



Inside ASP.NET Web Matrix

Alex Homer y Dave Sussman



Análisis en línea en: p2p.wrox.com
para mayores informes sobre libros de Wrox visite: www.wrox.com

Inside ASP.NET Web Matrix

Alex Homer
Dave Sussman

Wrox Press Ltd. ©

© 2002 Wrox Press

Se otorga permiso para redistribuir este documento en su forma completa y original. Todos los derechos reservados

El autor y el editor han realizado todo esfuerzo durante la preparación de este libro para asegurar la precisión del material. Sin embargo, la información contenida en este libro se vende sin garantía, expresa o implícita. Ni los autores, Wrox Press, ni sus distribuidores serán responsables de ningún daño provocado o argumentado que haya sido provocado en forma directa o indirecta por este libro.



Publicado por Wrox Press Ltd,
Arden House, 1102 Warwick Road, Acocks Green,
Birmingham, B27 6BH, Reino Unido

Reconocimientos de las marcas registradas

Wrox se ha esmerado en mantener las convenciones de las marcas registradas para todas las compañías y productos mencionados en este libro, tal como el uso adecuado de mayúsculas. Sin embargo, Wrox no puede garantizar la precisión de esta información.

Créditos

Autores

Alex Homer
Dave Sussman

Editor de la comisión

Daniel Kent

Editor técnico

Daniel Richardson

Editor administrativo

Viv Emery

Coordinador de producción y portada

Natalie O'Donnell

Acerca de los autores

Alex Homer es un experto en computación y desarrollo del Web, con una inmensa pasión por ASP.NET. Aunque debe invertir cierto tiempo realizando trabajo real (un poco de consultoría y capacitación, así como alguna sesión de conferencias), la mayoría de sus días los pasa jugando con la más reciente tecnología Web de Microsoft y escribiendo acerca de ella. Al vivir en la pintoresca soledad de Derbyshire Dales en Inglaterra, está alejado de las demandas del mundo real – sólo con una conexión a Internet para mantener cierta ligera apariencia de normalidad. Pero, ¿qué más se podría querer de la vida?

Puede comunicarse con Alex a través de su propia compañía de software, Stonebroom Limited: alex@stonebroom.com.

Dave Sussman es un *hacker*, en el sentido tradicional de la palabra. Es alguien que gusta de jugar con los códigos y ver cómo funcionan las cosas, razón por la cual pasa gran parte de su vida trabajando con software beta. Afortunadamente, esto coincide con escribir acerca de las nuevas tecnologías, proporcionando información a través de su inglés y gramática deficientes. Vive en un pequeño poblado en la campiña de Oxfordshire. Como muchos programadores en el mundo, posee un costoso equipo de sonido, una TV grande y no tiene vida personal.

Puede comunicarse con Dave a través de su propia compañía, Ipona Limited: davids@ipona.co.uk.

Inside ASP.NET Web Matrix

Durante su vida, relativamente corta, pero espectacularmente exitosa, Microsoft® Active Server Pages (ASP) ha evolucionado desde un sencillo ambiente de *script* para crear páginas Web dinámicas, hasta ser una plataforma poderosa y fácil de usar para el desarrollo de aplicaciones Web completas. En su más reciente versión, ASP.NET, proporciona una solución completa para crear casi cualquier tipo de interfaz interactiva, así como para implementar amplias operaciones de procesamiento de *back-end*.

Sin embargo, a pesar de las muchas funciones poderosas de ASP, nunca fue fácil elegir un ambiente de desarrollo completo y utilizable en el cual crear aplicaciones ASP. Muchos terceros proporcionan soporte a ASP en sus productos, por ejemplo, HomeSite y Macromedia UltraDev (entre otros) soportan ASP 3.0 y, por supuesto, el InterDev incluido en el propio Visual Studio 6.0 de Microsoft, que también estuvo disponible como un producto individual.

