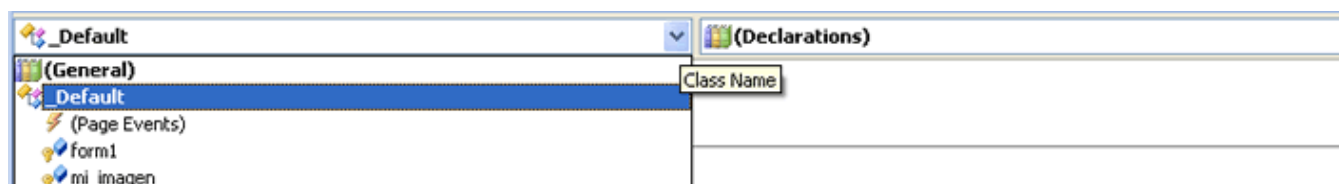


Bueno, de hecho es que ni siquiera tenemos el objeto en la lista de la izquierda. Vamos a mejorar esto, le pondremos un nombre a nuestro control en el código HTML y le diremos que se debe ejecutar en el servidor:

```

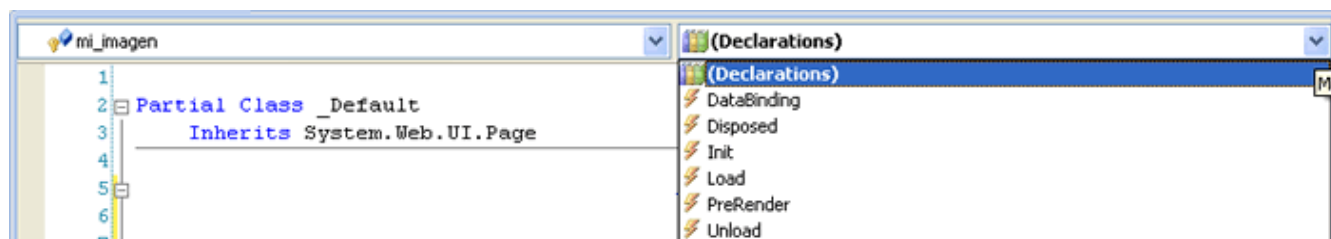
```

Es un gran paso, por lo menos la página ya sabe que existe, lo vemos en el desplegable de los objeto en nuestra página de código:

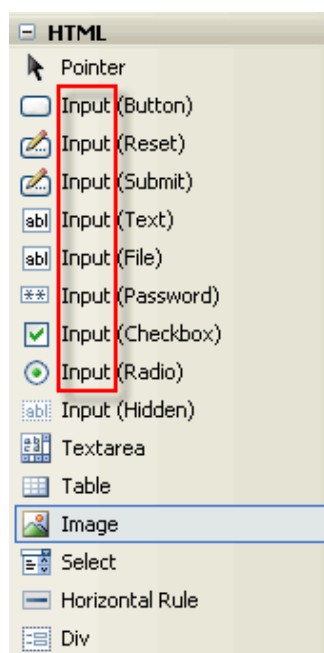


Formularios Web. La clase Page

Pero si nos fijamos en los eventos de este control:



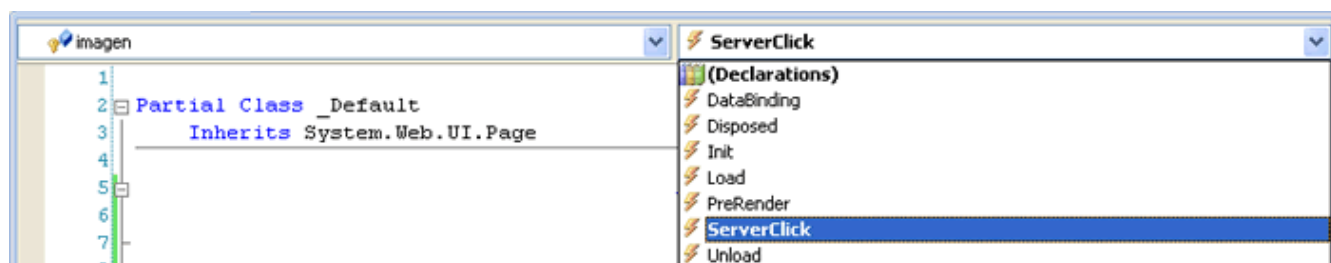
Vaya, no tenemos el famoso "ServerClick", la razón es por la que te he dicho antes, al no ser un control de entrada de datos no podemos controlar este evento. Fíjate en la colección de controles HTML:



Ves lo que te comentaba, los que he marcado aceptan datos y son por tanto de tipo Input, los demás: imágenes, tablas, ... son solo para mostrar datos. Bueno, como hemos navegado mucho por Internet sabemos que en muchos sitios web hay imágenes que cuando las pulsas van a un sitio web, por lo tanto si que aceptan algo de programación. Ciertamente, pero esto se hace con un pequeño truco que permite este control y es modificar su tipo de control: lo convertiremos a tipo "input" en lugar de "imagen":

```
<input type="image" runat="server" id="imagen" src="Images/bok.JPG" />
```

Es otro tipo más de "Input", en este caso de imagen, visualmente es igual pero si ahora nos vamos a sus eventos:



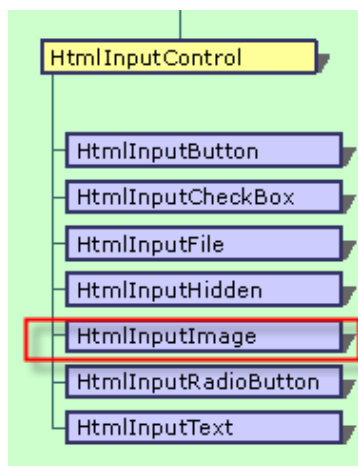
¡Y todo esto para ver que este control tiene un parámetro distinto a los demás! Pues si, así hemos practicado y entendido un poco mejor el tema de los eventos, así cuando lleguemos a la fase fácil en la que nuestro IDE nos pondrá casi todo este código sabremos porqué lo hace y para que sirve. Vemos la declaración de este evento:

```
Protected Sub imagen_ServerClick(ByVal sender As Object, ByVal e As ImageClickEventArgs)Handles imagen.ServerClick
```

```
End Sub
```

¿Y que parámetros tenemos mas en este control? Pues algunos poco útiles ahora pero interesantes para el futuro. Por ejemplo "e.X" y "e.Y" nos devuelve las posiciones X e Y de donde el usuario hizo clic en el control. Luego sabemos exactamente las coordenadas del clic. Las obtenemos del objeto "e" que son los parámetros opciones del control del evento y luego en sus propiedades X y Y: "e.X" y "e.Y".

Una cosa, no nos hemos inventado que el control imagen se pueda convertir así como así a uno de "Input", sino que es una variación del imagen ya que si recuerdas la jerarquía de estos controles HTML teníamos:



Es decir, aunque no lo teníamos como un control inmediato

3.2 La clase HtmlControls

Cada control HTML está heredado de la clase base "HtmlControl", recuerda el gráfico de las herencias que vimos antes. Cada control tiene sus propias características y propiedades, veamos las propiedades mas importantes para ir terminando con estos controles HTML:

Propiedad	Descripción
Attributes	Proporciona una colección de los atributos o propiedades del control y sus valores.
Controls	Proporciona una colección de los controles contenidos en ese control. Por ejemplo una control de tipo "<div>" puede contener varios controles, así que podríamos enumerarlos y referenciarlos con esta colección. Cada objeto devuelto es un objeto "System.Web.UI.Control" genérico.
Disabled	Deshabilita el control si esta propiedad vale "true". El usuario no podrá utilizarlo y no se dispararán los eventos.

Formularios Web. La clase Page

EnableViewState	Deshabilita el control de estado de este control. De esta forma cada vez que se muestra la página no recordará su contenido anterior y se "reseteará" a los valores originales.
Page	Referencia al objeto página: System.Web.UI.Page
Parent	Proporciona una referencia al control que contiene este control, si está colocado directamente en la página proporcionará una referencia al objeto página
Style	Proporciona una colección de estilos CSS para aplicar al control
Tagname	Indica el nombre del elemento HTML: "img", "div", ...
Visible	Si está a "false" el control no estará visible y por tanto no se enviará a la página HTML.

Las propiedades de los controles las podemos establecer en el código o en su etiqueta, por ejemplo podemos poner en la definición de la etiqueta este valor:

```
<img ID="grafico" runat="server" visible="false" ... />
```

Con lo cual ese control estaría oculto (visible="false") pero también lo podemos poner en el evento "load" de la página, es decir, cuando se cargue:

```
Sub page_load  
    grafico.visible=false  
    ...  
end sub
```

3.3 La clase HtmlContainerControl

Hay controles que engloban a otros controles. Por ejemplo, el control <input> es independiente, se define y se termina su definición. Sin embargo hay otros como el ya conocido "<form>" (define al formulario), que se declara, se añaden una serie de controles y por fin se finaliza </form>. Es decir contiene a otros controles: cuadros de texto, listas y botones, por ejemplo.

Si los controles contenedores tenemos dos propiedades adicionales a las que tienen los controles normales "HtmlControls":

- InnerHtml. Es el contenido HTML que tiene el control. Es decir, si dentro de él hay listas y otros controles, con esta propiedad podemos acceder a su contenido
- InnerText. Lo mismo que el anterior pero en texto. Es decir, el texto contenido entre las etiquetas de apertura y cierre del control. <div> textos.... </div>

3.4 La clase HtmlInputControl

Los controles de tipo "Input" ya los conocemos son los que van a contener valores y van a ser susceptibles de controlar sus eventos. Además sabemos que dependiendo del tipo de "input": type="text" (texto), type="image" (imagen), type="submit" (botón) se representa de forma diferente. Las propiedades pues serán las relativas a obtener su contenido:

- Type. Nos devuelve el tipo de control, por ejemplo para un control de tipo <input type="file"> nos devuelve el literal "file"
- Value. Devuelve el contenido del control como una cadena de caracteres o string.

4. La clase Page

Cada página web es una clase generada a partir de "System.Web.UI.Page", y al heredarse de ésta, adquiere todas las propiedades y eventos que podremos utilizar en nuestro programa. Sus propiedades las veremos a lo largo del curso, pero veamos las mas importantes:

Propiedad	Descripción
IsPostBack	Es un valor booleano (true o false) que nos indica si es la primera vez que se ejecuta la página. Por ejemplo, si se carga otra vez porque se haya pulsado un botón de enviar o detectado un evento esta variable será "true". Así que si queremos poner unos valores iniciales a la página los pondremos cuando ispostback sea falso: IF IsPostBack=false then
EnableViewState	Si está asignado a "false" borra el estado de los controles y los deja como si se pintaran de nuevo en la página, es decir no almacena su estado. Por ejemplo si dentro de un cuadro de texto hay un valor "jose" al no almacenar su estado se borrará al recargar la página
Application	Esta colección almacena información que será accesible por todos los usuarios en todos los sitios web de nuestro servidor. Digamos que pueden ser variables globales para nuestro Internet Information Server. Puede servir por ejemplo para contar cuantas visitas se hacen a una página.
Session	Igual que antes pero solo para un usuario. Tendremos variables y propiedades que solo serán accesibles por este usuario en su sesión web.
Cache	Técnica para mejorar el rendimiento de las páginas ya que nos permite almacenar objetos que son costosos de crear en tiempo y así poder reutilizarse en otras páginas si tener que volverse a crear.
Request	Hace referencia al objeto HttpRequest que contiene información sobre la actual petición de la página. Por ejemplo podemos obtener datos del navegador que hace la solicitud: Internet Explorer, FireFox,
Response	Se refiere al objeto HttpResponse que representa a lo que ASP.NET responde o envía al usuario.
Server	Hace referencia al objeto HttpServerUtility que tiene varias tareas o métodos interesantes que veremos mas tarde
User	Si el usuario ha sido "autenticado", es decir ha pasado por un proceso de validación "usuario/contraseña", nos proporcionará información del usuario.

Esta clase es muy importante y tendemos objeto de mucha e imprescindible utilidad.

Veamos ahora los objetos mas importantes de la clase Page que acabamos de comentar, ya que nos van a proporcionar muchas cosas interesantes.

4.1 Objeto Request

Este objeto integrado se utiliza para obtener información del usuario.

Cuando un navegador cliente se comunica con el servidor Web a través del protocolo HTTP, el navegador manda una información especial además de la página que se quiere cargar, esta información puede ser: variables de entorno, información sobre certificados, cookies, formularios, etc. Toda esta información está codificada y se envía a través de cabeceras del protocolo HTTP. ASP realiza las funciones de extraer, decodificar y organizar toda esta información a través del objeto Request y sus diferentes colecciones.

Esta información enviada por el navegador, se puede utilizar para: validar un usuario, obtener información sobre el usuario el navegador, almacenar información para su posterior uso, enviar información al servidor a través de formularios, etc. ASP almacena toda esta información en colecciones del objeto Request. Mediante llamadas del tipo Request.colección se puede obtener de forma sencilla la información que sea necesaria. El objeto Request tiene varias colecciones que veremos en el siguiente apartado

Colecciones del objeto Request

Las colecciones que ofrece el objeto Request del modelo de objetos de ASP contienen toda la información enviada por el navegador cliente al servidor Web. Estas colecciones son estas...

- **ApplicationPath**: Devuelve la ubicación de la página solicitada.
- **Path**: Igual que el anterior pero devuelve también el nombre de la página
- **PhysicalApplicationPath**: Devuelve la ubicación de la página solicitada pero en el disco donde está alojada: c:\datos\pagina.aspx
- **Browser**: Proporciona información sobre el navegador del cliente.
- **Cookies**: valores de las cookies del cliente.
- **TsSecureConnection**: indica de se está utilizando una conexión segura
- **Requesttype**: Devuelve si se está utilizando el método GET o POST
- **URL**: Devuelve la dirección completa enviada al navegador.
- **QueryString**: valores de las variables de la cadena de consulta HTTP enviada.
- **RawURL**: Igual que URL pero se omite el protocolo y el dominio
- **UserHostName**: Devuelve el nombre del equipo que está solicitando la página
- **UserHostAddress**: Devuelve la dirección del equipo que está solicitando la página
- **UserLanguages**: Devuelve el lenguaje del explorador del cliente

Para tener acceso a la información de las colecciones del objeto Request se puede utilizar la siguiente sintaxis general:

```
Request.NombreColeccion(variable)
```

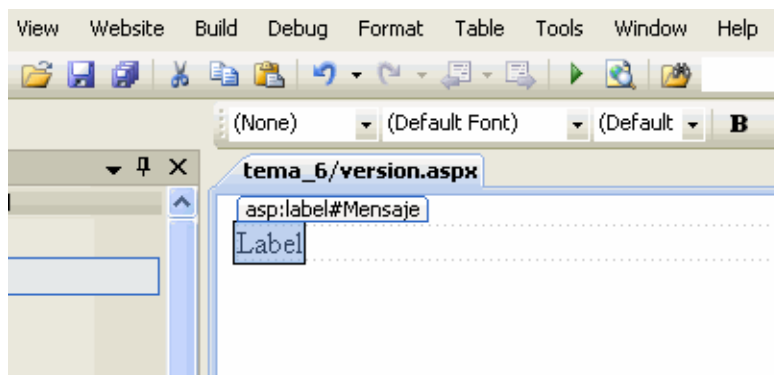
Donde NombreColeccion es el nombre de la colección que se quiere consultar y variable es el nombre de la variable de la colección a la que se quiere tener acceso y recuperar su valor.

