Visual Studio 2005 y ASP .NET 2.0

3

Temas clave

- ***Entorno de desarrollo VWD***
- ***Ciclo de vida de una página Web***
- ***Cuadro de herramientas de VWD***
- ***Depuración de aplicaciones***
- ***Despliegue de aplicaciones***

¿Qué trataremos en este capítulo?

En este capítulo tomaremos contacto con el entorno de desarrollo y se aprenderá a crear una aplicación Web con los elementos mínimos para poder así conocer la interfaz de desarrollo de Visual Web Developer.

Conoceremos todo el ciclo de vida de una página Web para comenzar a dominar el concepto de página ASP y analizaremos en detalle el contenido del cuadro de herramientas de Visual Web Developer para tomar contacto con los diversos controles que podemos colocar en la página.

La tarea de desarrollo de una aplicación trae de la mano una actividad prácticamente paralela a la codificación: la depuración de aplicaciones. En este capítulo se aprenderá el uso de las herramientas de depuración, imprescindibles en todo desarrollo por más básico que éste sea.

La culminación del proceso de desarrollo es la implantación o despliegue de la aplicación en un servidor de producción; en la parte final de este capítulo se aprenderán los distintos mecanismos de despliegue que nos ofrece la plataforma .NET.

Entorno de desarrollo

Aunque parezca mentira, cuando se trata de productos de Microsoft, todo lo que se precisa para desarrollar aplicaciones ASP .NET 2.0 es gratuito para el desarrollador. Con sólo descargar .NET de la web de Microsoft y con un editor de texto cualquiera, incluso el Bloc de notas, podríamos desarrollar aplicaciones Web.

Es cierto que podríamos desarrollar con este entorno espartano de programación, pero sufriríamos algún estrés agudo después de algunas horas de abrir y cerrar los muchos archivos que componen una aplicación.

Microsoft propone como principal opción para el desarrollo Web su Visual Studio 2005. No obstante, tal como ya se mencíonó en el capítulo anterior, existe una solución alternativa para quienes no pueden hacer un desembolso de tantos euros como cuesta Visual Studio. Esta alternativa se denomina Visual Web Developer 2005 Express Edition y resulta suficiente para desarrollar las aplicaciones tratadas en este libro.

Aplicación Web para conocer la interfaz

Como siempre, el camino más rápido para comprender las funcionalidades de los formularios Web es verlos en acción y después analizarlos para comprobar cómo están construidos. Por lo tanto, antes de hacer una aplicación más seria y completa, veamos un formulario Web muy simple: el típico caso ¡Hola ASP.NET!

Un formulario Web está formado por dos componentes básicos: los elementos visuales de la interfaz y el código de lógica. Estos componentes pueden almacenarse en un archivo común o en archivos independientes.

Requisitos del entorno de producción

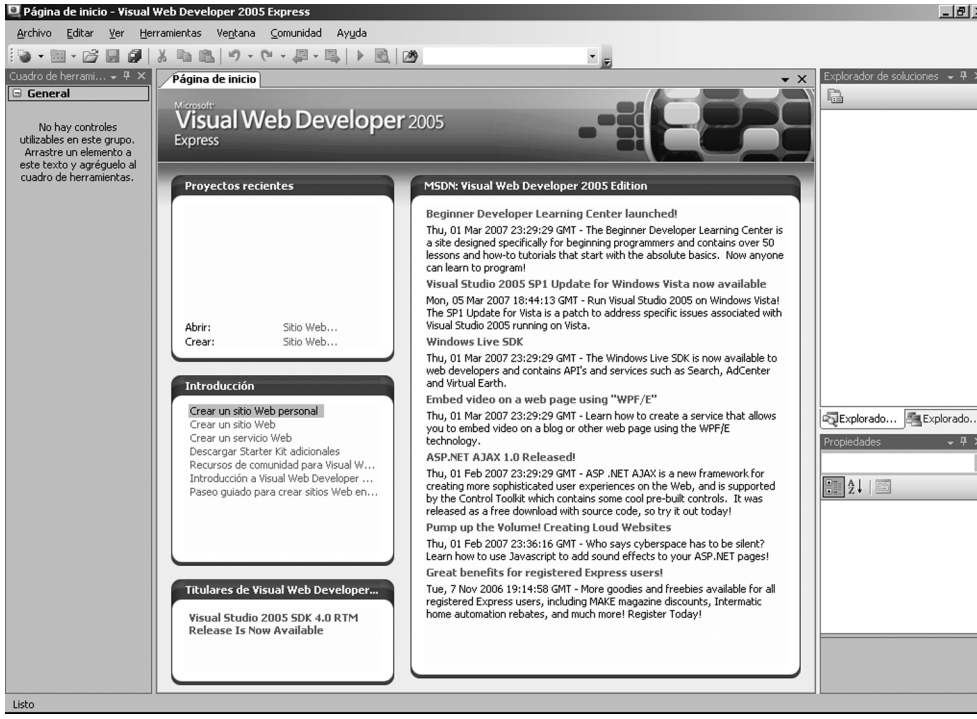
Aunque aún estamos en la fase de desarrollo inicial y de posterior prueba, debemos tener en cuenta que para poder poner en producción aplicaciones ASP.NET y Web Forms necesitaremos un servidor en el que se encuentre en ejecución **Internet Information Services (IIS)**. Además, Microsoft recomienda que el equipo servidor se encuentre formateado con NTFS (NT File System) en lugar de FAT (File Allocation Table). Esto brinda un mejor rendimiento y una seguridad sustancialmente mayor, así como también ofrece mayores opciones para el control de código fuente.

No obstante, en el entorno de desarrollo y prueba utilizaremos el servidor Web integrado con Visual Web Developer que actúa sin que tengamos que configurar nada en especial. El servidor Web se cierra al cerrar Visual Studio.

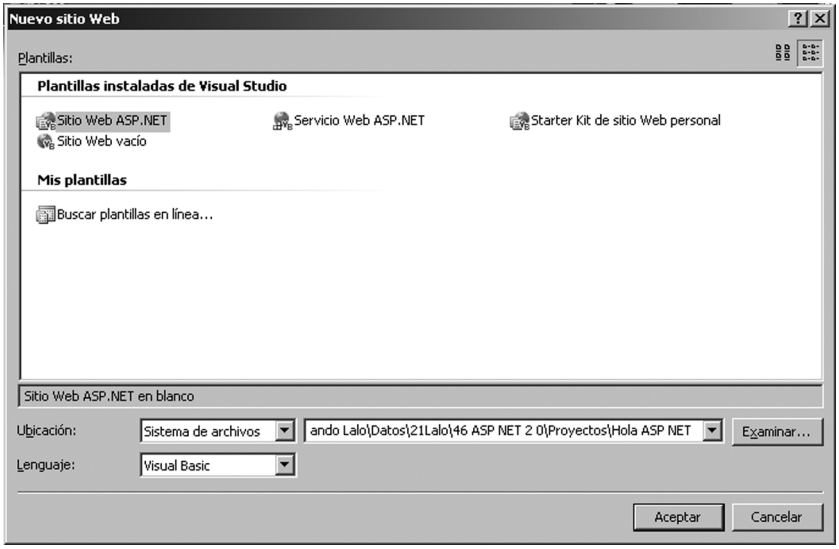
Creación de un nuevo sitio Web

Para crear nuestra primera aplicación Web utilizaremos Visual Web Developer y seguiremos estos pasos:

1. En Visual Web Developer crearemos un nuevo sitio Web.



2. De manera predeterminada queda seleccionada la plantilla Sitio Web ASP .NET. Al proyecto lo denominaremos `Hola ASP.NET` (el nombre del proyecto se escribe al final de la ruta de acceso de la carpeta del proyecto) y seleccionaremos como ubicación del proyecto la opción `Sistema de archivos`; el lenguaje de desarrollo será Visual Basic .NET:

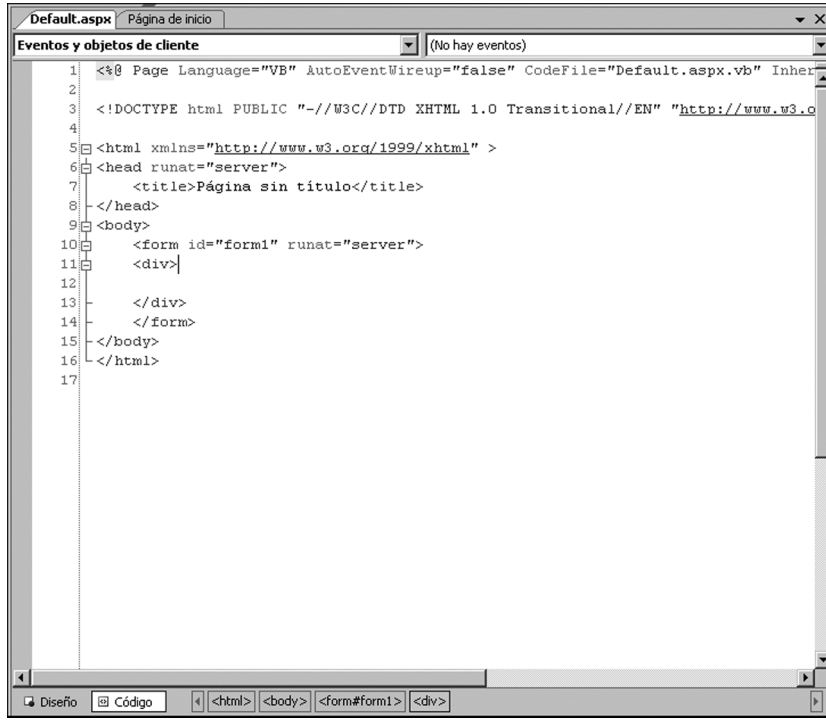


La lista desplegable `Ubicación` nos permite elegir uno entre tres tipos de sitios Web:

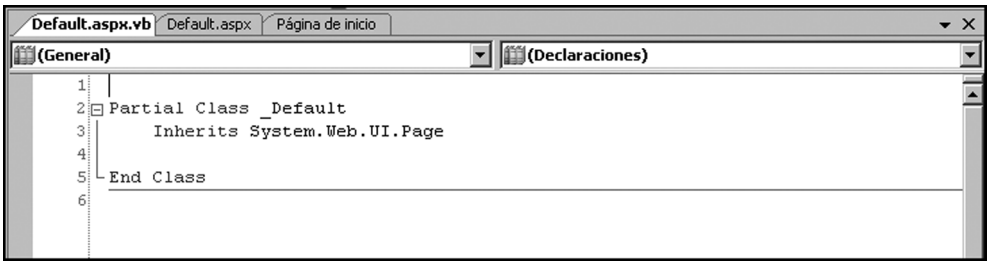
- **Sistema de archivos**: esta opción crea un sitio Web que se ejecuta con el servidor Web integrado que posee Visual Web Developer. Es una opción muy práctica para el entorno de desarrollo.
- **HTTP**: con esta opción crea un sitioWeb en un servidor IIS. El servidor IIS puede estar en el equipo local o en un equipo remoto. Es la opción que se utiliza en el desarrollo profesional.
- **FTP**: con esta opción crea un sitio Web en un servidor IIS de ubicación remota al que no se tiene acceso HTTP. FTP se utiliza para subir los archivos del sitio Web al servidor.

3. Haremos clic en `Aceptar` y se generará la nueva solución. Veamos ahora el Explorador de soluciones y comprobemos qué se ha creado: una carpeta `App_Data` y varios archivos. De manera predeterminada, Visual Web Developer ha creado un nuevo Web Form para nosotros, denominado `Default.aspx` (`.aspx` en la extensión para los archivos ASP.NET), con archivo de código Visual Basic, `Default.aspx.vb`, y un archivo de configuración, `web.config`:

Si observamos el código generado para este archivo nos encontraremos con este código HTML:



El código generado para el archivo `Default.aspx.vb` es el siguiente:

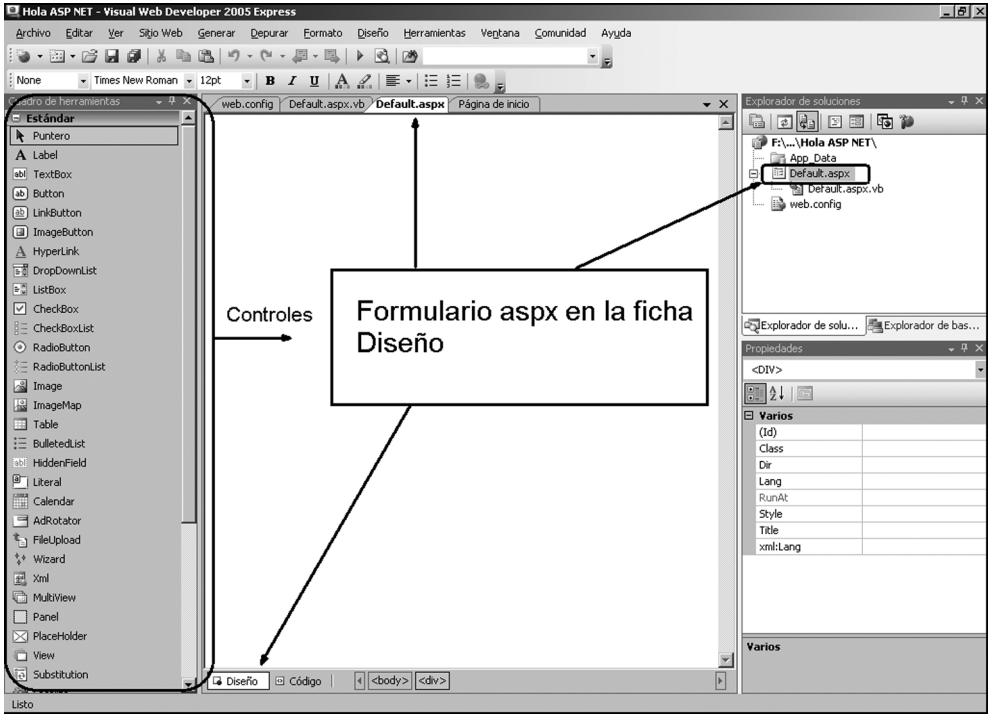


También echaremos un vistazo al archivo web.config estándar codificado en XML, aunque por ahora no le haremos ningún cambio. El archivo `Web.config` es un archivo relativamente nuevo de ASP que parece en la primera versión de .NET. Es un archivo formateado en XML que almacena los valores de configuración de nuestra aplicación Web. Incluye, entre otras cosas, el modo de depuración y las opciones de compilación.

```
<?xml   version="1.0"   encoding="utf-8"?>
<!—
       Nota: como alternativa para editar manualmente este archivo
     puede utilizar la herramienta Administración de sitios Web
     para configurar los valores de la aplicación. Utilice
      la opción Sitio Web->Configuración de Asp.Net en Visual Studio.
      Encontrará una lista completa de valores de configuración y
     comentarios en machine.config.comments, que se encuentra
     generalmente en
         \Windows\Microsoft.Net\Framework\v2.x\Config
—>
<configuration>
      <appSettings/>
       <connectionStrings/>
      <system.web>
          <!—
                Establezca debug="true" en la compilación para
                insertar símbolos de depuración en la página
                compilada. Dado que este proceso afecta al
                rendimiento, debe establecer este valor como true
                durante la depuración.

               Opciones de Visual Basic:
                Establezca strict="true" para no permitir las
                conversiones de todos los tipos de datos
                donde se pueda producir una pérdida de datos.
                Establezca explicit="true" para forzar la
                declaración de todas las variables.

          —>
```

```
            <compilation debug="false" strict="false" explicit="true" />
        <pages>
            <namespaces>
             <clear />
              <add namespace="System" />
               <add namespace="System.Collections" />
                <add namespace="System.Collections.Specialized" />
               <add namespace="System.Configuration" />
              <add namespace="System.Text" />
                <add namespace="System.Text.RegularExpressions" />
              <add namespace="System.Web" />
               <add namespace="System.Web.Caching" />
                <add namespace="System.Web.SessionState" />
               <add namespace="System.Web.Security" />
               <add namespace="System.Web.Profile" />
              <add namespace="System.Web.UI" />
               <add namespace="System.Web.UI.WebControls" />
                <add namespace="System.Web.UI.WebControls.WebParts" />
                   <add namespace="System.Web.UI.HtmlControls" />
            </namespaces>
        </pages>

        <!—
               La sección <authentication> permite configurar
               el modo de autenticación de seguridad utilizado por
               ASP.NET para identificar a un usuario entrante.
        —>

          <authentication mode="Windows" />

        <!—
              La sección <customErrors> permite configurar
              las acciones que se deben llevar a cabo/cuando un
              error no controlado tiene lugar durante la ejecución
              de una solicitud. Específicamente, permite a los
              desarrolladores configurar páginas de error html
              que se mostrarán en lugar de un seguimiento de pila
              de errores.

          <customErrors mode="RemoteOnly"
                    defaultRedirect="GenericErrorPage.htm">
                <error statusCode="403" redirect="NoAccess.htm" />
                 <error statusCode="404" redirect="FileNotFound.htm" />
         </customErrors>
        —>

    </system.web>
</configuration>
```

5. En la ficha `Diseño` podemos tratar el formulario `Web Hola ASPNET.aspx` como si fuese un formulario VB normal, colocando controles en él, arrastrándolos simplemente desde el cuadro de herramientas.