Con la llegada de .NET, el soporte para el desarrollo ASP.NET se integró completamente en Visual Studio .NET. Proporciona un ambiente extremadamente poderoso y utilizable para el desarrollo ASP.NET en forma de Web Forms, así como en otros tipos de aplicaciones más tradicionales (Windows Forms). Además, ahora Visual Studio .NET se acompaña con otro producto de Microsoft, llamado **Proyecto Web Matrix ASP.NET de Microsoft** (en lo subsecuente mencionado como "Web Matrix").

Al momento de escribir esto, Web Matrix sólo ha sido liberada como un producto Beta 1. La naturaleza completa del proyecto para Web Matrix ASP.NET de Microsoft consiste en que se desarrollará y crecerá con base en la retroalimentación de la comunidad que la utilice, de tal manera que el conjunto de funciones evolucionará con el tiempo. También debe tener en mente que, al ser éste un producto Beta, hay algunas funciones que aún no están completamente implementadas (por lo que no aparecerán algunas cosas que espera ver).

Sin embargo, incluso en esta etapa Web Matrix es una herramienta extremadamente utilizable y eficiente, la cual, ciertamente, bien vale la pena instalar y experimentar. Con el tiempo, sin duda madurará y se ampliará para ofrecer muchas más de las funciones requeridas para crear sitios Web y aplicaciones Web mediante ASP.NET.

En tres secciones, este documento analizará qué es Web Matrix, qué puede hacer y cómo la puede utilizar:

- ❑ *Parte 1 – ¿Qué es Web Matrix?* consiste en una descripción general de Web Matrix, incluyendo las funciones que proporciona y el IDE que contiene
- ❑ *Parte 2 – Poner Web Matrix a trabajar* lo guía a través del uso de Web Matrix para crear una aplicación que contenga muchos diferentes tipos de páginas y recursos
- ❑ *Parte 3 – Configurar y ampliar Web Matrix* demuestra cómo se puede configurar Web Matrix para adecuarla a sus requerimientos individuales y cómo se puede ampliar instalando sus propios complementos, o los de terceros

Parte 1 – ¿Qué es Web Matrix?

A partir de las primeras impresiones, tal vez piense que Web Matrix es sólo un ambiente de desarrollo simplificado para crear aplicaciones ASP.NET. De hecho, proporciona mucho más que esto. Al igual que las páginas ASP.NET (incluyendo páginas de dispositivos móviles), Web Matrix se puede utilizar para crear controles de usuario y archivos de clase (para compilar en ensambles), archivos de servicio web, e incluso Manejadores http. También proporciona soporte integrado para crear y editar páginas HTML, hojas de estilo, esquemas y documentos XML, archivos de texto y *scripts* SQL, así como archivos de configuración .NET (tales como `Web.config` y `global.asax`).

Web Matrix también proporciona poderosos asistentes que automatizan gran parte del proceso para crear páginas que manejan datos, páginas que utilizan memoria caché de salida y páginas que utilizan las funciones de autenticación integradas de ASP.NET. También se presenta en forma completa, con su propio servidor web y otros complementos útiles. Incluso puede crear e instalar sus propios complementos, si así lo desea.

¿Por qué utilizar Web Matrix en lugar de Visual Studio .NET?

Antes de ver detalladamente Web Matrix, vale la pena analizar las diferencias entre ésta y Visual Studio .NET. Después de todo, ¿por qué proporciona Microsoft dos diferentes ambientes de desarrollo para ASP.NET? La respuesta es que ambos se complementan – están dirigidos a diferentes tipos de desarrollo.

Visual Studio .NET es un excelente ambiente para un desarrollo en equipo el cual, integrado con un sistema de control de archivos fuente, tal como Visual SourceSafe, proporciona la seguridad y administración consistente de archivos de proyecto necesarias cuando un equipo de personas trabajan en un proyecto.

Una gran diferencia entre Web Matrix y Visual Studio .NET es que esta última insiste en crear proyectos ASP.NET utilizando la técnica de código detrás, en lugar de código en línea. Se utilizan muchos desarrolladores ASP tradicionales para incluir el contenido de la presentación (tal como HTML, texto, etc.) en el mismo archivo, a medida que el código ASP crea y maneja el contenido de la interfaz dinámica. Decidir si esta idea es buena depende de cómo usted (y su equipo) desarrollen realmente las aplicaciones. Si utilizan diseñadores gráficos para crear las partes visuales de las páginas, y después utilizan otros programadores orientados más técnicamente para crear el código, tal vez prefieran contar con archivos separados para estas dos secciones de la interfaz.