Las variables contenidas en las colecciones del objeto Request son únicamente de lectura.

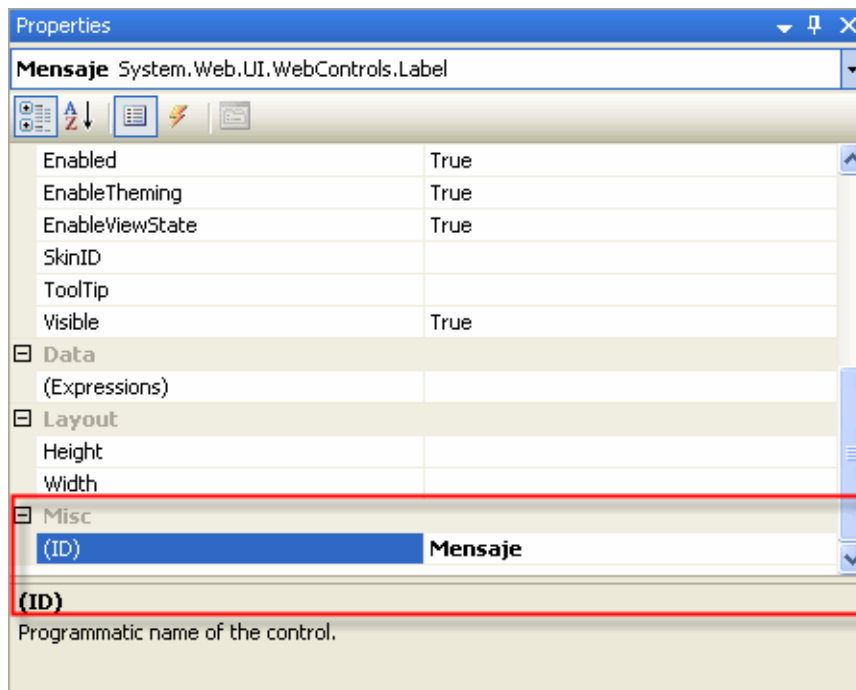
Ejemplo

Vamos a crear un ejemplo que nos devuelva el tipo de explorador del cliente y su versión.

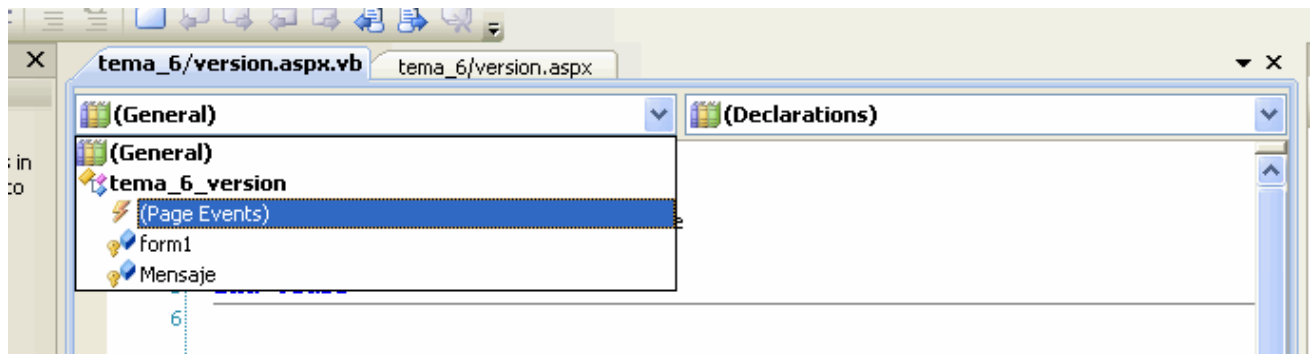
1. Crea una página web y llámala version.aspx. Ponle un control de tipo "Label" y en su propiedad TD le pones el nombre "mensaje". Es un control Label de ASP.NET de los que tenemos arriba no de los HTML que hemos visto anteriormente en este capítulo:



Y la propiedad:

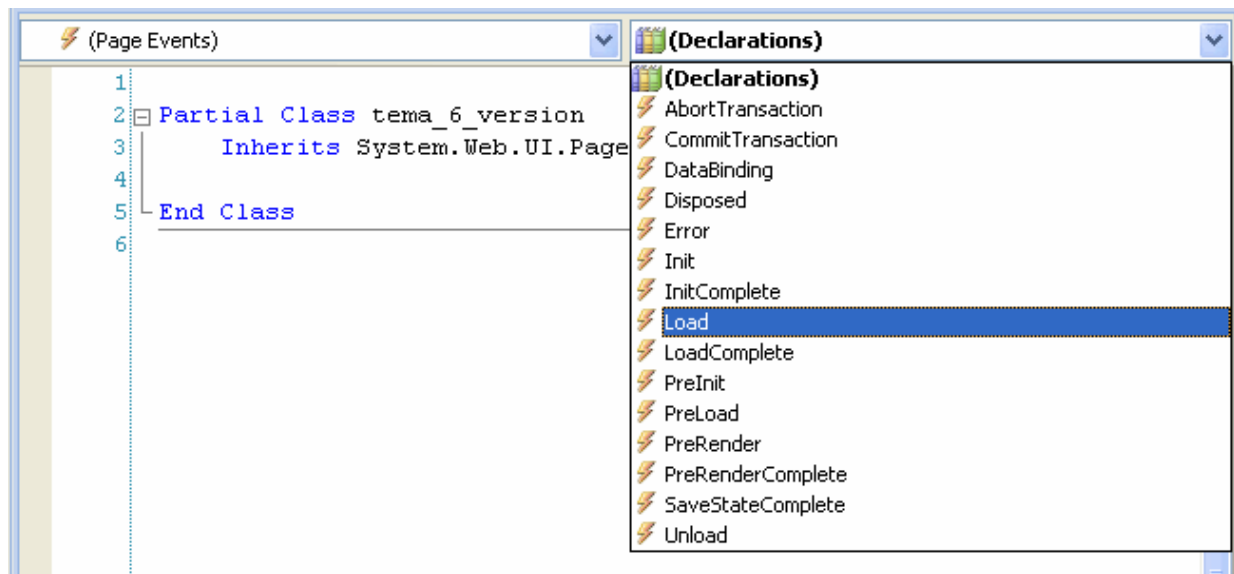


Vete al código de la página de version.aspx.vb y queremos poner un código para que se ejecute en el evento "Load" de la página, así que selecciona a la izquierda en la vista de código de "version.aspx.vb" "Page events":



Es un atajo para localizar todos los controles que tenemos en la página, si te fijas además está el obligatorio "form" y el label "Mensaje" que hemos añadido. AL seleccionar los eventos de la página podemos seleccionar el que queramos en el desplegable de la derecha, así que buscamos nuestro famoso "Load":

Formularios Web. La clase Page



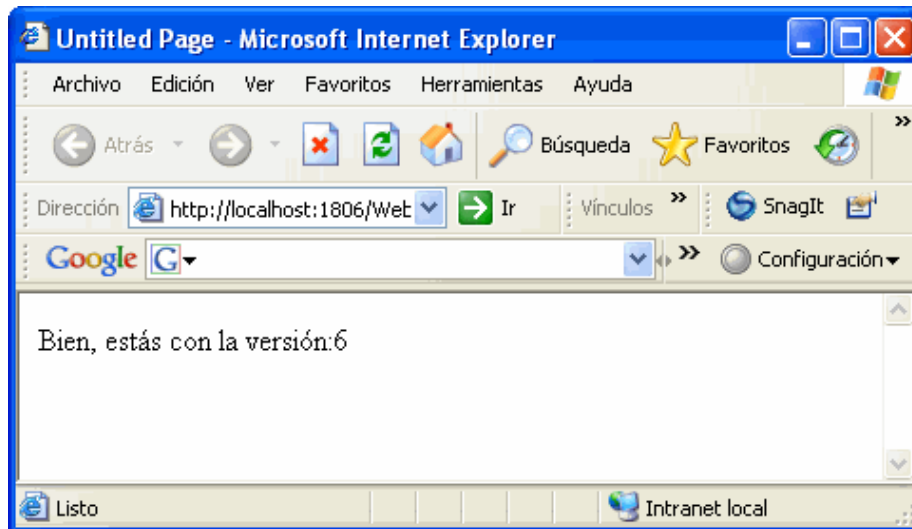
Lo seleccionamos y por fin podemos escribir dentro de este controlador de eventos, porque ya me ha escrito la declaración completa:

```
1
2 Partial Class tema_6_version
3     Inherits System.Web.UI.Page
4
5     Protected Sub Page_Load(ByVal sender As Object, ByVal e As
6         |
7     End Sub
8 End Class
9
```

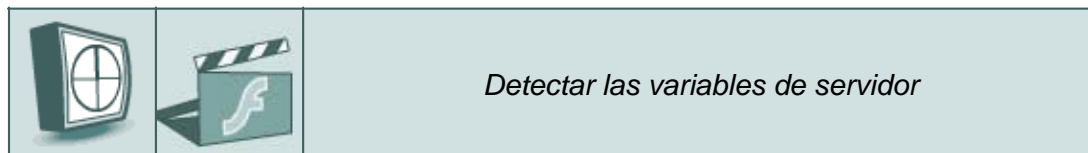
Ahora escribimos dentro de él:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If Request.Browser.Browser = "IE" Then
        If Request.Browser.MajorVersion < 6 Then
            Mensaje.Text = "Ya es hora de actualizar tu versión!"
        Else
            Mensaje.Text = "Bien, estás con la versión:" & Request.Browser.MajorVersion
        End If
    End If
End Sub
```

2. Ejecútalo en el navegador...



Lo que hemos hecho ha sido obtener mediante uno de los objetos del servidor unas propiedades que me dicen que navegador es y su versión. Así de fácil, ahora te explicas porque al visitar algunos sitios saben que lo estamos haciendo con tal navegador y versión, sólo tienen que poner una instrucción parecida a esta.



Luego hemos utilizado dos propiedades muy interesante "Request.Browser" contiene muchos detalles del navegador y entre ellos de que navegador se trata "browser" y de que versión "majorversion"

Ejemplo

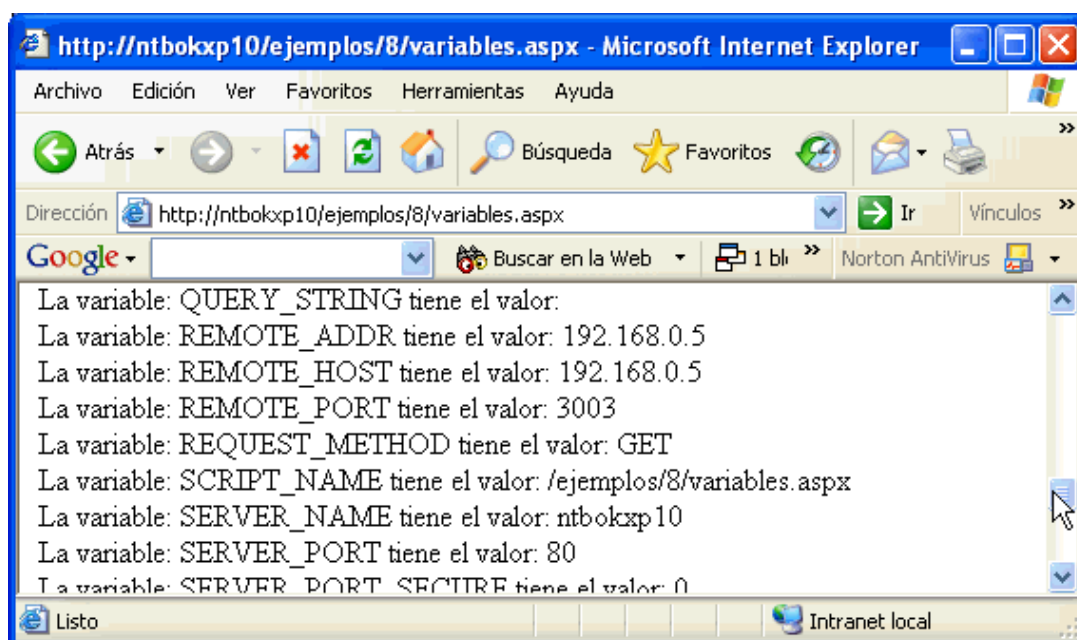
Veamos otro ejemplo, en esta caso queremos escribir todas las variables de servidor, así te harás una idea de todos los datos que podemos recuperar con estos objetos:

1. Crea una página que se llame "variables.aspx" e introduce este código en el evento "Load" de la página. Esta vez utilizaremos el método de "Response.Write" para escribir los datos en pantalla:

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

    For Each variable In Request.ServerVariables
        Response.Write("La variable: " & variable & " tiene de valor: " & Request.ServerVariables(variable) & "<br>")
    Next
End Sub
```

2. Ejecuta la página:



Verás una pantalla parecida a esta con todas las variables de servidor que podemos recuperar. Como ves podemos recuperar desde la dirección IP del cliente hasta su nombre de equipo y otros datos interesante. Muchas de estas variables pueden no tener valor porque dependen del contexto en que se estén ejecutando.

Todas estas variables pertenecen a una colección de variables de "Request.ServerVariables" por lo tanto la mejor forma de recorrer una colección como ya hemos mencionado antes es con un bucle "For... Each"

Accediendo a las variables de servidor del objeto Request hemos obtenido todos estos datos, pero antes no hemos utilizado esto sino que hemos utilizado otra forma de recuperar información específica del navegador. Es decir con este último ejemplo tenemos todas las variables de nuestra conexión y con el primer ejemplo accediendo a otro objeto hemos descubierto todos los valores de nuestro navegador. Ampliemos esta opción porque puede ser importante. Veamos todos los datos que nos ofrece el navegador:

Propiedad	Significado
ActiveXControls	Obtiene un valor que indica si el explorador del cliente admite controles ActiveX
AOL	Obtiene un valor que indica si el explorador del cliente es de la marca America Online (AOL). (curioso, estos americanos...)
BackgroundSounds	Obtiene un valor que indica si el explorador del cliente admite sonidos de fondo.
Beta	Obtiene un valor que indica si la versión del explorador es beta.
Browser	Obtiene la cadena del explorador que se transmitió en el encabezado User-Agent (en caso de que haya alguna).
CDF	Obtiene un valor que indica si el explorador del cliente admite Channel Definition Format (CDF) para las difusiones por Web.
ClrVersion	Obtiene el número de versión de Common Language Runtime .NET instalado en el cliente
Cookies	Obtiene un valor que indica si el explorador del cliente admite cookies.
Crawler	Obtiene un valor que indica si el explorador del cliente es un motor de búsqueda rastreador de Web.
EcmaScriptVersion	Obtiene el número de versión de la secuencia de comandos ECMA que admite el explorador del cliente.
Frames	Obtiene un valor que indica si el explorador del cliente admite marcos HTML.