Nunca hay que perder la oportunidad de ver el contenido de los archivos generados por el asistente. Siempre nos enseñan algo que nos servirá en el futuro.

6. Antes de seguir adelante cambiaremos los nombres predeterminados para que todo quede más claro. Al archivo `Default.aspx` lo denominaremos `HolaASPNET`; este cambio lo haremos en el `Explorador de soluciones` y automáticamente también se cambia el nombre del archivo `.aspx.vb`. También cambiaremos el nombre de clase a `HolaASPNET`, en la ficha de código del archivo `Hola ASPNET.aspx.vb` y en la cláusula `Inherits` del archivo `Hola ASPNET.aspx`.

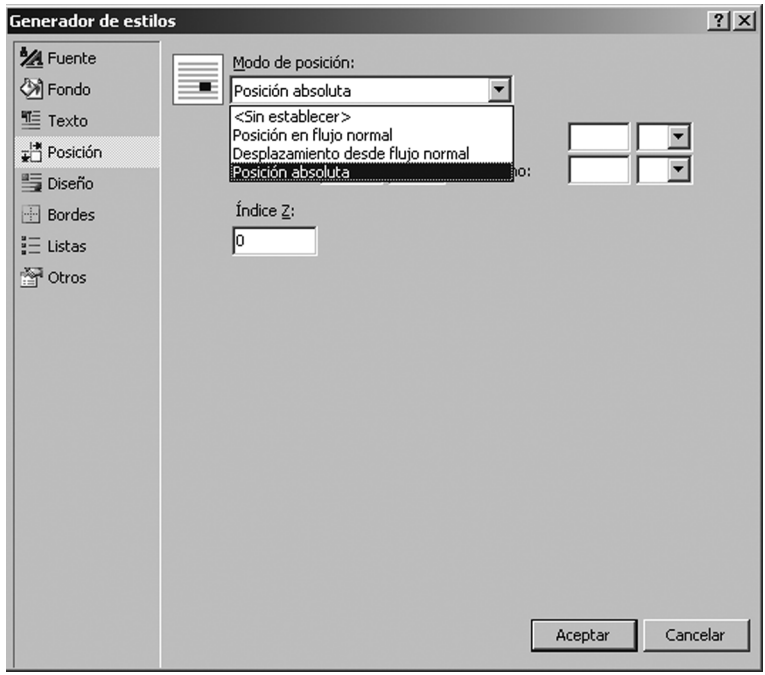
7. Volvemos a la ficha `Diseño` y desde el cuadro de herramientas, en la sección de controles estándar, arrastraremos un control `Label` y lo colocaremos en la parte superior izquierda del formulario y luego un control `Button` que colocaremos a continuación.

   Los controles en una página Web se pueden colocar de dos maneras básicas:

   - Modo flujo
   - Modo absoluto o cuadrícula

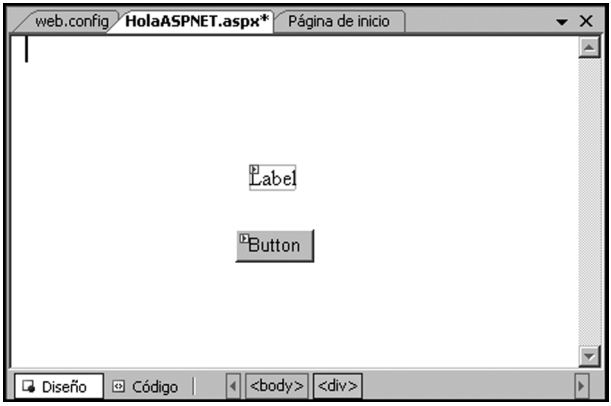
El modo estándar es el modo flujo, en el que los controles se van colocando una pegado a otro de izquierda a derecha siguiendo un "flujo" natural. Para diseñar la interfaz de un modo similar a las interfaces de las aplicaciones Windows tendríamos que elegir el modo absoluto.

8. Para acomodar los controles en cualquier parte de la página seleccionaremos cada uno de los controles en el diseñador y haremos clic con el botón secundario del ratón; del menú emergente elegimos la opción `Estilo` y del cuadro de diálogo que aparece utilizaremos la ficha `Posición`:



9. En la ficha `Posición` elegiremos `Modo de posicionamiento` igual a `Posicionamiento absoluto` y luego haremos clic en el botón Aceptar. A partir de este punto podremos reposicionar libremente en el control en toda la interfaz de usuario.

10. Repetiremos el paso 7 con el otro control (`Button`). La interfaz quedará con un aspecto similar a este:



11. Desde la ventana `Propiedades`, asignamos la propiedad `ID` del control `Label` (que define su nombre) con el valor `lblTexto`. Asignamos el texto "`Pulsar el botón para iniciar`" en la propiedad `Text`.

Si la ventana Propiedades no está visible podemos hacerla aparecer con F4.

12. Después hacemos clic en el control `Button` y le asignamos el valor `btnIniciar` en su propiedad `ID`. Hacemos clic una vez sobre el botón y pulsamos la tecla `Intro` o hacemos clic con el botón derecho sobre el botón y seleccionamos `Ver código`. Introduciremos el siguiente código para el evento `Click`:

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

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

             lblTexto.Text  =  "Hola  ASP.NET"

      End  Sub
End  Class
```

Al codificar la línea de código podemos comprobar cómo nos ayuda IntelliSense, tal como lo hace en un proyecto estándar de Windows.

---

13. Para tener la ayuda de la depuración de código tendremos que modificar el archivo `web.config`, en donde aparece esta línea:

```
          <compilation   debug="false"   strict="false"   explicit="true"/>
```

Cambiarla a

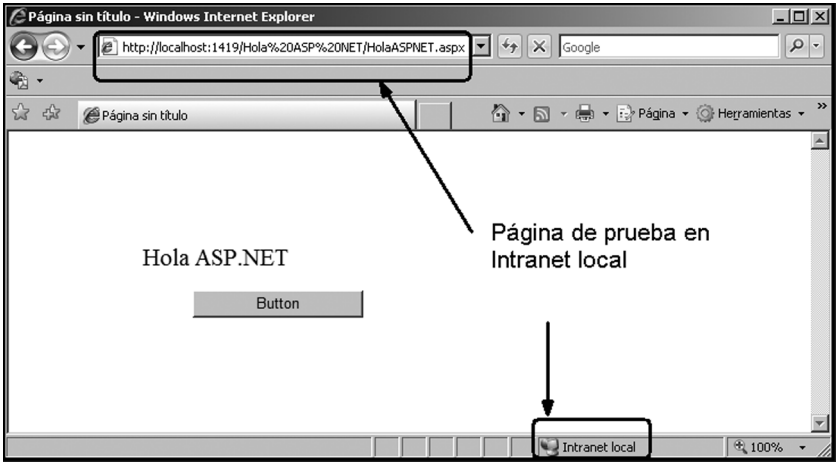
```
          <compilation   debug="true"   strict="false"   explicit="true"/>
```

Esto nos permite establecer puntos de interrupción en el código para comprobar el estado de las variables y de los objetos de la página en determinados puntos de la ejecución.

Para codificar en C# en lugar de Visual Basic se debe cambiar el atributo Language de la directiva Page de VB a C# y además modificar la extensión del archivo de vb a cs en el atribute CodeFile.

---

14. Ahora ya podemos probar la página desde el mismo IDE de Visual Web Developer haciendo clic en `Depurar/Iniciar depuración` o desde `Generar/Generar sitio Web`.



Cuando hacemos clic en el botón `Button`, se ejecuta el código subyacente del botón, tal como sucedería también con una aplicación para Windows.

La ejecución también se puede iniciar con F5 o hacuiendo clic con el botón secundario del ratón en Default.aspx en la ventana del Explorador de soluciones y eligiendo la opción Ver en el explorador.

Carpetas de una aplicación ASP

Aunque en este ejemplo sólo se ha utilizado una única carpeta (App_Data), ASP.NET reconoce una serie de carpetas que utiliza para determinados contenidos. Veamos brevemente la utilidad de cada una de estas carpetas:

- **App_Code**: contiene código fuente para clases y objetos de negocio que queremos compilar como parte de la aplicación Web. ASP.NET recompila lo que haya en esta carpeta cuando detecta que se han producido cambios. El código que esté dentro de esta carpeta estará referenciado automáticamente dentro de la aplicación.
- **App_Data**: contiene archivos de datos de la aplicación, por ejemplo, archivos de bases de datos o XML.
- **App_GlobalResources**: contiene recursos de la aplicación que se compilan en los ensamblados con ámbito global.
- **App_LocalResources**: contiene recursos asociados a un página determinada, a un control de usuario o a una página maestra de la aplicación.

- **App_WebReferences**: contiene archivos de contratos de referencia, esquemas y archivos tipo .disco (descovery) que definen una referencia Web que utilizará la aplicación.
- **Bin**: contiene ensamblados compilados para controles, componenentes u otra clase de código referenciado en la aplicación. El código que esté dentro de esta carpeta estará referenciado automáticamente dentro de la aplicación.
- **Themes**: contiene archivos relacionados a los temas, que es una nueva funcionalidad de ASP .NET que nos ayuda a mantener una presentación uniforme en el sitio Web a lo largo de las distintas páginas.

¿Qué es es un formulario Web?

Con .NET Microsoft ha logrado que los formularios Web establezcan un puente entre la programación VB y la tradicional programación ASP al ofrecer una técnica visual de arrastrar y soltar controles en la página y la posibilidad de codificar eventos para los controles; de esta manera, los Web Forms se crean y se ejecutan de un modo muy familiar para el desarrollador.

Tal como hemos podido comprobar en el pequeño sitio Web de presentación (Hola ASP.NET), un formulario Web está formado por dos componentes:

- Elementos visuales que podemos ver en la vista de diseño; estos elementos componen la plantilla para la presentación de la página Web en el navegador del usuario final.
- El código que hay detrás de los controles y la página; el código se ejecuta en el servidor cuando la página se carga y en respuesta a otros eventos previstos en la implementación de nuestro sitio Web.

Mediante la división de componentes de una página web en archivos independientes los Web Forms presentan un entorno que resulta muy familiar para los programadores VB y VB.NET.

En un entorno de programación con interfaces gráficas primero se dibuja el formulario mediante la técnica de arrastrar y soltar controles en él y después se incluye el código para los eventos que exponen los controles. De modo similar, cuando se desarrolla con Web Forms, primero crearemos el aspecto de la página Web utilizando la técnica de arrastrar y soltar controles en la página y después codificaremos los eventos expuestos por esos controles.

Plantilla para presentación

El archivo `.aspx` representa el componente de la interfaz de usuario del formulario Web y se utiliza como plantilla para su presentación en el navegador. Este archivo `.aspx` es la propia página y contiene el código HTML y los elementos Web Forms específicos. En el formulario Web podemos arrastrar y soltar distintos tipos de controles, entre los que se incluyen:

- Controles HTML
- Controles Web Form
- Controles de validación
- Controles relacionados con datos
- Componentes COM y .NET registrados en nuestra máquina

Veremos posteriormente estos diferentes tipos de controles.

El código que hay detrás de cada página

El archivo .vb que acompaña al archivo .aspx contiene el componente de código del formulario Web. Al visualizar este archivo en el editor de código nos encontramos con estas líneas:

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

End  Class
```

Definiciones parciales de clases: la palabra clave Partial en la definición de una clase indica que esta clase se puede definir en varias declaraciones. La restricción es que las definiciones parciales deben estar todas en el mismo ensamblado y en el mismo espacio de nombres.

La clase HolaASPNET, es decir, la que hemos creado para nuestro formulario Web, hereda de la clase System.Web.UI.Page. Esto permite que el código dentro de nuestro Web Form pueda acceder a los objetos incorporados Request, Response, Session, Application y Server que nos resultarán útiles para la gestión entre páginas de la aplicación Web; en este ejemplo no los hemos utilizado ya que se trataba de una página muy simple.

Cada control que coloquemos en el formulario, y, por lo tanto en el archivo .aspx, estará representado dentro del archivo .vb y podremos codificar los eventos que nos interesen para su ejecución en el servidor.

En nuestro caso, tenemos el evento Click del botón btnIniciar representado por este código:

```
      Private  Sub  btnIniciar_Click(ByVal  sender  As  System.Object,  _
                                            ByVal  e  As  System.EventArgs)  _
                                            Handles  btnIniciar.Click
```

```
        lblTexto.Text = "Hola ASP.NET"

    End Sub
```

Ciclo de una página Web

Veamos qué sucede cuando un usuario desde un navegador solicita una página `.aspx` a un servidor Web:

- En el equipo del usuario: El usuario hace referencia a una página.
- En el servidor Web: se carga la página referenciada.
- En el servidor Web: se procesa el código de la página.
- En el servidor Web: se envía el resultado HTML al navegador.
- En el equipo del usuario: el navegador presenta la página.
- En el equipo del usuario: el usuario puede interactuar con la página de modo local hasta que produce un nuevo envío al servidor para procesar alguna acción que tenga que procesarse en el servidor, esto provoca que el ciclo vuelva a repetirse.

Para que un proceso en el servidor supere las limitaciones de un protocolo sin estado como HTTP existen varias alternativas: ASP.NET mantiene el estado de los controles en la propiedad `ViewState`.

Si nos interesa dividir el proceso de una página Web en diversas fases podemos hacerlo de la siguiente manera:

- Inicio
- Carga de valores ViewState
- Validación
- Control de eventos

La iniciación de la página queda determinada por el evento `Init` y después de este evento se produce el evento `Load`.

Por ejemplo, en la carga de la página del ejemplo `Hola ASP.NET` podríamos asignar un valor determinado a alguna propiedad de la etiqueta `lblTexto`:

```
    Protected Sub Page_Load(ByVal sender As Object, _
              ByVal e As System.EventArgs) _
```

```
            Handles Me.Load

        lblTexto.BackColor = Drawing.Color.Beige

End Sub
```

El desarrollo ASP siempre estuvo relacionado con la generación de una página HTML más el añadido de código de secuencia de comandos. En las primeras versiones de ASP la página Web era un archivo de texto simple separado en bloques de código ASP y código HTML. Cuando desde un navegador se requería una página ASP al servidor Web, en donde el motor ASP analizaba la página antes de enviar la salida de respuesta al navegador. En tiempo de ejecución, el motor ASP interpretaba el código ASP línea a línea; ejecutaba cada línea que contenía código script ASP mientras que producía la salida sin cambios de cada línea de HTML puro. Por lo tanto, el modelo de desarrollo Web de ASP tradicional estaba compuesto por unas páginas HTML con un añadido de código.

En ASP.NET el enfoque cambió radicalmente: cada página es en realidad un programa ejecutable. La ejecución de la página da como resultado texto HTML. De este modo podemos centrarnos en el desarrollo con controles y elementos de código que generan salida HTML, en lugar de preocuparnos en el código que se intercala con el texto HTML.

Cuando se utiliza VS.NET para implantar una aplicación Web, se utiliza el modelo estándar de aplicaciones VB: el proyecto se compila y se implantan los archivos resultantes. Todos los archivos de código (salvo los archivos `.aspx`) de cada formulario Web se compilan en una DLL junto a todos los archivos ejecutables del proyecto. La biblioteca DLL se distribuye posteriormente en el servidor Web como una única unidad, sin su código fuente. Cuando el navegador solicita el archivo `.aspx` de la página, el archivo DLL y el archivo `.aspx` se compilan en una nueva clase y después se ejecutan.

Web Forms deriva su funcionalidad de la clase `System.Web.UI`. Cada archivo `.aspx` representa una página Web en una aplicación Web. Cuando un navegador requiere por primera vez un archivo `.aspx`, ASP.NET genera dinámicamente el archivo de clase para la página mediante la herencia desde el espacio de nombres `System.Web.UI.Page`. Esta clase expone los objetos `Request`, `Response`, `Server`, `Application` y `Session`, con sus propiedades y métodos que ya resultan familiares para los programadores clásicos ASP.

Cuadro de herramientas de Visual Web Developer

Si bien los controles Web son estéticamente semejantes a los controles utilizados en las aplicaciones para Windows en realidad son muy diferentes. La razón de esto es que los controles Web Form operan dentro de la plataforma de la página ASP.NET. Existen tres clases básicas de controles para utilizar en los formularios Web:

- Controles de servidor HTML
- Controles de servidor ASP.NET
- Controles de validación

Definición de controles en el lado del servidor

Si necesitamos crear una aplicación Web interactiva tendremos que responsabilizarnos de la obtención de los datos desde el objeto `Request`, pasarlos al navegador cuando la página retorna y mantenerlos bajo control. La tarea de mantener el estado es una de los desafíos de las aplicaciones Web.