Sin embargo, tal vez prefiera incluir tanto el contenido de código como el visual en la misma página en línea, quizá para evitar la complejidad adicional de tener que compilar el archivo de código detrás y luego heredar de él en la página de la interfaz visual (aunque Visual Studio .NET hace esto por usted). Desarrollar de esta manera, hasta que apareció Web Matrix, significaba regresar al enfoque anterior a ASP.NET, que utilizaba un editor sencillo de texto (tal como Notepad) o alguna otra herramienta de terceros.

En resumen, las diferencias entre Web Matrix y Visual Studio .NET son:

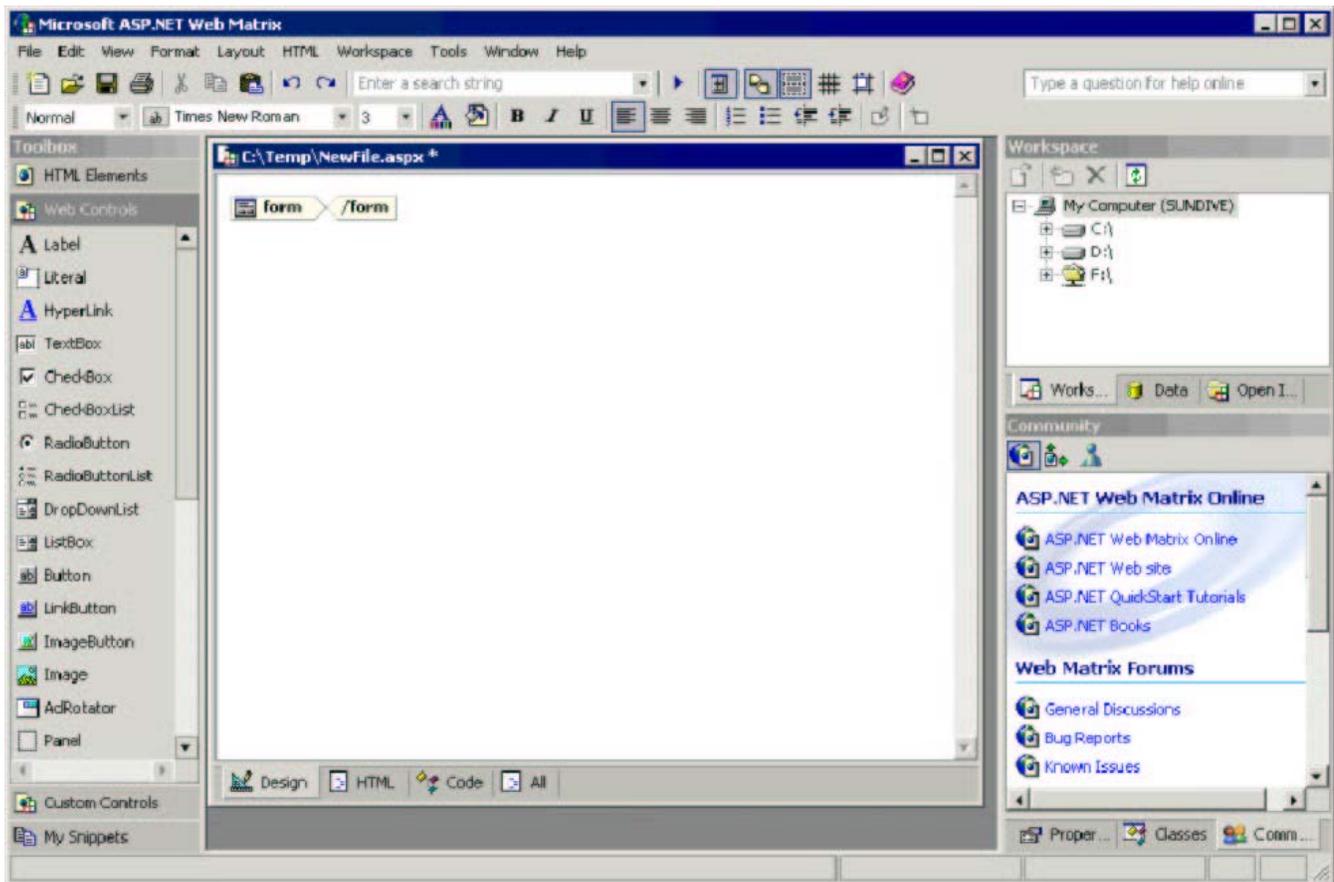
- ❑ **Soluciones basadas en proyectos** – Visual Studio .NET tiene el concepto de un **proyecto**, al cual se pueden agregar varios tipos de archivos y recursos. Web Matrix no utiliza un enfoque basado en proyectos; en cambio, trata cada archivo como un elemento separado.
- ❑ **Estructura de la página ASP.NET** – Web Matrix crea páginas ASP.NET usando el enfoque en línea, en lugar del enfoque de código detrás de Visual Studio .NET.