Formularios Web. La clase Page

JavaApplets	Obtiene un valor que indica si el explorador del cliente admite subprogramas Java.
JavaScript	Obtiene un valor que indica si el explorador del cliente admite JavaScript.
MajorVersion	Obtiene el número de versión principal (es decir, el número a la izquierda del separador) del explorador del cliente.
MinorVersion	Obtiene el número de versión secundario (es decir, los números a la derecha del separador) del explorador del cliente.
MSDomVersion	Obtiene la versión de HTML (MSHTML) Document Object Model (DOM) de Microsoft que admite el explorador del cliente.
Platform	Obtiene el nombre de la plataforma que utiliza el cliente.
Tables	Obtiene un valor que indica si el explorador del cliente admite tablas HTML.
Type	Obtiene el nombre y el número de versión principal (es decir, el número a la izquierda del separador) del explorador del cliente
VBScript	Obtiene un valor que indica si el explorador del cliente admite VBScript.
Version	Obtiene el número de versión completo (los números a la izquierda y a la derecha del separador) del explorador del cliente.
W3CDomVersion	Obtiene la versión de XML Document Object Model (DOM) del Consorcio WWC que admite el explorador del cliente.
Win16	Obtiene un valor que indica si el cliente es un equipo basado en Win16
Win32	Obtiene un valor que indica si el cliente es un equipo basado en Win32

Veamos el resultado de estas variables mediante un ejemplo.

1. Crea una página web "variables2.aspx" con este código (puedes copiarlo...):

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

    Response.Write("ActiveXControls: " & Request.Browser.ActiveXControls & "<br />")

    Response.Write("AOL: " & Request.Browser.AOL & "<br />")

    Response.Write("BackgroundSounds: " & Request.Browser.BackgroundSounds & "<br />")

    Response.Write("Beta: " & Request.Browser.Beta & "<br />")

    Response.Write("Browser: " & Request.Browser.Browser & "<br />")

    Response.Write("CDF: " & Request.Browser.CDF & "<br />")

    Response.Write("Cookies: " & Request.Browser.Cookies & "<br />")

    Response.Write("Crawler: " & Request.Browser.Crawler & "<br />")

    Response.Write("Frames: " & Request.Browser.Frames & "<br />")

    Response.Write("JavaApplets: " & Request.Browser.JavaApplets & "<br />")

    Response.Write("JavaScript: " & Request.Browser.JavaScript & "<br />")

    Response.Write("MajorVersion: " & Request.Browser.MajorVersion & "<br />")

    Response.Write("MinorVersion: " & Request.Browser.MinorVersion & "<br />")

    Response.Write("Platform: " & Request.Browser.Platform & "<br />")
```

Formularios Web. La clase Page

```
Response.Write("Tables: " & Request.Browser.Tables & "<br />")

Response.Write("Type: " & Request.Browser.Type & "<br />")

Response.Write("VBScript: " & Request.Browser.VBScript & "<br />")

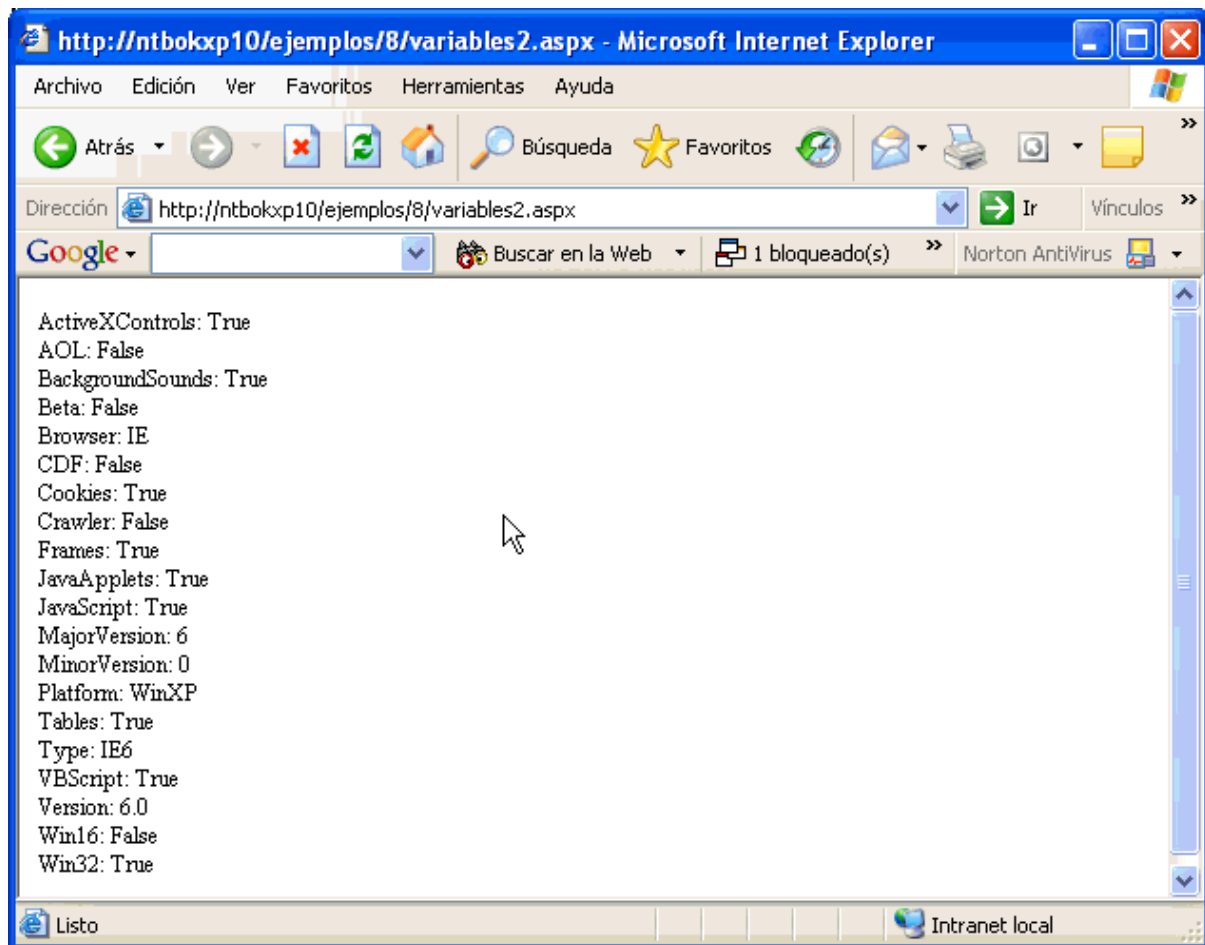
Response.Write("Version: " & Request.Browser.Version & "<br />")

Response.Write("Win16: " & Request.Browser.Win16 & "<br />")

Response.Write("Win32: " & Request.Browser.Win32 & "<br />")

End Sub
```

Ejecuta la página y obtendrás algo parecido a esto:



Donde te detalla muchas cosas interesantes: es un windows XP con TE versión 6, soporta java... Así que claro, cuando visitamos una página el servidor sabe todos nuestros detalles: sistema operativo, versión del explorador, idioma, y así nos puede redirigir a una página personalizada para esta configuración.

4.2 Objeto Response

Este objeto tiene la función de enviar datos al cliente, es decir, al navegador que ha cargado la página ASP.NET. A través de este objeto podremos escribir en la página que visualizará el usuario e incluso

podremos redirigir al usuario a otra dirección en Internet.

También a partir de este objeto podremos definir cookies para el usuario y asignarles un valor.

Colecciones del objeto Response

El objeto Response posee una única colección llamada Cookies, que le permite crear y asignar valores a una cookie. Una cookie, físicamente, es un fichero que se escribe en la máquina local del cliente que se conecta a un sitio Web y que contiene información relativa a la conexión.

Las cookies se utilizan para mantener información entre diferentes conexiones HTTP. Recuerda que el protocolo HTTP es un protocolo sin estado, es decir, no se retiene información entre las diferentes conexiones que se realicen. Por esta razón, ni el cliente ni el servidor pueden mantener información entre diferentes peticiones o a través de diferentes páginas Web.

En la referencia de los objetos de servidor al final de este curso puedes encontrar información de las propiedades, métodos y eventos de las cookies. Éstas son poco útiles en nuestro curso ya que vamos a utilizar la programación para realizar operaciones similares a las cookies y muchas mas y de mayor complejidad. Por lo tanto nos quedamos con que son un simple sistema para almacenar pequeños datos del cliente, como fecha de visita o su nombre.

Propiedades del objeto Response

Las propiedades y métodos mas importantes del objeto Response son:

- Buffer: indica si los datos de la página se almacenan en un búfer.
- ContentType: especifica el tipo de contenido HTTP de la respuesta. Los tipos de datos que se pueden enviar se especifican por los tipo "MTME". Por ejemplo una páginas web es de tipo "text/html", un gráfico GIF es de tipo "image/gif"...
- Clear: Se borra el búfer actual.
- Flush: Se envia al navegador todo el contenido HTML del búfer pero admite enviar mas código HTML.
- End: Envia todo el contenido al navegador y finaliza la página.
- Redirect: (muy utilizado) Permite redireccionar la página a otra. Para que funcione correctamente el búfer debe estar a "off"
- Write: Escribe HTML en la conexión, si está activado el bufering, no se enviará hasta que no se libere éste (Flush, End).
- WriteFile: Tgual que el anterior pero escribe el contenido de un fichero a la salida HTML.

Buffer

Si utilizamos la propiedad Buffer del objeto Response asignándole el valor True, conseguiremos que ASP procese todo el script en el servidor antes de enviar algo al usuario, por defecto esta propiedad tiene el valor True. El contenido del búfer no es enviado al navegador hasta que no se haya terminado de procesar la página o hasta que los métodos End o Flush del objeto Response no son invocados.

Formularios Web. La clase Page

Una vez activado el almacenamiento en búfer, es decir, la propiedad Buffer posee el valor True, y una vez que ya se ha enviado contenido al navegador del cliente, no se puede modificar su valor, tampoco se puede modificar ningún valor de las propiedades del objeto Response.

En las versiones anteriores de ASP la propiedad Buffer tenía por defecto la propiedad False, de esta forma según se iba procesando la página ASP se enviaba el resultado de la ejecución al cliente. Según Microsoft con este cambio se obtiene un mejor rendimiento en la ejecución de páginas ASP. En realidad lo que se tiene también es un mayor control sobre la salida que va a recibir el usuario.

El uso de este búfer puede ser útil en el caso de que sea necesario procesar el contenido de una página ASP antes de enviar ningún contenido previo al navegador del cliente. De esta forma se puede redirigir al navegador hacia otra página con el método Redirect del objeto Response, este método se comentará en su apartado correspondiente, pero se puede adelantar que una llamada a este método producirá un error en el caso de que se haya enviado anteriormente algún contenido al usuario.

La propiedad Buffer también puede ser útil para verificar errores antes de enviar información al navegador, de esta forma si se detecta un error se puede detener el proceso de la página y redirigir al navegador del cliente hacia una página determinada.

Cuando la propiedad Buffer, tiene su valor por defecto, es decir, el valor True se pueden utilizar una serie de métodos del objeto Response, estos métodos, cuya función se verá en el apartado correspondiente, son: Clear, End y Flush y básicamente lo que permiten es controlar el resultado o salida que se le envía al cliente.

Write

A lo largo de los distintos ejemplos que hemos visto hasta ahora en el curso, en alguno de ellos hemos utilizado la sentencia Response.Write, por lo tanto ya sabemos exactamente la función de este método del objeto Response.

Como ya se ha visto, el método Write se puede utilizar para escribir texto en el navegador. Las llamadas a métodos y las variables dentro del método Write serán evaluadas y sólo se escribirá el resultado en el navegador del cliente.

Por ejemplo, el siguiente ejemplo mostrará el valor de la variable contador en el navegador del cliente

```
Response.Write("La variable contador contiene el valor: " & contador)
```

Redirect

Este método recibe como parámetro una cadena de texto en forma de URL que especifica la nueva dirección a la que se va a redirigir el navegador del usuario. La URL puede ser absoluta, del tipo www.midominio.com o del tipo /directorio/pagina.aspx, o relativa del tipo pagina.aspx. Al lanzar el método Redirect en realidad lo que estamos haciendo es enviar una cabecera HTTP al navegador Web del cliente para indicarle que la página se ha movido y le indicamos la nueva localización, que es la URL que hemos indicado al método Redirect. La información que se envía al navegador es similar a la siguiente:

```
HTTP/1.1 Object Moved  
Location /directorio/pagina.aspx
```

Formularios Web. La clase Page

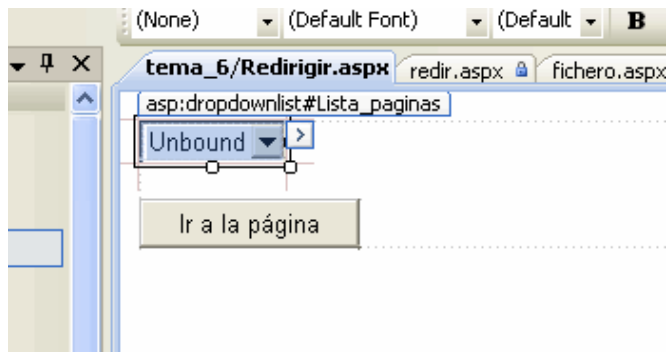
Como se puede observar la redirección se realiza en el cliente, indicándole que la página se ha movido a una nueva dirección, que en realidad es otra página completamente distinta. Aquí nos hemos centrado en redireccionar el navegador a otra página (HTML o ASP), pero se puede generalizar diciendo que podemos redireccionar al navegador a cualquier recurso, como puede ser una imagen, un fichero zip, un documento, etc.

Vamos ahora un interesante ejemplo

Ejemplo

El ejemplo será sobre el método redirect. Vamos a crear un cuadro desplegable con una serie de direcciones, pondremos un botón para navegar a esa dirección y controlaremos su redirección mediante el evento "click" de un botón. Con esto volvemos a agrupar varias de las cosas vistas hasta ahora.

1. Crea un página nueva que se llame "redirigir.aspx" e incluye un control de tipo "listbox" que se llame lista_paginas y un botón que se llame "btn_ir". Recuerda poner todo esto dentro del área de formulario, de lo contrario no funcionará al pulsar el botón:



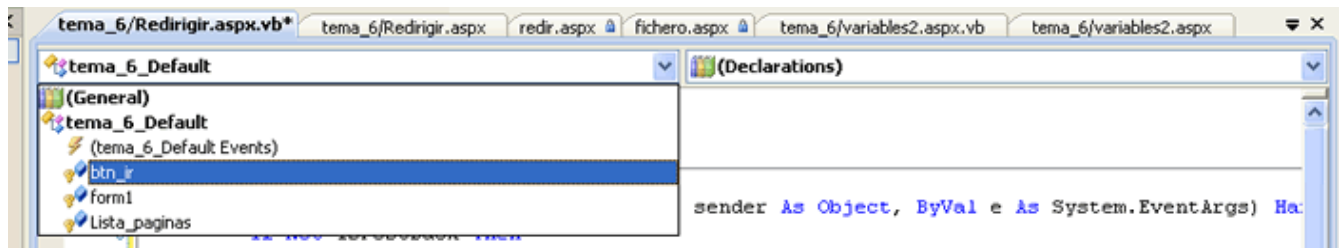
Ahora escribiremos lo siguiente en el evento Load de la página:

```
Protected Sub tema_6_Default_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Not IsPostBack Then
        btn_ir.Text = "Viajar"
        Lista_paginas.Items.Add("http://www.microsoft.com")
        Lista_paginas.Items.Add("http://www.elmundo.es")
        Lista_paginas.Items.Add("http://www.elpais.es")
        Lista_paginas.Items.Add("http://www.marca.es")
    End If
End Sub
```

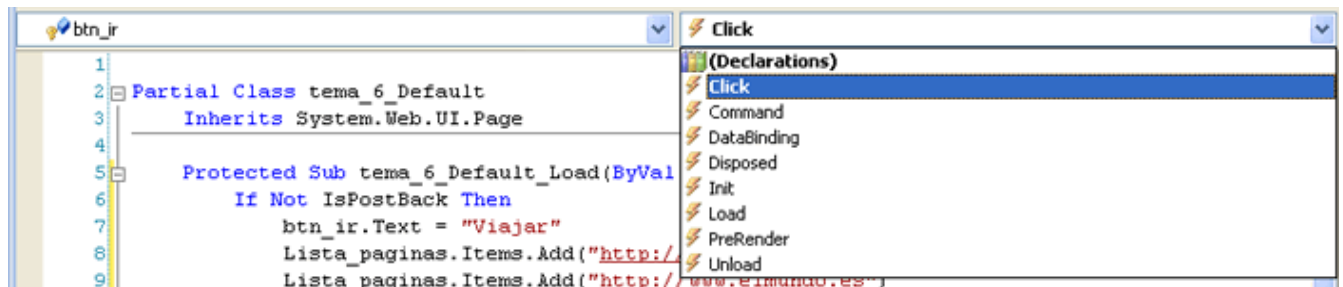
Que significa lo siguiente. La instrucción "IsPostBack" es nueva. Cuando pulsamos un botón para procesar datos se produce lo que se llama un "postback" que veremos en el siguiente tema. Es decir si la página por primera vez "ispostback" devuelve "false" porque no se ha cargado como producto de un "postback". Por tanto estas líneas se ejecutarán al cargarse la página por primera vez. Como lo que estamos haciendo es poner los elementos en la lista y el texto del botón sería un error que lo ejecutásemos cada vez que se acceda a la página ya que solo se debe ejecutar una vez para rellenar los datos.

Ahora nos queda procesar la selección del usuario. Cuando el usuario haga clic en el botón navegaremos a la página que ha seleccionado. Así que debemos trabajar con el evento clic del botón. Para esto nos vamos a la página de código y seleccionamos a la izquierda arriba el botón:

Formularios Web. La clase Page



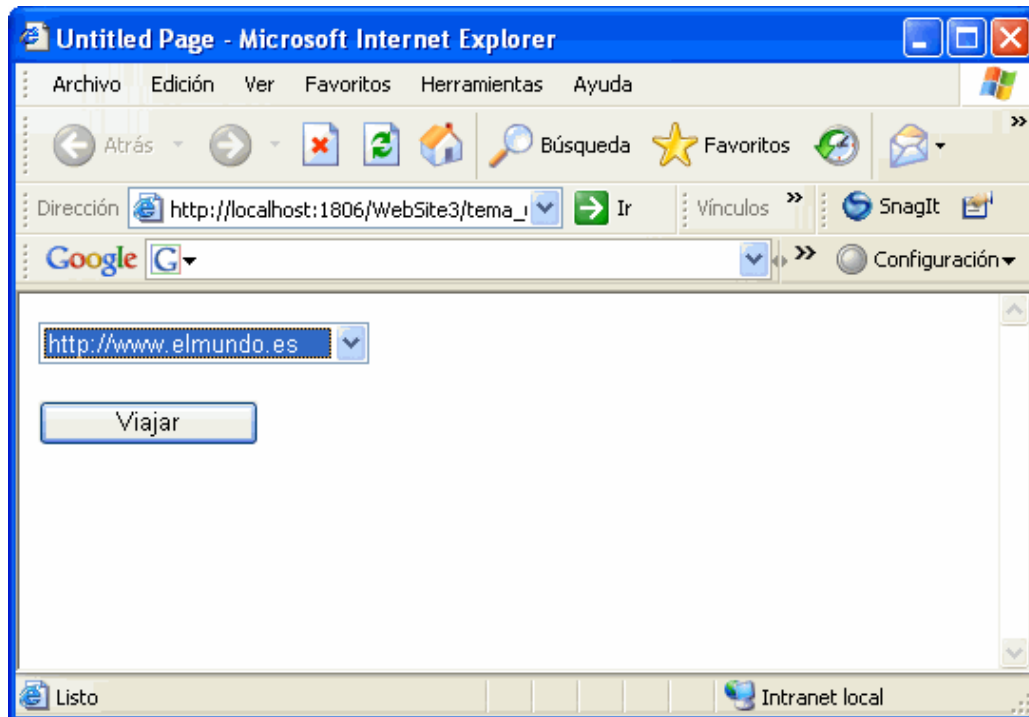
A la derecha nos aparecerán todos los eventos a los que puede atender este control, seleccionaremos el de clic:



Y escribiremos el código siguiente:

```
Protected Sub btn_ir_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Redirect(Lista_paginas.SelectedItem.Text)
End Sub
```

Que simplemente llama al método Redirect para navegar a la página que haya seleccionado el usuario en el cuadro de lista y que recogemos en el valor "lista_paginas.SelectedItem.Text". Ya veremos mas adelante estas instrucciones, no te preocupes. Ahora ejecutemos la página:

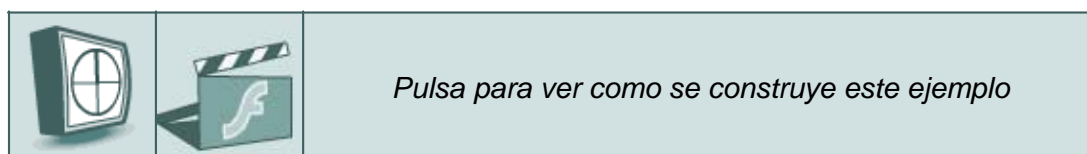


Formularios Web. La clase Page

Para empezar ha funcionado bien porque me ha rellenado los elementos de la lista desplegable y además en tiempo de ejecución le hemos puesto un texto al botón. Ahora veremos si funciona, seleccionamos una dirección y veremos si hace el “response.redirect”:



Pues si, ha funcionado, nos ha redirigido al texto contenido en “lista_paginas.SelectedItem.Text” que es la dirección web.



4.3 Objeto Server

El objeto Server nos permite ampliar las capacidades de las páginas ASP.NET mediante la posibilidad de la creación y utilización de objetos externos y componentes en el lenguaje de secuencias de comandos.

El objeto Server está diseñado para realizar tareas específicas en el servidor. Además de la posibilidad de instanciar (crear) componentes el objeto Server ofrece una serie de métodos muy útiles como pueden ser los que permiten dar formato URL o HTML a cadenas de caracteres, los que modifican la línea de ejecución de un script de una página con la posibilidad de ejecutar distintas páginas, y también existe un método utilizado para el tratamiento de errores dentro de ASP.NET, etc.

La clase página nos proporciona acceso a una propiedad llamada “Server” que devuelve un objeto de tipo “HttpServerUtility” con una serie de pequeñas ayudas para nuestras páginas:

- CreateObject: método por excelencia del objeto Server, crea una instancia de un componente. Este componente debe estar registrado en el servidor Web.
- MachineName. Propiedad que devuelve el nombre del equipo donde se está ejecutando ASP.NET
- HTMLEncode: devuelve una cadena codificada en HTML a partir de la cadena que se le pasa como parámetro. Si queremos escribir en el navegador el carácter “>” no se puede porque no existe, para el navegador ese carácter es el “>”. Con este método conseguimos que se traduzca correctamente cuando utilizemos caracteres no admitidos directamente.
- HTMLDecode: Es el contrario al anterior, por ejemplo toma el carácter “>” y devuelve “>”
- MapPath: devuelve la ruta física de una ruta virtual dada a través de una cadena en formato URL. Por ejemplo “/misdatos/pagina.aspx” devuelve “c:\web\misdatos\paginas.aspx”
- URLEncode: devuelve una cadena a la que se le ha aplicado la codificación URL

Formularios Web. La clase Page

correspondiente a las cadenas de consulta (QueryString).

- `URLEPathEncode`: devuelve una cadena a la que se le ha aplicado la codificación URL correspondiente a las cadenas de rutas.
- `ScriptTimeout`: expresa en segundos el periodo de tiempo durante el que puede ejecutarse una secuencia de comandos (script) antes de que termine su intervalo de espera.

Codificación de HTML

Sabemos que hay caracteres especiales en el lenguaje HTML, por ejemplo los ">" y "<" que pasa si queremos escribir estos caracteres como el contenido de una página web. Me explico, supón que tenemos una base de datos con los temas de un curso, por ejemplo este. El usuario solicita el tema 3 y le proporcionamos por código el contenido del texto, bien no hay problema ... o si? Pues si, imagina que le enseñamos una operación matemática donde haya "5<x", cuidado, hemos escrito un carácter "<" que es especial por tanto podrá identificarlo como una etiqueta HTML el navegador y no producir la salida adecuada. Otro ejemplo si escribimos:

Introduce un valor <aquí>

Si lo queremos poner dentro de la página o en un control, solo nos pondrá:

Introduce un valor

Ya que los caracteres "<" los interpreta como etiquetas HTML. Así que cuando mandemos este texto debemos traducirlo para que lo entienda bien, por ejemplo el carácter "<" se representa como "<" que si nos escribirla lo que queremos. y estos otros:

Resultado	Descripción	Codificado
	Espacio en blanco	
<	Simbolo de menor que	<
>	Simbolo de mayor que	>
&	"Ampersand"	&
"	Comillas	"

Podríamos explorar todo el texto que vamos a devolver y sustituir estos caracteres de la primera columna por los de la tercera. Pero es una dura labor, para esto tenemos un método del objeto `Server` que nos hará esta operación traduciendo el texto a HTML perfectamente válido, así que utilizáramos:

```
texto=server.HtmlEncode(texto_origen)
```

Dada la importancia de este objeto lo veremos mas adelante deteniéndonos en lo que nos afectará en nuestra programación.

4.4 Almacenar estados y las aplicaciones con ASP.NET

Una aplicación ASP.NET se puede definir de manera sencilla como un conjunto de páginas activas de servidor que tiene una función común y que comparten un lugar de almacenamiento de variables común.

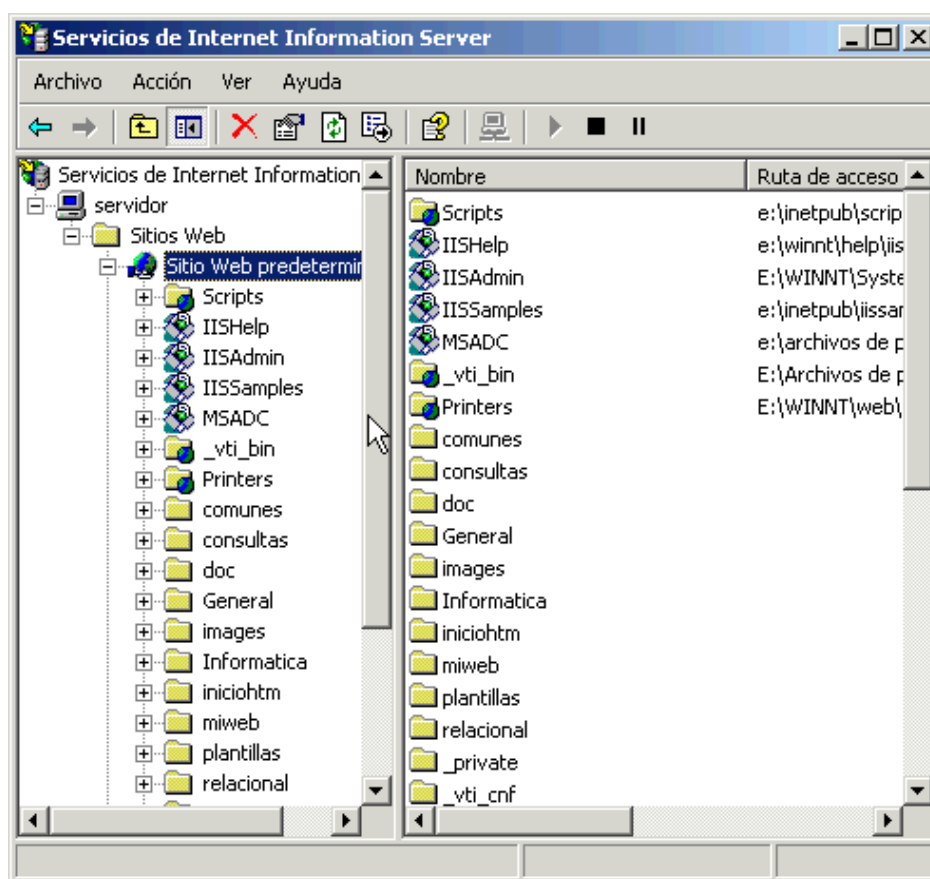
Formularios Web. La clase Page

Una aplicación ASP.NET se define a partir de un directorio o directorio virtual en un sitio Web. Debíamos indicar el punto de inicio de la aplicación ASP.NET, a partir de ese punto de inicio todas las páginas ASP formarán parte de la aplicación ASP y compartirán un mismo contexto. Un ejemplo de una completa aplicación ASP es una Intranet.

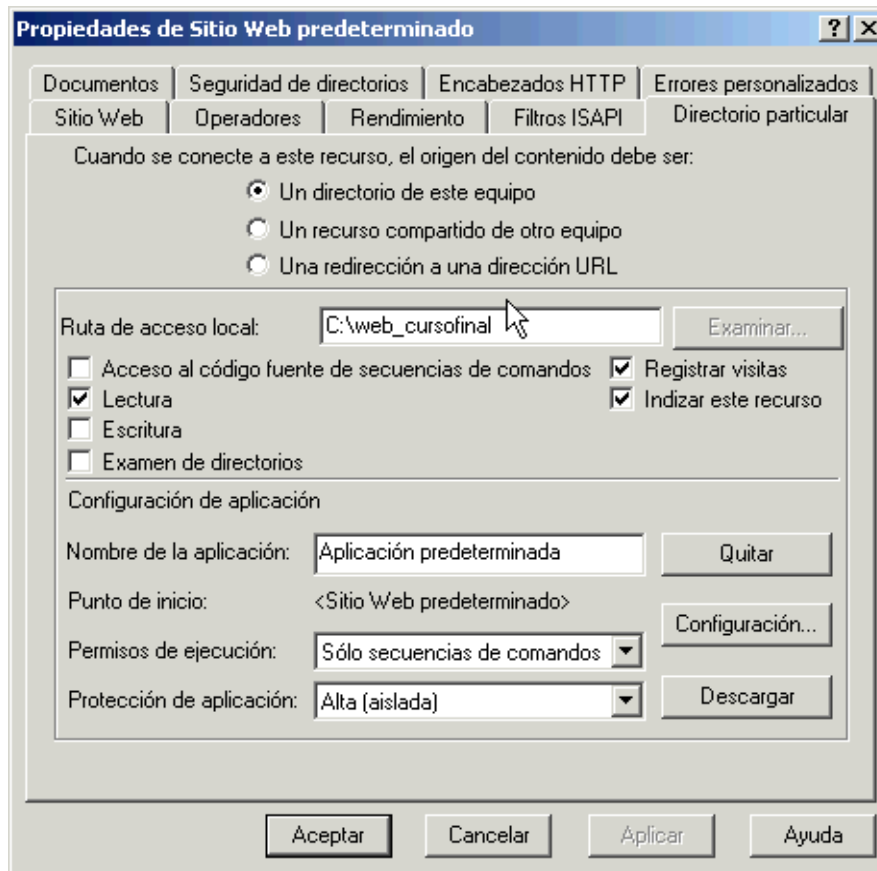
Una aplicación Web es un conjunto de páginas Web y otros recursos que se encuentran diseñados para realizar un tipo de tarea determinado para obtener unos resultados. Las aplicaciones Web no son aplicaciones al uso, ya que no disponen de un proceso de instalación real, sino que se encuentran contenidas en un servidor Web. En nuestro caso, para las aplicaciones ASP el servidor Web que vamos a utilizar es Internet Information Server.

El fichero global.asax es un fichero especial en el que se pueden definir una serie de eventos relativos a la aplicación ASP.NET y las sesiones que los usuarios tienen con la aplicación. Cada aplicación ASP.NET suele tener su fichero global.asax, más adelante volveremos a dar más información sobre este fichero.

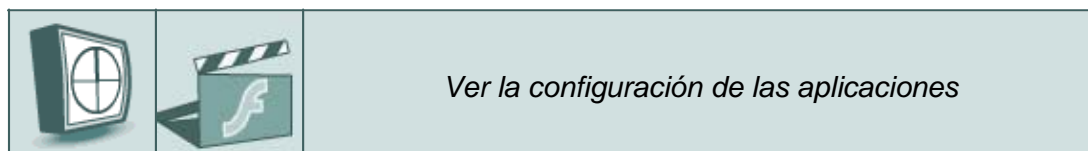
Veamos el aspecto que tiene una aplicación ASP.NET desde el Administrador de servicios de Internet.



Como se puede ver es algo similar a una bola metida en una caja, esto es lo que identifica gráficamente a una aplicación ASP.NET en el Administrador de servicios de Internet. Esta son las propiedades tanto si es del web raíz como de un nuevo web de aplicación:



Cada aplicación ASP.NET tendrá en conjunto de variables globales que podrán ser utilizadas por todos los usuarios de esa aplicación. No es recomendable anidar distintas aplicaciones ASP, cada aplicación ASP.NET debe tener su directorio bien diferenciado.



Una de la formas en las que el servidor web almacena los cientos de conexiones de los usuarios es implementando los estados de conexión. El almacenamiento de los estados de los clientes permite al navegador almacenar valores de la conexión que pueden ser muy útiles, por ejemplo variables que estén disponibles para varias páginas web, por ejemplo:

- Si nuestro web utiliza un usuario podemos almacenar esa información del usuario para que aparezca en todas las páginas del web, de esta forma se identifica en todas estas páginas.
- Otro ejemplo es una tienda virtual. Podemos crear un “carrito de la compra” que estará disponible en todas las páginas del sitio con todas sus variables
- ...

Este estado de la conexión como veremos es una de las partes fundamentales de nuestras páginas porque va a ser vital el disponer de elementos (variables, matrices, objetos) que estén disponibles en todas las páginas del web, o por lo menos en las que visite el usuario.

¿Qué alcance o duración tiene este estado?

Vale, tenemos dos objetos según queramos utilizarlos, uno es el "Application". Todo lo que hagamos a nivel de "Application" estará visible para todos los usuarios que se conecten, muy útil para definir variables o procedimientos globales. La otra forma es el objeto "Session" que almacena los datos sólo para el usuario. Es decir, me conecto y el servidor web crea el objeto Session con una serie de valores que he puesto, cuando me desconecto se destruye esa sesión, de esta forma consigo tener una área sólo para cada usuario pudiendo tener sus propios valores.

Por ejemplo podemos definir cómo es una cesta de la compra a nivel de aplicación (Application) pero luego cada usuario tendrá su propia cesta con compra (Session).

4.5 Estado "Application"

Un objeto Application representa una aplicación ASP. Una aplicación basada en ASP.NET, como ya hemos visto en el apartado anterior, consta de un directorio en un servidor Web y de todos los subdirectorios y archivos contenidos en él. Una aplicación puede ser una página principal sencilla, o bien puede estar formada por un conjunto completo de páginas interrelacionadas entre sí.

Se debe recordar que el protocolo HTTP es un protocolo sin estado, es decir, no se puede almacenar información entre diferentes conexiones HTTP. No se puede mantener el estado entre diferentes páginas Web a través del protocolo HTTP, sino que se deben utilizar otros mecanismos como las cookies. Pero el objeto Application junto con el objeto Session (que lo veremos luego) nos permite de forma sencilla y directa almacenar información sin tener que pensar en el uso de cookies y de encabezados HTTP.

La información almacenada en los objetos Session y Application difieren en el ámbito de la misma, una información tendrá el ámbito de la sesión de un usuario concreto y la otra el ámbito de la aplicación general, respectivamente.

Las variables almacenadas dentro del objeto Application son visibles para todos los usuarios que están utilizando la misma aplicación ASP.NET, es decir son compartidas por varios usuarios. Por otro lado el objeto Session, cuyas variables son para cada uno de los usuarios conectados, es decir, no se comparten y son propias de cada sesión. Podremos acceder a una variable a nivel de aplicación en cualquiera de las páginas ASP.NET contenidas en la aplicación ASP.NET actual.

En este punto se debe señalar que una variable en una aplicación ASP.NET puede tener cuatro ámbitos diferentes, se va a ir del más global al más particular. La creación de una variable con un ámbito u otro dependerá del uso que queramos hacer de ella.

- **Ámbito de aplicación:** esta variable la podrán manipular todos los usuarios de la aplicación ASP.NET, es común a todos ellos. **Se almacena dentro del objeto Application.**
- **Ámbito de sesión:** las variables con este ámbito son propias de cada uno de los usuarios de la aplicación, es decir, cada usuario tendrá acceso a las variables de su sesión. Estas variables se almacenan dentro del objeto integrado Session.
- **Ámbito de página:** estas variables son las que se crean dentro del script de servidor de una página ASP.NET, sólo tienen vigencia dentro de la página en la que se declararon, al abandonar la página se destruirán.
- **Ámbito de procedimiento:** a este ámbito pertenecen las variables declaradas dentro de un procedimiento (Sub o Function). Estas variables sólo existirán dentro del procedimiento en el que son declaradas, se dice que son variables locales al procedimiento.

Formularios Web. La clase Page

Para almacenar una variable dentro de un objeto Application, es decir, crear una variable cuyo ámbito sea la aplicación se debe utilizar la siguiente sintaxis:

Algunas veces queremos almacenar información que esté accesible desde cualquier página de mi sitio web. El sitio para almacenar estas variables u objetos es el “estado Application” que utiliza un objeto creado a partir de la clase “HttpApplicationState” y su uso es muy similar a lo visto antes de Response o Request.

Las variables almacenadas en el objeto Application son visibles para todos los usuarios de la aplicación ASP.NET. Debido a esto se pueden dar problemas de concurrencia, es decir, cuando dos usuarios acceden a la misma información y la quieren modificar al mismo tiempo, por lo tanto se debe tener un acceso exclusivo a las variables del objeto Application cuando se quiera modificar su valor.

Aquí entendemos por usuarios a sentencias del lenguaje de secuencias de comandos, es decir, serían dos sentencias en distintas páginas que intentaran modificar la misma variable de aplicación al mismo tiempo.

Para implementar este mecanismo de exclusión mutua el objeto Application ofrece dos métodos: Lock y Unlock. Cuando se llama al método Lock, éste impide que otros usuarios modifiquen el contenido de las variables de la aplicación. Y cuando se llama al método Unlock, se permite el acceso a las variables a todos los usuarios de la aplicación.

Es un método de exclusión mutua, sólo un usuario puede a la vez estar utilizando las variables del objeto Application, las variables quedarán libres para el resto de los usuarios cuando se lance sobre el objeto Application el método Unlock. Ejemplo de utilización de estos dos métodos:

```
<%Application.Lock  
  
Application("NumVisitas")=Application("NumVisitas")+1  
  
Application.Unlock%>
```

Lock y Unlock se utilizarán normalmente en el momento en el que se vaya a modificar el valor de una variable de aplicación.

4.6 El fichero GLOBAL.ASAX

El fichero GLOBAL.ASAX (ASAX, Active Server Application, aplicación activa de servidor o mejor, aplicación ASPX), que hemos mencionado antes, es un fichero opcional que se encuentra en el directorio raíz de la aplicación ASPX y que está relacionado de forma directa con los objetos integrados Application y Session. Este fichero, que debe ser único para cada aplicación ASPX, tiene varias funciones:

- Definir como son tratados los eventos de los objetos Session y Application. Como hemos comentado en el apartado anterior.
- Permitir crear objetos con el ámbito de aplicación y sesión.
- Inicializar y crear variables en el ámbito de la sesión y aplicación.

Este fichero no es un fichero cuyo contenido se muestre al usuario, si un usuario intenta cargar el fichero GLOBAL.ASAX en su navegador recibirá el siguiente mensaje de error:

```
HTTP Error 500-15 - Requests for global.asax not allowed  
  
Servicios de Internet Information Server
```

Formularios Web. La clase Page

El fichero GLOBAL.ASAX puede contener:

- Información acerca del tratamiento de los eventos de los objetos Session y Application.
- Declaraciones de objetos mediante la etiqueta <OBJECT>.

La estructura general de GLOBAL.ASAX es la siguiente:

```
<SCRIPT LANGUAGE="VB" RUNAT="Server">

    SUB Application_OnStart

        .....

    END SUB

    SUB Session_OnStart

        .....

    END SUB

    SUB Session_OnEnd

        .....

    END SUB

    SUB Application_OnEnd

        .....

    END SUB

</SCRIPT>
```

Si se escribe script fuera de las etiquetas <SCRIPT></SCRIPT> o se declara un objeto que no tiene ámbito de sesión o de aplicación se producirá un error. Tampoco se pueden incluir ficheros de ningún tipo mediante el uso de la directiva <INCLUDE>.

El lenguaje de secuencias de comandos a utilizar se indica en el atributo LANGUAGE de la etiqueta <SCRIPT>, y también se debe indicar en que lugar se ejecutará el script, esto se consigue a través del atributo RUNAT de la etiqueta <SCRIPT>, en realidad este atributo sólo tiene un valor posible: Server, es decir, el script se ejecutará en el servidor.

Para declarar objetos mediante la etiqueta <OBJECT>, para que tengan el ámbito de sesión o aplicación se deberá seguir la siguiente sintaxis:

```
<OBJECT RUNAT=Server SCOPE=Ambito ID=Identificador

    {PROGID="IDprog" | CLASSID="IDclase"}>

    .....

</OBJECT>
```

Formularios Web. La clase Page

La declaración de los objetos deberá ir fuera de las etiquetas de script. Los objetos declarados de esta forma no se crearán hasta que el servidor procese una secuencia de comandos en el que se haga referencia a ellos. Así se ahorran recursos, al crearse sólo los objetos que son necesarios.

Estos objetos pasaran a formar parte de la colección StaticObjects del objeto Application.

El parámetro “Ambito” especifica el ámbito del objeto, podrá tener los valores Session o Application; Tidentificador especifica un nombre para la instancia del objeto; TDprog e TDclase para identificar la clase del objeto.

La propiedad SCOPE de la etiqueta <OBJECT> de HTML, tiene un valor de ámbito más, además de los de sesión y de aplicación, y se trata del ámbito de página SCOPE=“PAGE“. El ámbito de página indica que el objeto que se instancie sólo existirá en el página actual, pero este ámbito no se puede utilizar en el fichero global.asax sino que se puede utilizar en cualquier página ASP.NET. Sin embargo en las páginas ASP.NET no se puede utilizar la etiqueta <OBJECT> con el ámbito de aplicación o de sesión, únicamente se pueden instanciar objetos con ámbito de página.

El siguiente código dentro del fichero GLOBAL.ASAX crearla un objeto Connection de ADO (acceso a datos) con ámbito de sesión.

```
<OBJECT RUNAT=Server SCOPE=Session ID=conexion PROGID="ADODB.Connection">

</OBJECT>
```

Para utilizar este objeto posteriormente dentro de una página ASP.NET, por ejemplo, para ejecutar una sentencia SQL sobre la conexión con la base de datos, bastarla con escribir:

```
<% conexion.Execute "DELETE FROM Usuarios WHERE id<10" %>
```

Es decir, accedemos directamente al nombre del objeto, que será el indicado en la propiedad TD de la etiqueta <OBJECT>. Veamos el código de un fichero GLOBAL.ASA concreto.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">

Sub Application_OnStart

    Application("variableUno")="valor"

    Application("variableDos")=2

    Application("variableTres")=True

End Sub

Sub Application_OnEnd

End Sub

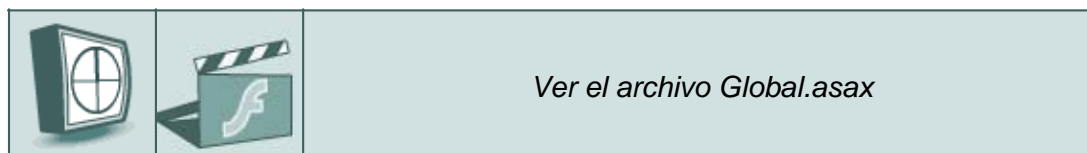
</SCRIPT>

<OBJECT RUNAT="Server" SCOPE="Application" ID="conex" PROGID="ADODB.Connection">

</OBJECT>

<OBJECT RUNAT="Server" SCOPE="Application" ID="recordset" PROGID="ADODB.RecordSet">
```

</OBJECT>



Cuando se inicie la aplicación ASP correspondiente se crearán tres variables de aplicación que pasarán a formar parte de la colección Contents, también se crean dos objetos a nivel de aplicación que pasarán a formar parte de la colección StaticObjects.

El fichero global.asax sólo se ejecutará si se ha indicado el punto de inicio de la aplicación correspondiente. Esta es la secuencia de ejecución del fichero GLOBAL.ASAX:

El objeto Application se crea cuando el primer usuario se conecta a una aplicación ASP.NET y pide una sesión, es decir, carga la primera página de la aplicación ASP.NET. Cuando se crea el objeto Application, el servidor busca el fichero GLOBAL.ASAX en el directorio raíz de esa aplicación ASP.NET, si el fichero existe se ejecuta el script del evento Application_OnStart.

A continuación se crea el objeto Session. La creación del objeto Session ejecuta el script que se encuentra en el evento Session_OnStart. El script asociado al tratamiento de este evento se ejecutará antes de cargar la página indicada por la petición del navegador cliente. Cuando el objeto Session caduca o se lanza el método Session.Abandon, se ejecutará el script que se corresponde con el evento Session_OnEnd, el código de este evento se procesará antes de destruir la sesión.