Analicemos qué sucede en una aplicación Web en la que se tiene un formulario con un cuadro de texto simple y un botón que ordena el envío del formulario.

Cuando se envía el formulario, supongamos que se devuelve con el valor del cuadro de texto sin cambios. Para poder hacer esto, necesitaremos codificar un formulario de la siguiente manera:

```
<html>
<head>
      <title>Ejemplo  formulario  HTML/ASP</title>
</head>

<body>
    <form  action="test.asp">
       Informe  su  identificador:
        <input  type="text"  name="iden"  value="<%=  Request("iden")  %>"
size="10"
                maxlength="10"><br>
       <input  type="submit"  name="cmd"  value="  Submit  ">
   </form>
</body>
</html>
```

El programador es quien debe hacerse responsable del mantenimiento del valor del texto introducido en el cuadro de texto "`iden`" y de devolverlo al navegador del usuario:

```
value="<%=   Request("iden")   %>"
```

El valor se obtiene del cuadro de texto ("iden") del objeto `Request`, y usando ese valor como atributo `Value` del cuadro de texto. En la primera solicitud de la página, el objeto `Request` no tiene ningún valor denominado `iden`, por que estará en blanco. Cuando el usuario introduzca un valor y haga clic en el botón `Submit`, el formulario se enviará al servidor Web pero esta vez con un valor en el cuadro de texto.

Con ASP.NET se ha incorporado un nuevo mecanismo que se ocupa automáticamente de la gestión del estado sin que tengamos que codificar nada especial. Web Forms nos permite indicar que necesitamos que un determinado control debería mantener estado cuando se envía desde el navegador al servidor.

Esto se consigue con el atributo `runat="server"`, tanto para los controles del formulario como para el propio formulario. Este atributo hace que nuestros controles del formulario se comporten como **controles del lado del servidor** en lugar de ser controles del lado del cliente. Volviendo al ejemplo del formulario anterior podríamos escribirlo así:

```
<head>
      <title>Ejemplo formulario HTML/ASP.NET</title>
</head>

<body>
    <form action="test.aspx" runat="server">
      Informe su identificador:
       <asp:textbox runat="server" name="iden"
                    size="10" maxlength="10" /><br>
      <input type="submit" name="cmd" value=" Submit ">
   </form>
</body>
</html>
```

La etiqueta `FORM` tiene una indicación con el atributo `runat="server"`:

```
<form  action="test.aspx"  runat="server">
```

Esto hace que el entorno de ejecución ASP.NET cree código adicional para gestionar el estado del formulario; lo que se implementa mediante el añadido de un campo oculto del formulario. Si ahora visualizásemos el código fuente del archivo .aspx en el navegador veríamos que aparece un nuevo elemento de tipo oculto con el nombre `name="__VIEWSTATE"` y con un valor codificado. ASP.NET utiliza este campo oculto para transferir la información de estado entre el navegador y el servidor Web. Comprime la información necesaria en un valor de campo cifrado. De esta manera, a todos los controles de la página Web que necesiten mantener su información de estado se le añadirá automáticamente un campo oculto.

ASP.NET también permite pasar eventos entre páginas mediante el mecanismo de Cross Page Posting. Se utiliza la propiedad PostBackUrl del control cuyos eventos queremos gestionar desde otra página asignándole en esa propiedad la ruta de acceso virtual a dicha página. Esto se complementa con la propiedad PreviousPage de la nueva página, en la que recibirá los datos de los controles de la página anterior.

Controles del servidor HTML

Ya sabemos qué son genéricamente los controles en el lado del servidor. Los controles del servidor HTML son elementos HTML expuestos al servidor utilizando el atributo `runat="server"`. En realidad, los controles del servidor HTML son idénticos en apariencia y en comportamiento a los controles HTML comunes, salvo que incluyen el atributo `runat="server"`, lo que permite programarlos dentro del entorno de la página Web Form.

Los controles del servidor HTML están disponibles para la mayoría de los elementos HTML utilizados en una página Web: por ejemplo, la etiqueta `FORM`, los distintos elementos `<input>` de HTML (`TextBox`, `CheckBox`, botón `Submit`), `ListBox` (select), `Table` e `Image`.

En el entorno de desarrollo Visual Web Developer se puede crear un control HTML normal haciendo clic en el grupo HTML dentro del cuadro de herramientas y después arrastrando un control, por ejemplo, `Text` hasta el formulario Web. Le podemos asignar un nombre identificador cambiándole su propiedad `ID`. Esto dará como resultado un control HTML normal con este código:

```
<input id="Text1" type="text" /></div>
```

La conversión de un control HTML normal en un control de servidor HTML (o viceversa) se consigue simplemente haciendo clic con el botón derecho del ratón sobre el control en modo de diseño y seleccionando (o anulando la selección) la opción del menú `Ejecutar como control del servidor`. Al seleccionarlo, se obtiene este código:

```
<input id="Text1" type="text" runat="server" /></div>
```

Los controles HTML se crean a partir de clases en el espacio de nombres `System.Web.UI.HtmlControls` de la biblioteca de clases de .NET Framework. ASP.NET trata a los controles HTML normales como tales, es decir como simples controles HTML.

Pero al convertirlos en controles del servidor HTML, capacidad funcional se modifica sustancialmente:

- Nos permite que codifiquemos eventos generados en el control que se ejecutan en el lado del servidor, en lugar de ejecutarse en el lado del cliente. Por lo tanto, podemos responder con código en el lado del servidor al evento `Click`, por dar un ejemplo, de cualquier de estos controles.
- No perdemos la capacidad para codificar eventos en la secuencia de comandos en el cliente. Como siempre ha sucedido, al tratarse también de controles HTML estándar, estos controles mantienen la funcionalidad que les permite gestionar secuencia de comandos en el lado del cliente.

- Implementa automáticamente el mecanismo que permite mantener los valores del control en los viajes de ida y vuelta entre navegador y servidor.

La presencia de los controles del servidor HTML también facilita la compatibilidad hacia atrás con las aplicaciones ASP existentes y la migración de ASP a ASP.NET. No obstante todo lo dicho, se debe tener en cuenta que todo lo que se puede hacer con controles del servidor HTML, también se puede hacer, con mayor control y eficacia mediante el uso de los nuevos controles del servidor ASP.NET.

En el cuadro de herramientas de Visual Web Developer se incluyen los siguientes controles HTML:

Control	Objetivo
Input (Hidden)	Para campos ocultos que se envían con el formulario y sin representación visual. Se implementa como <input type="hidden">.
HorizontalRule	Define una línea horizontal. Se implementa con la etiqueta <hr>
Image	Define una imagen. Se implementa con la etiqueta <img>.
Input (Button)	Botón para activar un evento conectado al control. Se implementa como <input type="button">.
Input (CheckBox)	Cuadro de verificaciones que permite elegir varias opciones en todo el grupo. Se implementa como <input type="checkbox">.
Input (File)	Para indicar un archivo. Se implementa como <input type="file">.
Input (Password)	Para introducir un campo de contraseña. Se implementa como <input type="password">.
Input (Radio)	Botones de opciones que permite elegir una única opción en todo el grupo. Se implementa como <input type="radio">
Input (Submit)	Botón para enviar datos al servidor. Se implementa como <input type="submit">.
Input (Text)	Para introducir un campo de texto en una línea. Se implementa como <input type="text">.
Input(Reset)	Botón para reasignar el contenido del formulario HTML. Se implementa como <input type="reset">.
Select	Para seleccionar opciones. Se implementa con la etiqueta <select>.
Table	Define una tabla. Se implementa con la etiqueta <table>.
Textarea	Para introducir un campo de texto en más de una línea. Se implementa como <textarea>.

Controles del servidor ASP.NET

Visual Web Developer se distribuye con varias decenas de controles del servidor ASP.NET, desde sencillos controles como `TextBox` o Button, hasta controles del servidor más complejos para el enlace de datos. Si utilizamos el examinador de objetos de Visual Studio encontraremos a todos estos controles en el espacio de nombres `System.Web.UI.WebControls`.

La utilización de estos controles desde el cuadro de herramientas es similar a lo ya conocido del entorno de aplicaciones para Windows.

En el documento .aspx, estos controles utilizan el prefijo `asp:`. Por ejemplo, si colocamos un control `Label` en la página y le asignamos su propiedad `Text` con el valor `Nombre`, el control `Label` aparece en la vista de código del documento `.aspx` con la siguiente codificación generada automáticamente por el entorno .NET:

```
<asp:Label ID="Label1" runat="server" Text="Nombre">
      </asp:Label>
```

Tal como se puede observar el código no corresponde a un control HTML normal, sino que se trata de un control ASP.NET y, por supuesto, los atributos se refieren a propiedades del control ASP.NET.

Sin embargo, cuando llega el momento de la ejecución de la página, todo control ASP.NET se genera en la página Web utilizando HTML simple, lo cual depende del tipo de navegador así como también de la configuración del control.

En la situación en que una misma función podría ser implementada tanto con un control ASP.NET como con un control HTML se debería elegir el control ASP.NET sólo si le vamos a sacar provecho con código en el servidor, en caso contrario, el control HTML debería ser la elección adecuada ya que provoca menos carga de proceso.

Con Visual Web Developer se distribuyen los siguientes controles del servidor ASP y están disponibles para su uso en un Web Form, los podemos encontrar utilizar directamente desde el Cuadro de herramientas:

Control	Objetivo
AdRotator	Visualiza una secuencia de anuncios en la página Web.
BulletedList	Crea un control que genera una lista de elementos con formato de viñetas.
Button	Visualiza un botón, usado normalmente para realizar alguna acción. Puede ser de tipo submit (sin propiedad `CommandName`) o de tipo command. Se representa en la página como un elemento `INPUT`.

Control	Objetivo (*continuación*)
Calendar	Crea un calendario interactivo.
CheckBox	Visualiza un cuadro de verificación simple que permite al usuario la elección de una opción del estilo Sí/No o Verdadero/Falso. Se representa en la página como un elemento `INPUT`.
CheckBoxList	Crea un conjunto de cuadros de verificación en forma de grupo que permite la selección múltiple.
DataList	Visualiza filas de una base de datos con un formato personalizable.
DetailsView	Se utiliza para mostrar un único registro de un origen de datos en una tabla. Suele usarse como detalle en el enfoque maestro-detalle.
DropDownList	Presenta una lista en un cuadro desplegable combinable del que se puede hacer una selección.
FileUpload	Crea un conjunto de controles (un cuadro de texto y un botón `Examinar`) para que el usuario pueda elegir un archivo que se cargará en el servidor. Se representa en la página como un elemento `INPUT`.
FormView	Actúa como contenedor de controles; está compuesto por plantillas totalmente personalizables.
GridView	Visualiza información (normalmente desde una base de datos) en formato tabular con filas y columnas, se pueden seleccionar y editar elementos.
HiddenField	Crea un campo oculto para almacenar un valor que no se visualiza. Se representa en la página como un elemento `INPUT`.
HyperLink	Crea un hipervínculo para gestionar la navegación.
Image	Permite la visualización de una imagen en la página.
ImageButton	Variante de `Button`: se visualiza un botón con una imagen. Se representa en la página como un elemento `INPUT`.
ImageMap	Crea un gráfico con regiones de punto sensibles ("Hot spots") en las que el usuario puede hacer clic.
Label	Visualiza un texto no editable controlable por programa.
LinkButton	Variante de Button: se comporta como un botón, pero se visualiza como un hipervínculo.
ListBox	Presenta una lista de elementos de la que se puede hacer una selección simple o múltiple.
Literal	Para añadir texto dinámico sin necesidad de incluir nuevas etiquetas HTML.
Localize	Tiene alguna similitud con `Label` o `Literal`. Sirve para crear una localización en la página Web para visualizar un bloque localizado de texto estático.
MultiView	Actúa como contenedor de controles View y permite presentar vistas alternativas de la información.

Control	Objetivo (*continuación*)
Panel	Suministra un contenedor en la página para colocar otros controles.
RadioButton	Visualiza un botón de radio o de opciones. Se representa en la página como un elemento `INPUT`.
RadioButtonList	Visualiza un grupo de botones de radio o de opciones donde sólo se puede seleccionar un botón de radio.
Repeater	Visualiza información de una base de datos repitiendo la visualización para cada registro.
Substitution	Este control especifica una sección en una página Web que queda excluida de la operación de almacenamiento en memoria cache, donde se quiere una actualización dinámica del contenido.
Table	Crea una tabla en la página Web.
TextBox	Visualiza un cuadro en el que se puede editar texto. Se representa en la página como un elemento INPUT.
View	Representa un control que contiene otros controles dentro de un control `MultiView`.
Wizard	Suministra un mecanismo de navegación y una interfaz de usuario para obtener datos en diversos pasos, tal como se realiza en los asistentes.
XML	Permite leer y escribir datos en formato XML.

Controles de acceso a datos

Control	Objetivo
SqlDataSource	Nos permite utilizar un control Web para tener acceso a los datos almacenados en una base de datos relacional o en orígenes de datos OLE DB y ODBC.
AccessDataSource	Funcionalmente similar al control SqlDataSource pero es específico para base de datos Access (archivos .mdb).
ObjectDataSource	Actúa como objeto intermediario para la recuperación de datos y de actualización de un objeto de datos genérico.
XMLDataSource	Gestiona datos XML para controles enlazados a datos.

Controles de validación

Es habitual que nos encontremos con la necesidad de comprobar que los usuarios han escrito la información con el formato correcto o que no han omitido ninguno de los campos del formulario.

Los controles de validación difieren de los controles de servidor ASP.NET o HTML que ya conocemos ya que su objetivo es aportar funciones de validación para distintos controles, tanto en el lado del cliente como en el lado del servidor, sin necesidad de realizar una doble validación.

Se puede desactivar la validación en el lado del navegador de un control utilizando la propiedad `EnableClientScript` y asignándole el valor `False`.

Para conseguir lo contrario, es decir, para que la validación, ya no de uno sino de todos los controles, se realice sólo en el servidor, se asigna la propiedad `ClientTarget` con el valor "`DownLevel`". Obviamente, por razones de rendimiento no conviene que la validación se realice sólo en el servidor ya que esto provoca muchas idas y vueltas entre navegador y servidor:

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

       ' fuerza que sólo se realice la validación en el servidor

       Me.ClientTarget = "DownLevel"