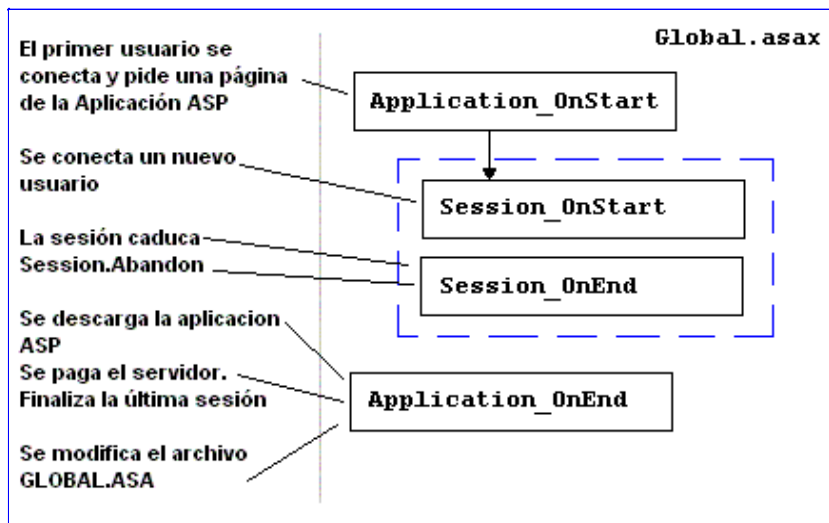
Cuando finaliza la última sesión de los usuarios, es decir, se ejecuta el código del último evento Session_OnEnd, se lanza el evento Application_OnEnd antes de que se destruya el objeto Application.

Si se modifica el fichero GLOBAL.ASAX, el servidor lo recompilará, para ello el servidor deberá destruir el objeto Application actual y los objetos Session actuales. Primero, el servidor procesa todas las peticiones activas, el servidor no procesará más peticiones hasta que no se procese el evento Application_OnEnd. Los usuarios que se intenten conectar durante este proceso recibirán un mensaje que le indica que la petición no puede ser procesada mientras la aplicación es reiniciada. Durante este proceso se dan los siguientes pasos:

- Las sesiones activas se destruyen, dando lugar al procesamiento del evento Session_OnEnd.
- La aplicación se destruye, produciéndose el evento Application_OnEnd.
- La primera petición reiniciará el objeto Application y creará un nuevo objeto Session, es decir, se darán los eventos Application_OnStart y Session_OnStart, respectivamente.

También se ejecutará la el evento Application_OnEnd cuando se descargue la aplicación ASP desde el Administrador de servicios de Internet o cuando se apague el servidor Web.

En la siguiente figura se puede apreciar a modo de resumen el orden en el que los eventos de los objetos Session y Application se producen, y en que condiciones.



El fichero Global.ASAX se mantiene por compatibilidad de las antiguas páginas ASP. Es decir, los antiguos ficheros ASA pueden pasarse a ASAX y así integrarse en nuevas páginas web- A partir de la versión 1.1 y sobre en nuestra 2.0 los ficheros de configuraciones que utilizaremos serán varios, pero el mas importante el que veremos adelante “web.config”

4.6 Estado "Session"

Como hemos comentado antes este objeto es bastante mas práctico que el anterior Application. En este caso las variables y demás datos que se almacenan en un objeto de tipo “session” están visibles sólo para el usuario que ha iniciado una sesión.

Al igual que ocurría con el objeto Application, el objeto Session nos va a permitir almacenar información entre diferentes páginas ASP.NET incluidas en una misma aplicación ASP.NET. La diferencia con el objeto Application se encuentra en el ámbito de las variables, cada variable del objeto Session es particular a una sesión de un usuario determinado, no a toda la aplicación. De esta forma, cada usuario tendrá sus variables y sus valores, sin dar lugar a problemas de concurrencia, tampoco se podrá acceder a distintas variables de sesión, cada usuario tiene su espacio de almacenamiento.

Las variables de aplicación son valores globales y comunes a toda la aplicación, y las variables de sesión son particulares para cada usuario de la aplicación.

El servidor Web crea automáticamente un objeto Session, cuando un usuario que aún no estableció una sesión solicita una página ASP.NET perteneciente a la aplicación ASP.NET actual.

Un uso común del objeto Session es almacenar las preferencias del usuario o información personal. Para una aplicación **ASP.NET tendremos tantas sesiones como usuarios conectados a la aplicación ASP.NET.**

La sintaxis para manipular variables del objeto Session es la misma que en el objeto Application. El siguiente ejemplo crea una variable del objeto Session con el nombre que el usuario facilita en un formulario.

```
Session("nombreUsuario")="JoseRM"
```

En cuanto a las propiedades y métodos mas importantes tenemos:

- ♦ **SessionTD:** contiene la identificación de la sesión para el usuario. Cada sesión tiene un identificador único que genera el servidor al crearla. No se debe utilizar esta propiedad como clave de una tabla de una base de datos, ya que, al reiniciar el servidor Web, algunos de los valores de SessionTD pueden coincidir con los generados antes de que se apagase el servidor. Es únicamente de lectura.
- ♦ **Timeout:** la propiedad Timeout especifica el intervalo de inactividad para el objeto Session en minutos. Si el usuario no actualiza o solicita una página durante ese intervalo, la sesión termina. El valor por defecto de esta propiedad es de 20 minutos, es decir, por defecto la sesión permanecerá inactiva 20 minutos. Una sesión se dice que está inactiva mientras el navegador cliente no realice una petición. El valor de esta propiedad se puede modificar dinámicamente a lo largo de la ejecución de la aplicación ASP.NET
- ♦ **TsNewSession.** Devuelve “true” si la sesión se ha creado cuando el usuario accede a la página actual.
- ♦ **Clear.** Borra toda la información del estado de la sesión pero no la termina.
- ♦ **Abandon.** Método que termina con la sesión actual, se borra toda la información de la sesión.

Un sencillo ejemplo del uso de una variable de este tipo mezclado con el anterior fichero global.asax sería:

Para el fichero global.asax:

```
<script language="vb" runat="server">  
  
Sub session_Onstart ()  
  
    Session ("mensaje")="Bienvenido a mi WEB"  
  
End Sub  
  
</script>
```

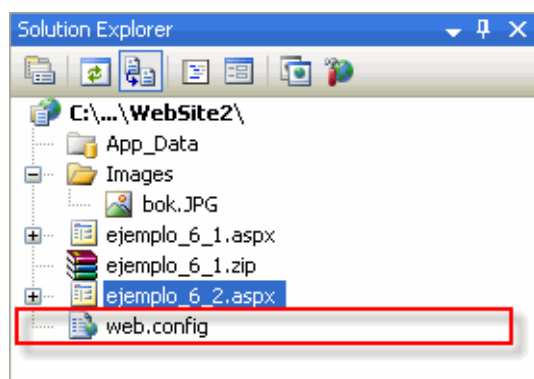
Y luego podríamos recuperar este valor con:

```
<% Response.write (Session ("mensaje")) %>
```

Con estos objetos de Session y Application nos sucede lo mismo que con el de Server, son muy importantes así que veremos detalles de ellos mas adelante en unos capítulos.

5. Configuración de ASP.NET

Hemos comentado en varios ejemplos que disponemos en nuestro web de un fichero de configuración: “web.config”:



Este es distinto y único para cada web de nuestro servidor TTS. Como detalles mas importantes tenemos que:

- No está cerrado. Es decir, en cualquier momento lo podemos modificar para añadir distintas propiedades para personalizar nuestro sitio web.
- Es fácil de acceder, está a nuestra disposición con sencillas instrucciones. Incluso remotamente desde otro equipo.
- Los valores son fáciles de entender y editar.

Nota: No confundas este fichero con el opcional que vimos antes de “global.asax”. Este fichero de web.config nos va a configurar distintos parámetros de nuestra aplicación web y siempre existe. El global.asax es opcional y sirve para poner unos procedimientos y declaraciones de alcance de usuario (de tipo session) y de alcance para todos los usuarios (de tipo application)

Este fichero tiene el formato estándar XML, tan famoso y que ya veremos mas adelante en detalle. De momento lo veremos un poco:

```

1  <?xml version="1.0"?>
2  <!--
3      Note: As an alternative to hand editing this file you can use the
4      web admin tool to configure settings for your application. Use
5      the Website->ASP.NET Configuration option in Visual Studio.
6      A full list of settings and comments can be found in
7      machine.config.comments usually located in
8      \Windows\Microsoft.Net\Framework\v2.0.50727\Config
9  -->
10 <configuration>
11     <configSections>...
24     <appSettings/>
25     <connectionStrings/>
26     <system.web>...
101    <system.codedom>...
114    ...
118    <system.webServer>...
134    <runtime>...
146 </configuration>
    
```

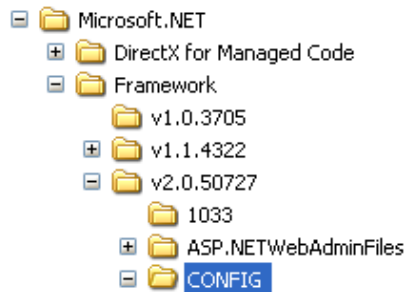
Podemos observar que tiene un aspecto jerárquico ya que hay etiquetas “<configSections>” que podemos ampliar y que contienen otras hasta que termina con </configSections>. En los ficheros XML se distinguen entre mayúsculas y minúsculas, así que debemos tener cuidado ya que “appSettings” no podemos llamarlo como “AppSettings” porque no es lo mismo.

Hay muchas configuraciones importantes que pondremos aquí cuando vayamos creando nuestro web. Recuerda que una propiedad nos habilitaba la depuración “debug”. Por ejemplo definiremos una sección “<connectionStrings>” donde definiremos nuestras conexiones a las bases de datos.

5.1 Configuraciones anidadas

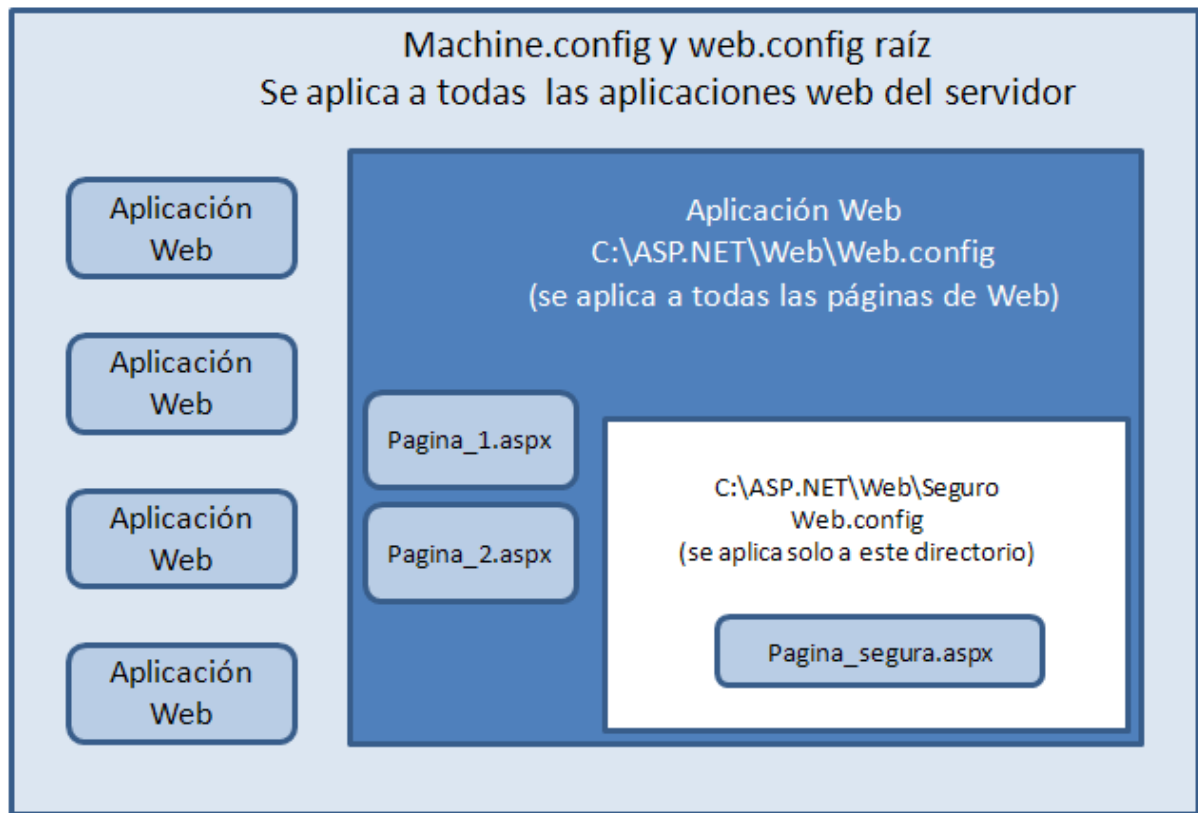
ASP.NET utiliza configuraciones en capas para permitirnos establecer distintos valores en distintos niveles, veamos que quiere decir esto...

Cada servidor web comienza con los valores básicos que se definen en dos ficheros de configuración en “c:\windows\Microsoft.NET\Framwork\v2.0.50727\Config:



Estos dos ficheros son “machine.config” y “web.config”. Normalmente no necesitaremos modificar estos ficheros ya que afectarían al funcionamiento del equipo completo. Esto explica lo que te he comentado antes de ser de “varios niveles”. Por un lado tenemos unos ficheros de configuración globales y por otro tenemos ficheros de configuración para cada web, que son los que modificaremos habitualmente.

Mas interesante es utilizar diferentes configuraciones en distintas partes de nuestra aplicación web. Para esto crearemos distintos directorios en nuestro directorio virtual principal. Recuerda que cada web es un directorio virtual de nuestro servidor Web. Cada uno de estos directorios pueden contener sus propios ficheros “config.web” y así tener valores de configuración distintos. No será habitual pero ya sabes que puede hacerlo. Los valores que pongamos aquí no anularán los del web principal sino que los amplían, es decir podemos definir un pequeño conjunto de propiedades que se utilizarán en esa parte del web y que se añadirán a las del “config.web” raíz. Veamos como será jerárquicamente:



5.2 Almacenar información en el fichero web.config

ASP.NET nos permite almacenar valores de nuestra aplicación web dentro de la sección llamada “<appSettings>” que está anidado dentro de la raíz <configuration>, su estructura sería:

```
<?xml version="1.0" ?>

<configuration>

    ...

    <appSettings>

        <!-- pondremos aquí nuestros valores -->

    </appSettings>

    ...

    <system.web>

        <!-- ASP.NET configuración del web -->

    </system>

    ...

</configuration>
```

Formularios Web. La clase Page

Como ves podemos poner comentarios dentro de los ficheros XML con los caracteres `<!-- comentarios... -->`. Los valores que escribiremos en este fichero son cadenas de caracteres estándar. Modificaremos estos valores frecuentemente para:

- Centralizar valores importantes de configuración que se utilizarán en varias páginas, por ejemplo variables que almacenen una consulta a una base de datos. Cualquier página que la necesite simplemente accederá a esta variable de configuración.
- Pasar fácilmente de un estado a otro de ejecución. Por ejemplo para la depuración, si leemos una variable que nos indique el nivel de depuración escribiremos información adicional en pantalla para comprobarlos valores de las páginas.
- Establecer valores iniciales.

Para añadir elementos a este fichero simplemente escribiremos dentro de él lo siguiente:

```
<appSettings>
```

```
<add key="ruta_base_de_datos" value="e:\datos" />
```

</appSettings>

Así que simplemente añadimos una variable “key” con un valor “value”. Para utilizar este fichero de configuración disponemos de una clase llamada “WebConfigurationManager” que nos va a proporcionar todos los métodos necesarios para realizar fácilmente el mantenimiento de este fichero.