End Sub
```

Los controles de validación nos permiten generar secuencias de comandos de validación (para el cliente o el servidor) con sólo unas simples selecciones en las propiedades de los controles involucrados.

Los controles de validación nos permiten hacer esta clase de verificaciones:

- Entrada de datos en un campo requerido.
- Entradas en un determinado rango de valores.
- Valores específicos o patrones de caracteres.

Para utilizar estos controles de validación basta con arrastrarlos desde el cuadro de herramientas hasta el formulario de diseño y después vincularlo con el control que va a contener el dato que se quiere validar. Estos controles de validación se mantienen ocultos mientras tanto no se detecten errores de validación.

La validación en el cliente se realiza utilizando código JavaScript en el navegador, que no tendremos que codificar nosotros, lo hará .NET automáticamente.

Cada control con contenido "validable" puede tener vinculados cero, uno o más controles de validación. En cambio, un control de validación sólo se puede vincular con un único control mediante la propiedad `ControlToValidate`.

El control `ValidationSummary` se utiliza para presentar un resumen de todos los errores de un grupo de controles o de todo el formulario.

Cabe señalar que no todos los controles se pueden utilizar en combinación con controles de validación, aunque ese conjunto está formado por los principales controles utilizados en la introducción de texto y datos:

- **HtmlInputText**: propiedad `Value`.
- **HtmltextArea**: propiedad `Value`.
- **TextBox**: propiedad `Text`.
- **HtmlSelect**: propiedad Value.
- **ListBox**: propiedad `SelectedItem.Value`.
- **DropDownList**: propiedad `SelectedItem.Value`.
- **RadioButtonList**: propiedad `SelectedItem.Value`.
- **HtmlInputFile**: propiedad `Value`.

Visual Web Developer se distribuye con los siguientes controles de validación, que se encuentran también en la sección `Validación` del Cuadro de herramientas:

Control	Objetivo
CompareValidator	Compara la entrada del usuario contra otro valor, que puede ser un constante, una propiedad de otro control o un valor de una tabla de una base de datos.
CustomValidator	Verifica la entrada de datos del usuario con una función de validación personalizada.
RangeValidator	Nos asegura que la entrada de datos del usuario se encuentre dentro de un rango de valores especificados.
RegularExpressionValidator	Verifica que la entrada de datos coincide con un modelo definido por una expresión regular.
RequiredFieldValidator	Dato obligatorio. Nos asegura que el usuario no haya dejado el campo en blanco.
ValidationSummary	Muestra los mensajes de error enviados por los controles de validación.

Además del mensaje de error en el navegador, que le sirve al usuario, el desarrollador de la página también puede conocer el estado de la validación comprobando la propiedad `IsValid` del objeto `Page`.

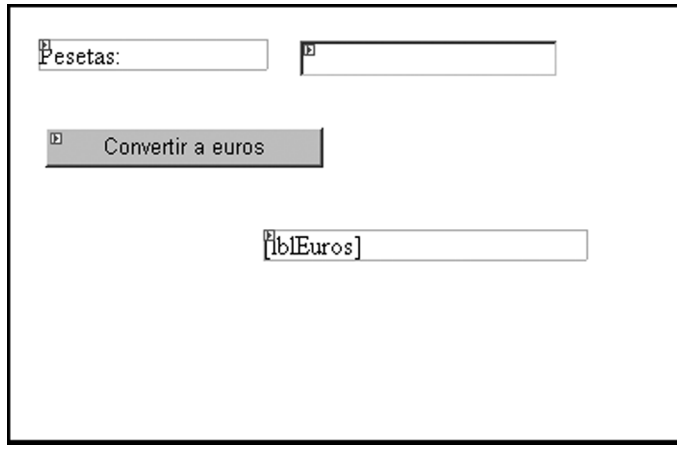
Depuración de aplicaciones Web

La tarea de programación trae consigo la necesidad de depurar; Visual Web Developer incluye una amplia gama de funciones asociadas para la depuración. Una aplicación como `Hola ASPNET` difícilmente precise depuración por lo que definiremos una aplicación simple pero nos brinde la posibilidad de realizar algún tipo de seguimiento. Haremos una aplicación Web de conversión de pesetas a euros para poder analizar las funcionalidades de depuración de Visual Web Developer.

Una aplicación para depurar

La interfaz de la aplicación Web acepta un número como entrada de datos y cuando el usuario hacer clic en el botón Convertir se visualiza el resultado de la conversión a euros.

Ya sabemos cómo crear una aplicación de tipo `Sitio Web`. Ahora crearemos la aplicación 02Conversión con una interfaz como la que se muestra en la siguiente figura.



La aplicación tiene una instrucción muy inocente que se confía en lo que se haya introducido en un cuadro de texto y lo convierte a tipo `Decimal`. El dato para que todo funcione bien debe ser numérico, en caso contrario habría problemas.

El listado del código `Default.aspx` tiene este contenido:

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

<!DOCTYPE  html  PUBLIC  "-//W3C//DTD  XHTML  1.0  Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html   xmlns="http://www.w3.org/1999/xhtml"   >
<head   runat="server">
      <title>Página  sin  título</title>
</head>
<body>
      <form  id="form1"  runat="server">
            <asp:Label  ID="Label1"  runat="server"
                 Style="left:  19px;  position:  absolute;  top:  47px"
                 Text="Pesetas:"  Width="138px">
           </asp:Label>
             <asp:TextBox  ID="txtPesetas"  runat="server"

                 Style="left:  176px;  position:  absolute;
                top:  48px">
           </asp:TextBox>
             <asp:Button  ID="Button1"  runat="server"
                 Style="left:  23px;  position:  absolute;  top:  100px"
                 Text="Convertir  a  euros"  Width="167px"  />
             <asp:Label  ID="lblEuros"  runat="server"
                 Style="left:  185px;  position:  absolute;  top:  164px"
                Width="195px">
           </asp:Label>
    </form>
</body>
</html>
```

Este es el código que da soporte al la funcionalidad de la página, denominado *code-behind*:

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

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

           x  =  Decimal.Parse(txtPesetas.Text)  /  166.386
           lblEuros.Text  =  "Convertidas  a  euros  son..."  &  x.ToString

     End  Sub
End  Class
```

Web.Config

La depuración no está activada de manera predeterminada; en el archivo `web.config` el parámetro `debug` es `false`. Para activar la depuración debe asignarse `true`.

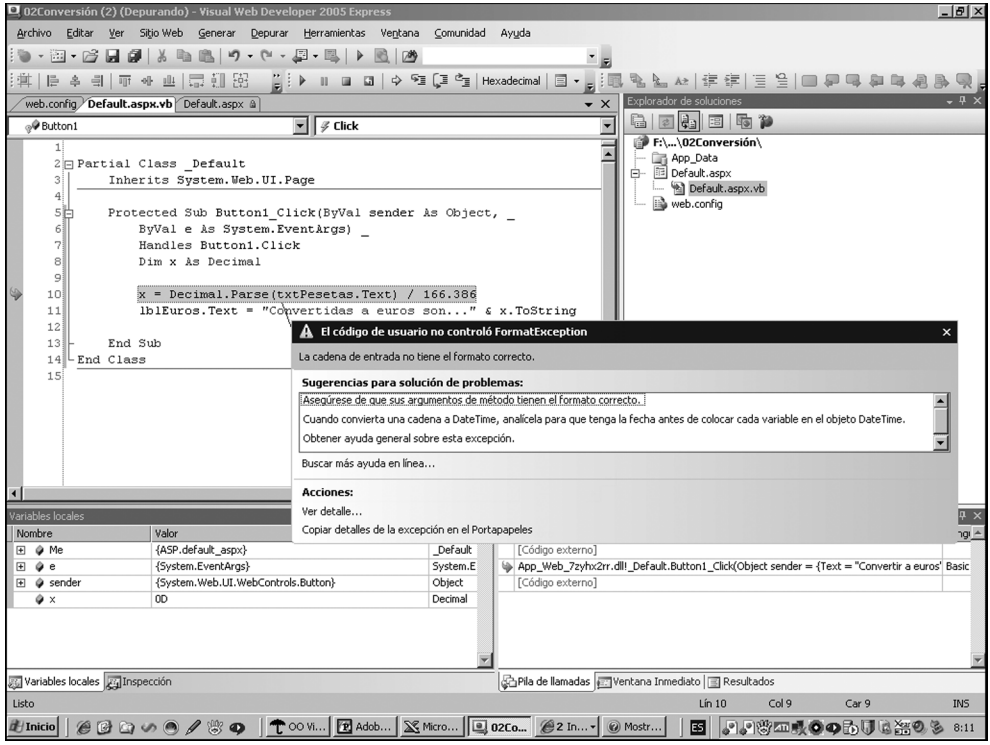
```
...
<configuration       xmlns="http://schemas.microsoft.com/.NetConfiguration/
v2.0">
     <connectionStrings/>
     <system.web>

          <!—
               Establezca debug="true" en la compilación para
               insertar símbolos de depuración en la página
               compilada. Dado que este proceso afecta al
               rendimiento, debe establecer este valor como true
               durante la depuración.

              Opciones de Visual Basic:
               Establezca strict="true" para no permitir las
               conversiones de todos los tipos de datos
               donde se pueda producir una pérdida de datos.
               Establezca explicit="true" para forzar la
               declaración de todas las variables.
          —>

           <compilation   debug="false"   strict="false"   explicit="true"/>
           <pages>
...
```

Ejecución de la aplicación

La aplicación se ejecuta pulsando `F5` o mediante la opción de menú `Depurar/Iniciar depuración`. Si en el cuadro de texto pulsamos siempre valores numéricos la aplicación funcionará correctamente:

¿Pero qué sucede si se introduce un valor no numérico?

Esto provocaría que el programa generase una excepción no controlada (al no haberse incluido un bloque `Try/Catch/End Try`) que hace que Visual Web Developer pase al modo Interrupción (*Break Mode*).

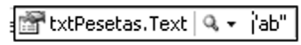


La sentencia que provoca la excepción queda resaltada y se muestra un mensaje de ayuda que sirve para orientarnos sobre la naturaleza del error.

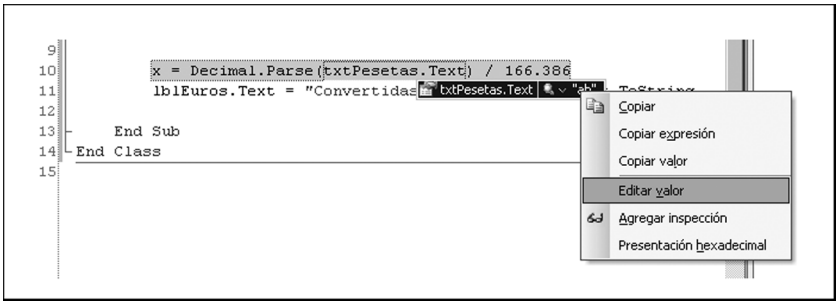
Visualización de las propiedades y variables del programa

Bien, ya sabemos qué instrucción provoca la excepción y también sabemos que debemos revisar el contenido de los datos porque hay algo que no es correcto. ¿Cómo podemos comprobar el valor que tiene cada variable activa del programa? Más específicamente necesitamos saber qué valor almacena la propiedad `Text` del control `TxtPesetas`.

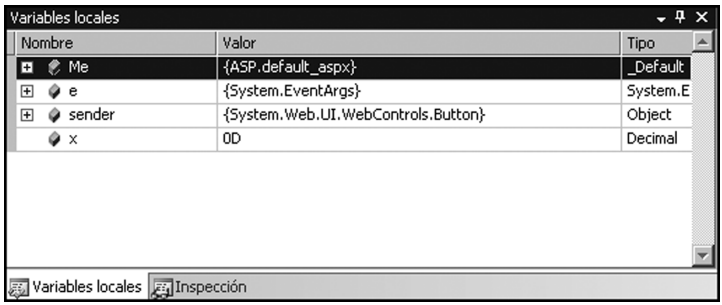
Visual Web Developer incluye una funcionalidad muy práctica que nos permite visualizar los valores de las variables simplemente señalándolas con el ratón cuando el programa está en modo interrupción.



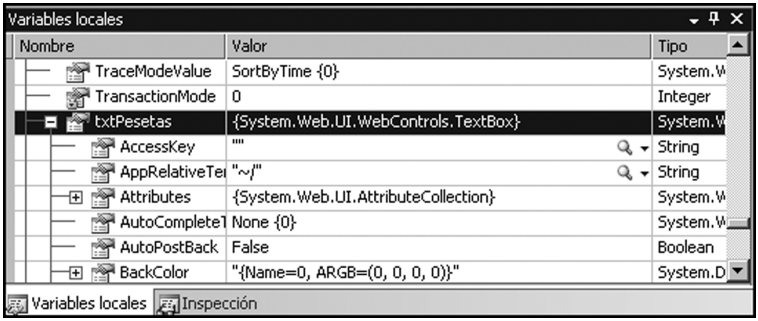
Si hacemos clic con el botón secundario del ratón sobre la microayuda (*DataTip*) aparece un menú con distintas alternativas. Entre las posibilidades que nos da se encuentra la edición de la variable, `Editar valor`; desde esta opción se puede modificar el valor durante la ejecución.



Otro modo de comprobar el valor de las variables es mediante el uso de la ventana `Locales`. En esta ventana podemos visualizar el valor de todas las propiedades de todos los objetos activos de la aplicación. Por ejemplo, el objeto `TxtPesetas` se encuentra en el nodo `Me` y abriendo este nodo, ordenados alfabéticamente, encontraremos todos sus objetos, es decir, todos los controles de la página vigente. A continuación del nodo `Me` aparecen las variables locales del método vigente, en este caso, del método `Button1_Click`.



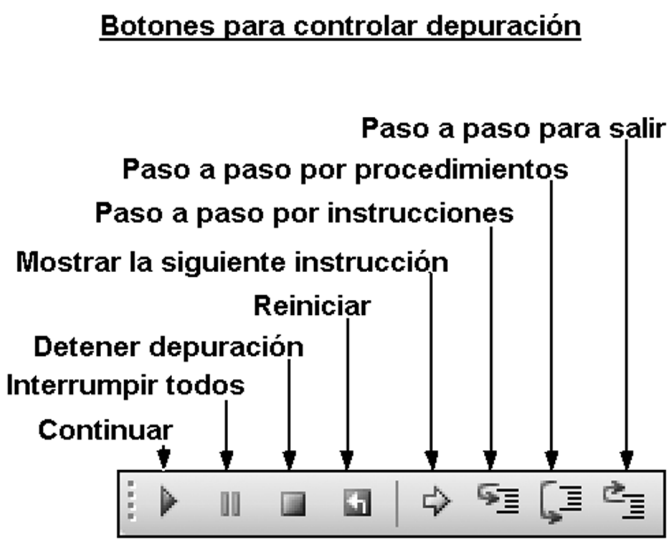
Desde la ventana Locales podemos verificar todas las propiedades de un objeto.



Control de la ejecución

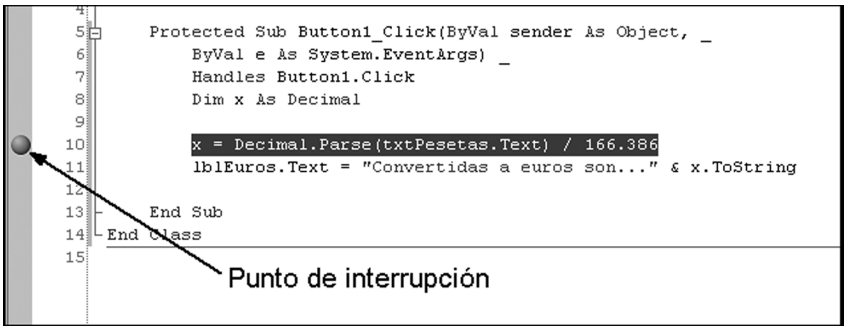
Al iniciar una ejecución en modo depuración tenemos varias opciones para seguir el procesamiento; la barra de herramientas de depuración nos permiten controlar la ejecución del programa:

- **Continuar**: continua la ejecución hasta la siguiente interrupción o hasta el final.
- **Interrumpir todos**: interrumpe la aplicación y establece el modo Interrumpir (Break mode).
- **Detener la depuración**: detiene la aplicación.
- **Reiniciar**: reinicia la aplicación desde el inicio.
- **Mostrar la siguiente instrucción**: resalta la siguiente instrucción que se ejecutará.
- **Paso a paso por instrucciones**: ejecuta la instrucción siguiente y se interrumpe nuevamente el proceso. Si en la instrucción se realiza una llamada a un método, el proceso se detiene antes de ejecutar la primera instrucción del método llamado.
- **Paso a paso por procedimiento**s: ejecuta la instrucción siguiente y se interrumpe nuevamente el proceso. Si en la instrucción se realiza una llamada a un método, el proceso dentro del método se realiza sin interrupción.
- **Paso a paso para salir**: finaliza la ejecución del método vigente y después se interrumpe.



Colocación de puntos de interrupción

Podemos establecer puntos de interrupción en cualquier instrucción ejecutiva del programa haciendo clic en el margen izquierdo de color gris en la ventana de diseño de código. El punto de interrupción queda marcado con un punto grueso.



Para quitar un punto de interrupción se hace clic sobre el punto de interrupción o se pulsa F9. También se pueden quitar todos los puntos de interrupción vigentes con Ctrl+Mayús+F9 o desde la opción de menú Depurar/ Eliminar todos los puntos de interrupción.

Despliegue de aplicaciones ASP.NET

Cuando se llega al final del desarrollo de una aplicación se entra en la fase de implantación o despliegue en la que la aplicación entyra en producción real y abandona el entorno de desarrollo. En este momento es cuando se utilizan las interesantes y prácticas facilidades de despliegue que nos ofrece Visual Web Developer.

Existen tres mecanismos básicos para la implantación de aplicaciones ASP.NET:

- **Xcopy**: es el mecanismo más simple y básico. Se copian los archivos desde nuestro servidor de desarrollo a nuestro servidor de producción. Esto se realiza sencillamente desde el menú `SitioWeb/Copiar Sitio Web`. El único problema, si se puede llamar así, de este enfoque es que la aplicación no se compila en el servidor de producción hasta que no se utiliza por primera vez. Esto se evita utilizando la opción precompilada.
- **Precompilado**: en esta opción lo que se copia son los ensamblados compilados. Se debe compilar toda la aplicación `Generar/Publicar Sitio Web` (opción no disponible en Visual Web Developer) o usar el comando `aspnet_compiler`.
- **Proyecto de instalación**: Visual Studio nos permite crear un proyecto de instalación de sitio Web. Este proyecto se puede instalar en cualquier servidor. Es una opción no disponible en Visual Web Developer.

El proyecto que se instala en el servidor de producción no debe incluir algunos de los parámetros que se utilizan durante el desarrollo y la depuración. Por este motivo deberemos comprobar lo siguiente:

- Eliminar todo cuadro de mensaje que hemos incluido en la aplicación por razones de depuración y prueba.
- Desactivar la depuración en el archivo `web.config`:

```
<compilation   debug="false"   strict="false"   explicit="true"/>
```

- Verificar que en la directiva Page de todas las páginas de la aplicación no queda ningún atributo `trace="True"`.
- Verificar que no se muestran páginas de error detalladas al usuario en producción. Esto implica que el archivo `web.config` incluya esta línea:

```
<customErrors   mode="RemoteOnly"   />
```