Por ejemplo, para leer un valor llamamos al método `“WebconfigurationManager.AppSettings[“ruta base de datos“]”`. Veamos un ejemplo.

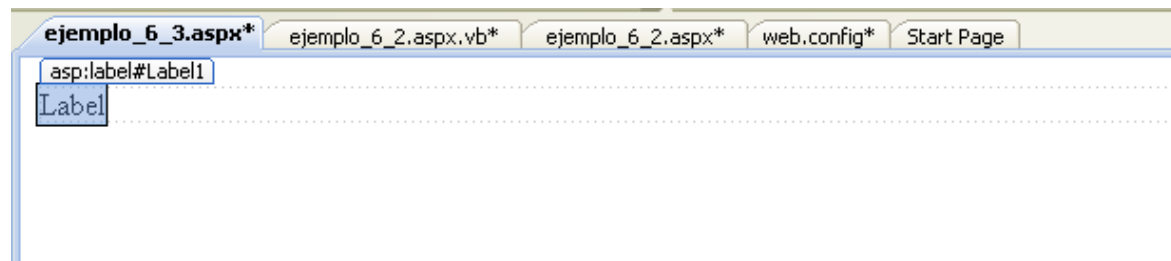
Metemos en el fichero config.web de nuestro web de ejemplos el valor:

```

        <section name="profileService" type="System
        <section name="authenticationService" type=
        <section name="roleService" type="System.We
    </sectionGroup>
</sectionGroup>
</sectionGroup>
</configSections>
<appSettings>
    <add key="ruta_base_de_datos" value="e:\datos" />
</appSettings>

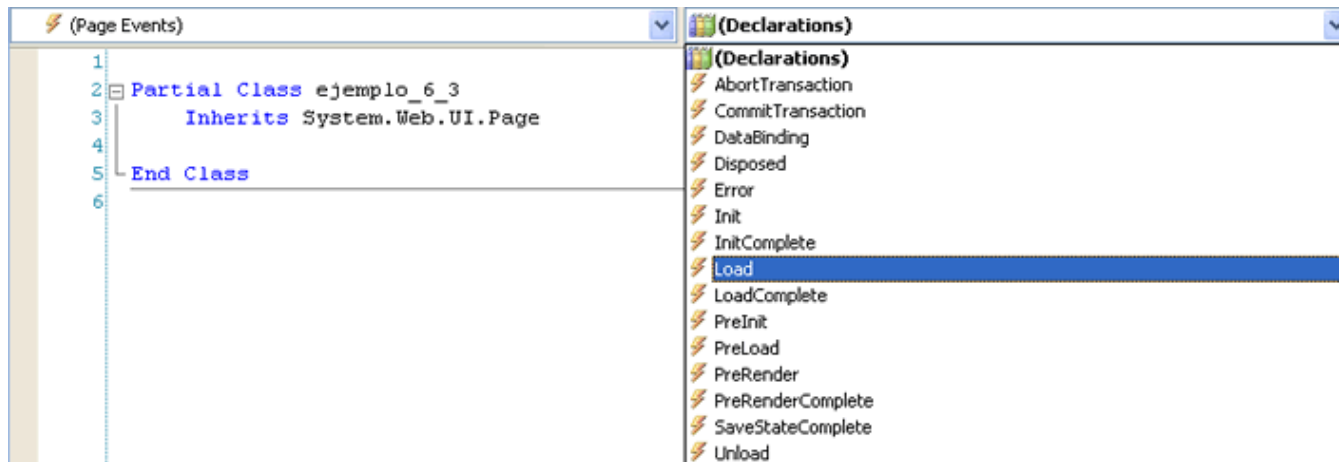
```

Ahora creamos una sencilla página web con un control de tipo “label“ donde escribiremos un valor cuando se cargue la página (Page Load):



Nos vamos a la página de código, seleccionamos arriba a la izquierda el objeto de la página y a la derecha el evento “Load”:

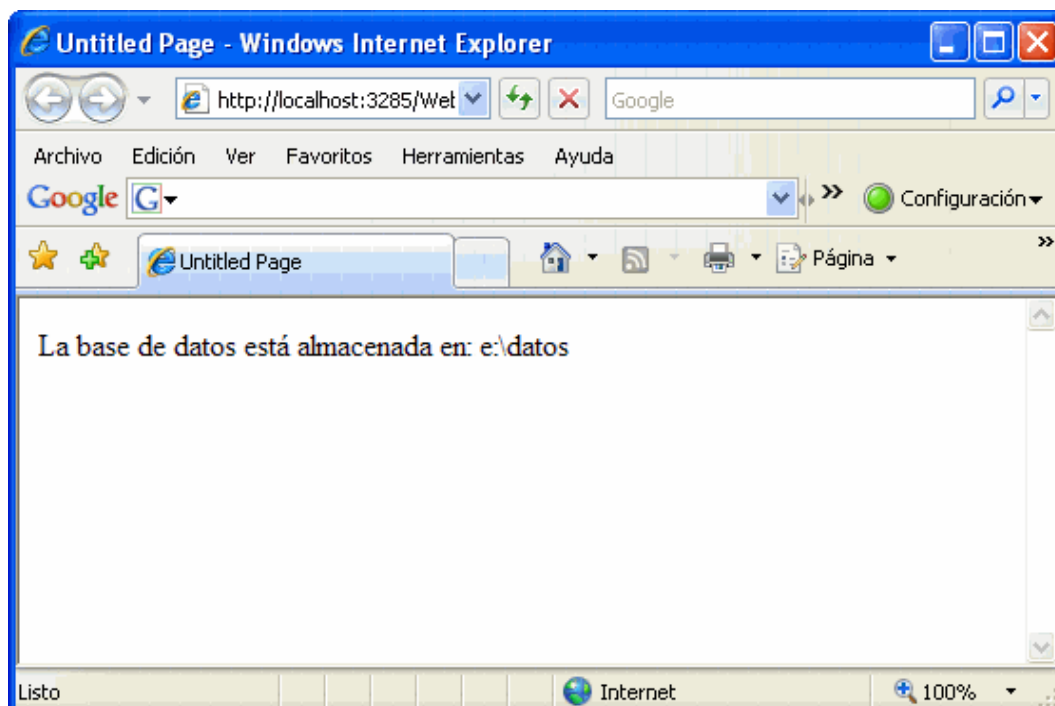
Formularios Web. La clase Page



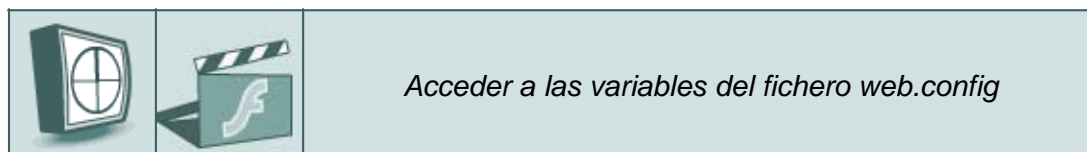
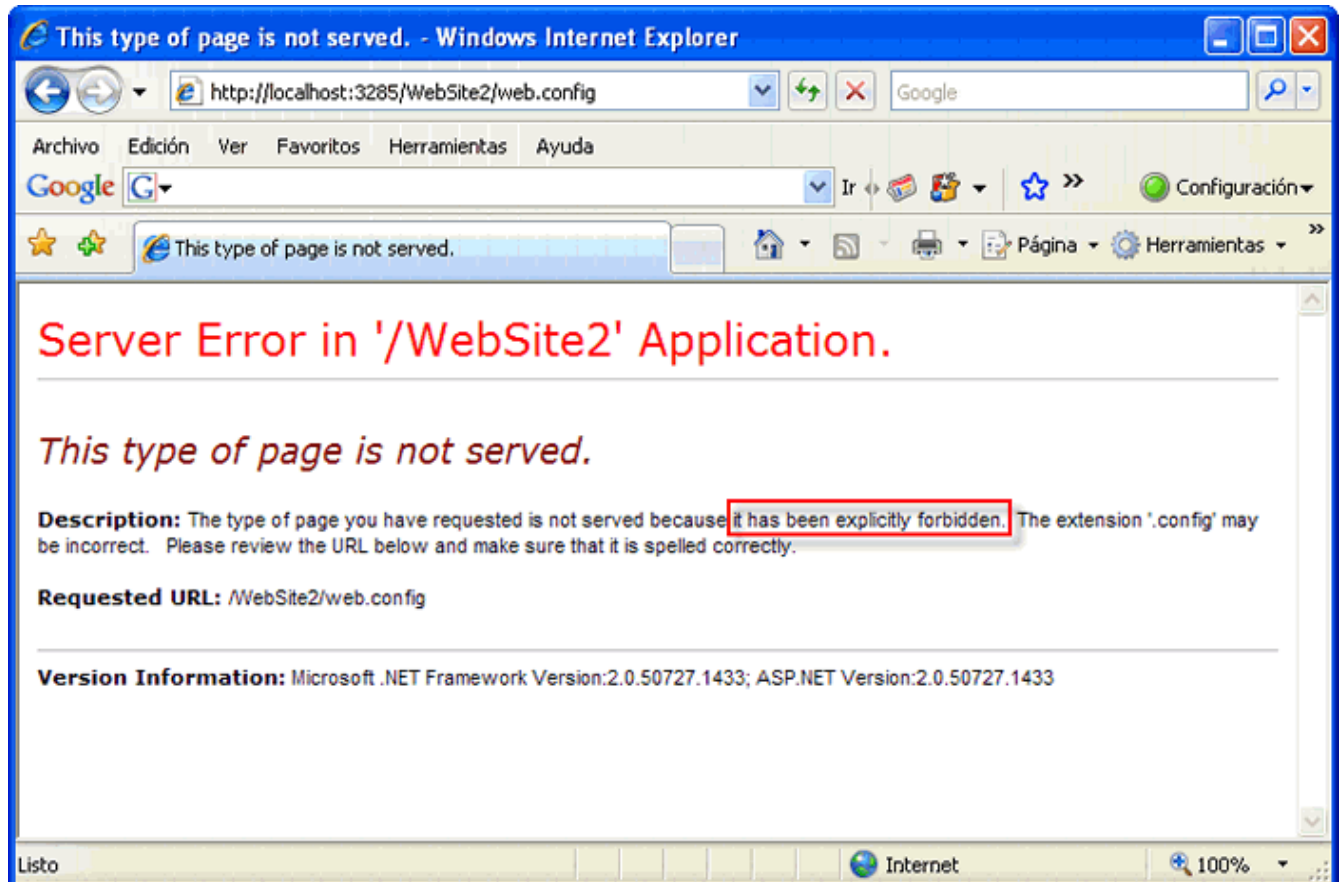
Para que nos cree el controlador del evento “Load” de la página, donde pondremos:

```
1 Imports System.Web.Configuration
2 Partial Class ejemplo_6_3
3     Inherits System.Web.UI.Page
4
5     Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Page_Load
6         Label1.Text = " La base de datos está almacenada en: "
7         Label1.Text &= WebConfigurationManager.AppSettings("ruta_base_de_datos")
8     End Sub
9 End Class
```

Fíjate que hemos importado el espacio de nombres de “System.Web.Configuration” para poder tener accesibles estos objetos, seguramente lo hayas notado al escribirlo ya que gracias al “intellisense” es fácil escribir el código del ejemplo. Ejecutamos esta página y tendremos:



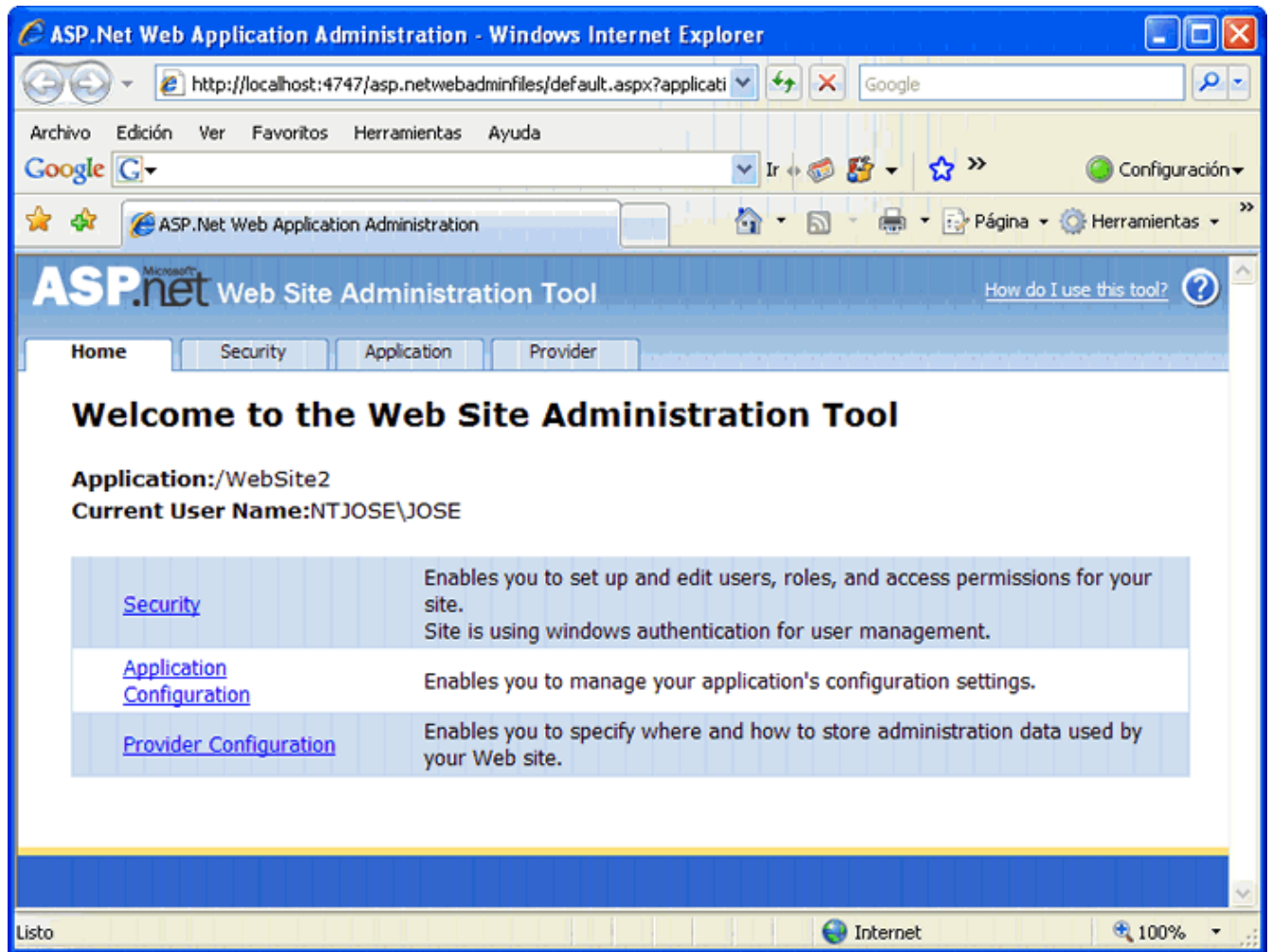
Bien, hemos recuperado el valor de una variable del fichero de configuración de nuestra aplicación web y la hemos mostrado en pantalla. Si algún gracioso quisiera leer esta página web con el navegador no podría porque nuestro TTS lo impide. Intentémoslo escribiendo la página “web.config” en el navegador:



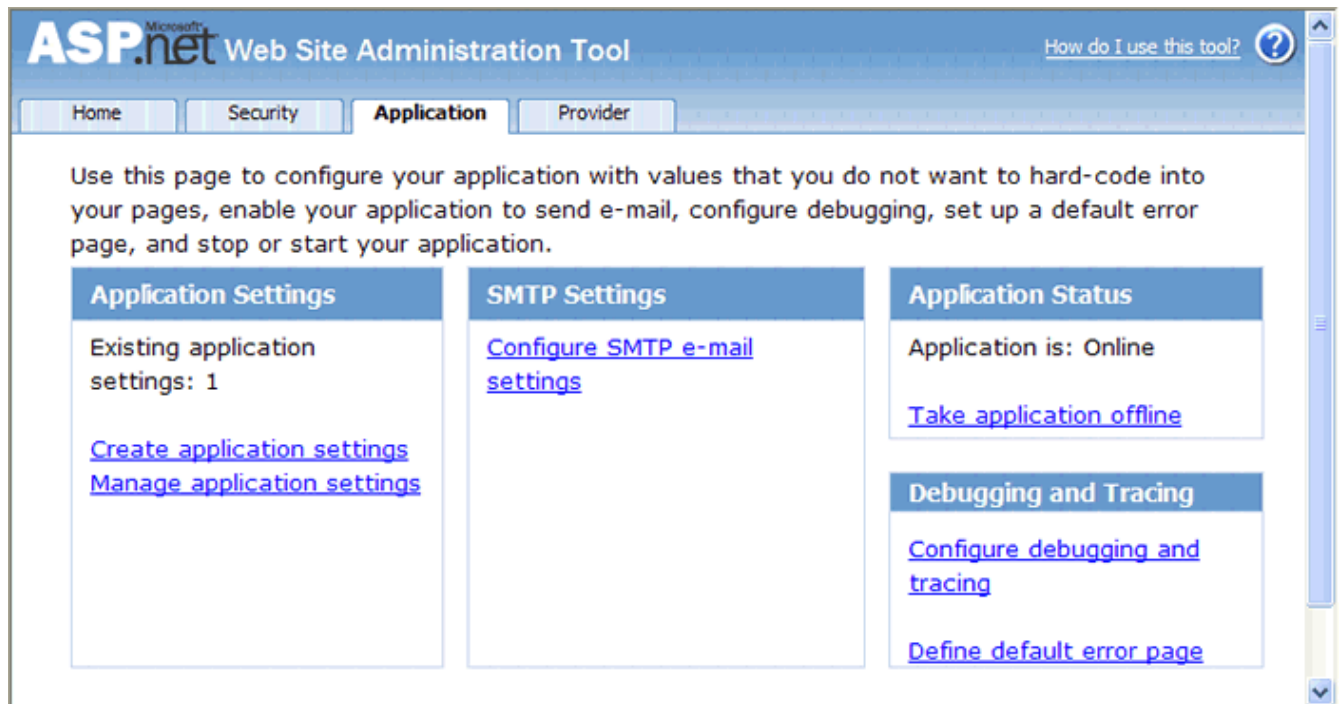
6. Administración del sitio web.

Mantener el fichero de configuración siempre es algo complejo, aunque sea sencillo de editar será mas fácil si tuviéramos una herramienta de configuración que nos mantuviese las opciones de este fichero. Afortunadamente tenemos un programa, mejor dicho, un pequeño web de administración para mantener ciertas partes de la configuración.

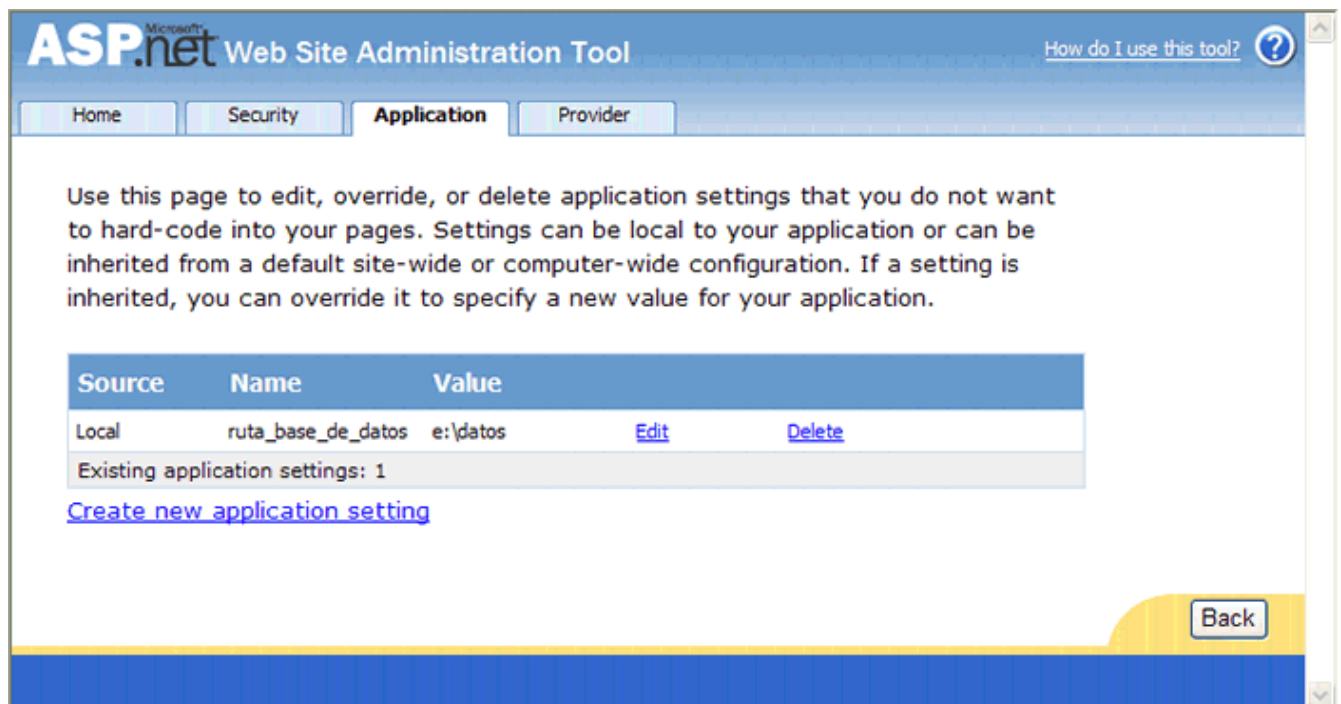
Para acceder a él nos iremos al menú y después seleccionaremos la opción WebSite y “ASP.NET Configuration“. En unos segundos veremos una página web como esta:



Puedes ver las distintas pestañas para las opciones de configuración de seguridad, aplicaciones y proveedores de información. Si hacemos clic en "Application":

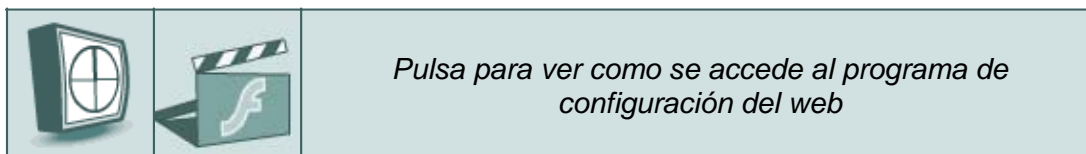


Veremos una lista de todos los valores de para configurar nuestra aplicación web, por ejemplo haz clic en “Manage Application settings“:



Donde podrás ver la variable que hemos creado antes para nuestra aplicación, y que llamamos: “ruta_base_de_datos“. Como ves, la idea no es mas que tener una herramienta gráfica para mantener mas fácilmente este fichero de configuración.

Formularios Web. La clase Page



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

Ejercicios

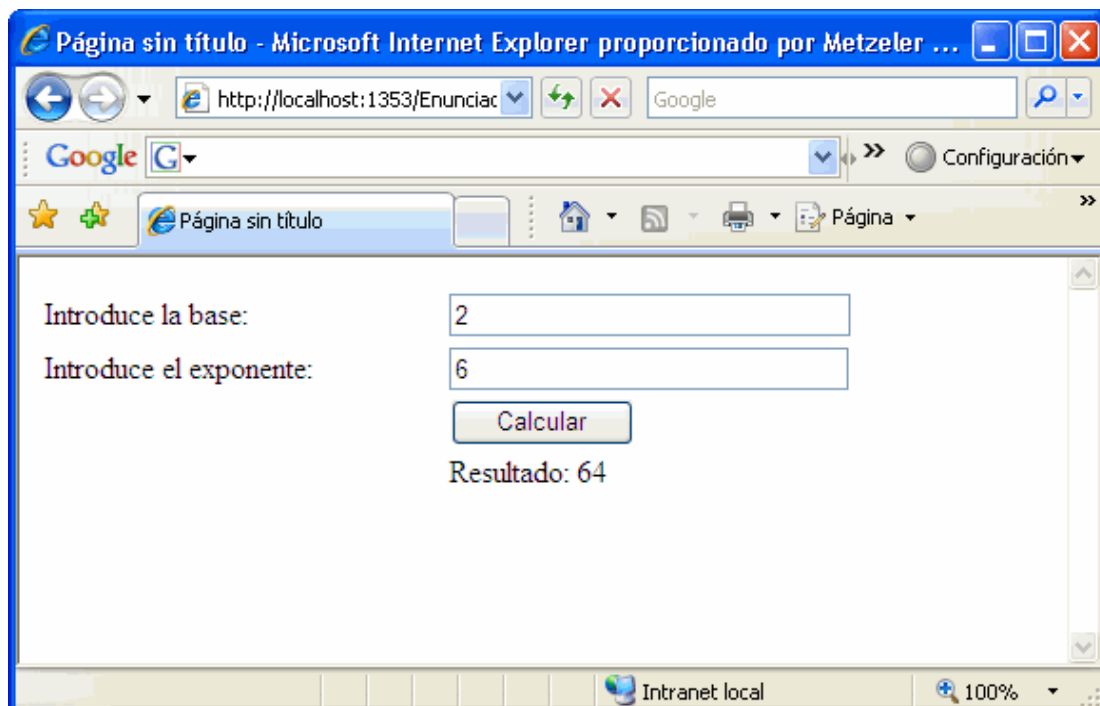
Hola, los ejercicios de este tema son de envío obligatorio. Si tienes alguna duda, ya sabes mándame una tutoria...

Ejercicio 1

Realiza la operación del exponente.

Introduce la base:	<input type="text"/>
Introduce el exponente:	<input type="text"/>
	<input type="button" value="Calcular"/>
	Resultado: [lb_resultado]

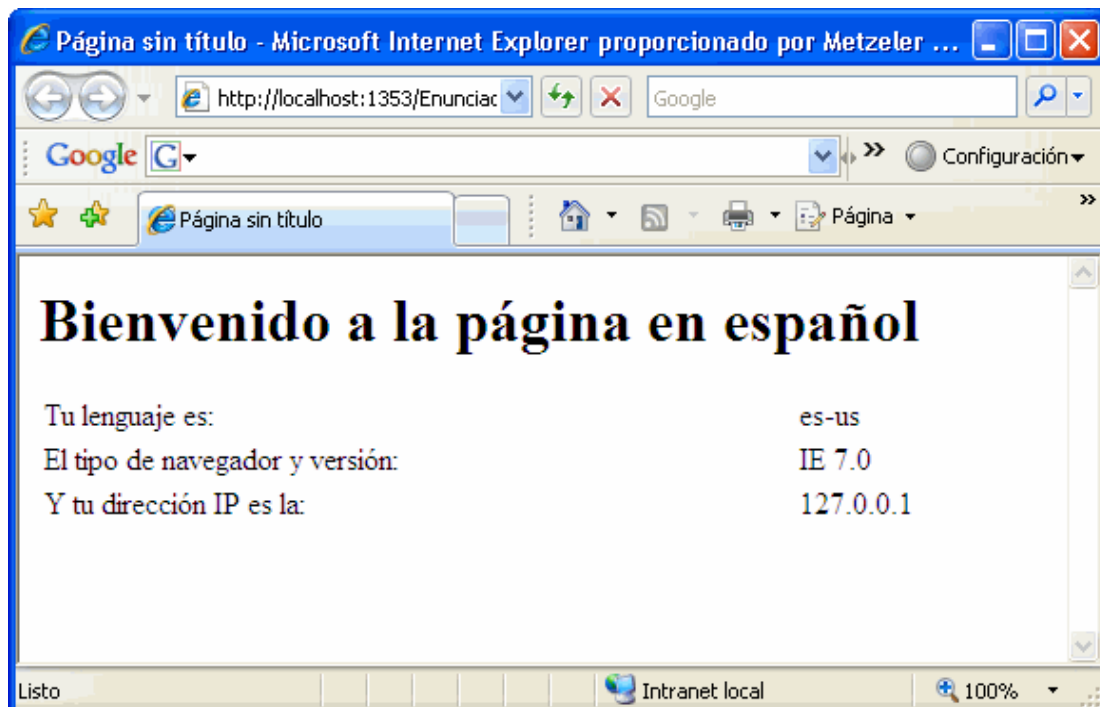
Para el cálculo hazlo manualmente, una solución para 2^4 sería $2 \times 2 \times 2 \times 2 = 16$. Es decir 2 elevado a la 4 será multiplicar 4 veces la base (2)



Ejercicio 2

Crea varias página simulando varios idiomas: inicio_es.htm inicio_fr.es ... y crea una página de inicio que lea el idioma del cliente y redireccione a estas páginas según éste. Para leer el idioma utiliza la variable de servidor: `Request.ServerVariables("HTTP_ACCEPT_LANGUAGE")`, haz el `"response.redirect"` oportuno

En la página para el español muestra el idioma, el navegador y su versión y la dirección IP del que se está conectando.



Ejercicio 3

Crea un formulario de este tipo y escribe los valores que ha seleccionado el usuario:

Formularios Web. La clase Page

Nombre:	<input type="text"/>	Estado civil:	<input checked="" type="radio"/> Soltero <input type="radio"/> Casado <input type="radio"/> Separado <input type="radio"/> Otros
Apellidos:	<input type="text"/>	Dispone de vehículo:	<input type="checkbox"/> [chk_vehiculo]
Dirección:	<input type="text"/>		
Población:	<div>Logroño Madrid Barcelona Valencia</div>		
<input type="button" value="Enviar datos"/>			
Resultado:	[lb_nombre]	Estado civil:	[lb_estado]
Dirección:	[lb_direccion]	Vehículo:	[lb_vehiculo]
Población:	[lb_poblacion]		

El resultado será la escritura de lo que ha seleccionado el usuario:

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

http://localhost:1353/Enunciados/tema_6/ejercicios06/ejercicic Google

Google Ir Configuración

Página sin título

Nombre: Estado civil: ☐ Soltero ☐ Casado ☒ Separado ☐ Otros

Apellidos: Dispone de vehiculo: ☒

Dirección:

Población:

Resultado:	Jose Rodriguez	Estado civil:	Separado
Dirección:	Espolón	Vehiculo:	Si
Población:	Madrid		

Intranet local 100%