- ❑ **Interfaz** – Web Matrix es ligera (el archivo de instalación tiene un tamaño aproximado de 1MB), delgada y rápida. Sin embargo, no proporciona todo el conjunto de aspectos positivos de la interfaz que se incluyen en Visual Studio .NET. Por ejemplo, Web Matrix no ofrece terminación de declaraciones, listas de miembros de objetos, o sugerencias instantáneas en la ventana de editar.
- ❑ **Compilación de archivos de clase** – A diferencia de Visual Studio .NET, Web Matrix no compila automáticamente los archivos de clase en ensambles. Esto se debe hacer desde la línea de comando.
- ❑ **Archivos de ayuda de .NET Framework** – Web Matrix no incluye documentación de referencia para .NET Framework. En cambio, proporciona una lista útil y plegable, basada en carpetas, de las clases comúnmente utilizadas y de sus miembros, junto con una lista completa de todos los demás espacios de nombre y clases dentro de la Biblioteca de clases de .NET Framework. Web Matrix también contiene un explorador de clases, que muestra los miembros individuales de cualquier clase y ofrece un vínculo hacia el SDK de .NET local (si está instalado) y las páginas de referencia de .NET de MSDN en línea.
- ❑ **Comunidad** – Web Matrix está diseñada para ser una herramienta de la comunidad y contiene diversos tipos de vínculos al sitio de la comunidad en línea en <http://asp.net/WebMatrix/>, así como vínculos a los grupos de noticias, servidores de listas y otros sitios que proporcionan soporte a la comunidad para Web Matrix.
- ❑ **Costo** – ¡Web Matrix es gratuita!

En palabras del equipo de desarrollo de Web Matrix en Microsoft, Web Matrix es "un producto virtual diseñado para soportar un enfoque de la comunidad para el desarrollo de ASP.NET, al tiempo que también es divertido de usar". Tiene como objetivo enfocarse en aquellas tareas que cumplen el 80% de los requerimientos para crear aplicaciones ASP.NET.

Una guía básica hacia el IDE de Web Matrix

El IDE de Web Matrix se diseñó para ofrecer una interfaz familiar a los desarrolladores que han usado Visual Studio .NET. Incluye las barras de menú y las barras de herramientas usuales en la parte superior, una Caja de herramientas en la parte inferior izquierda, varias "ventanas de proyectos" en la parte inferior derecha y una barra de estado a lo largo de la parte inferior. La siguiente pantalla muestra el diseño predeterminado para una nueva página ASP.NET:

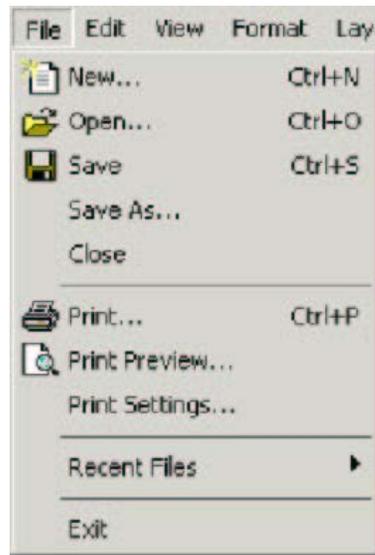


En la versión actual, la Caja de Herramientas y las ventanas de proyectos no son movibles, aunque se pueden redimensionar y la Caja de herramientas se puede ocultar. El área principal de trabajo es una interfaz de documentos múltiples (MDI), por lo que puede abrir varios archivos al mismo tiempo para edición. En las siguientes secciones, veremos detalladamente cada sección del IDE.

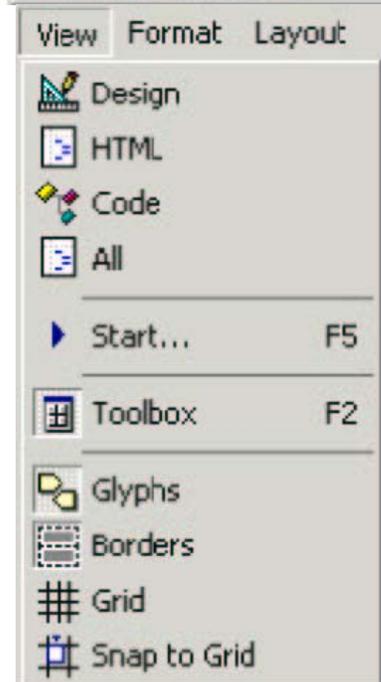
Menús y barras de herramientas

Muchas de las entradas e iconos del menú en la barra de herramientas son elementos familiares para cualquier usuario de Windows. El menú File contiene comandos para crear, abrir, guardar, cerrar, imprimir y ver previamente los archivos. El menú Edit contiene comandos que le permiten deshacer o rehacer sus ediciones recientes, así como cortar, copiar, pegar y seleccionar texto, además de las opciones usuales de Find y Replace, con un comando para ir a una línea específica en la página.

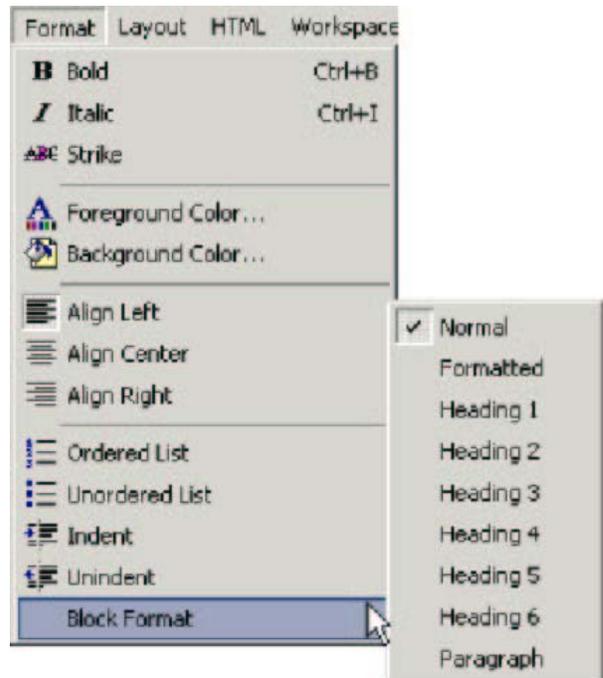
El menú **Edit** también contiene comandos para agregar complementos de código (algo que veremos cuando examinemos la Caja de herramientas más detalladamente) y comandos para editar el elemento seleccionado en el momento de estar en la vista **Design**, o editar las plantillas para un control `DataList`. Asimismo, veremos más sobre estas funciones más adelante. El grupo final de comandos en este menú le permite formatear páginas y comentar o eliminar los comentarios al texto o los controles seleccionados en ese momento, ya sea en la vista **HTML** o **Code** (ambas se cubrirán con mayor detalle cuando analicemos la ventana de edición más adelante).



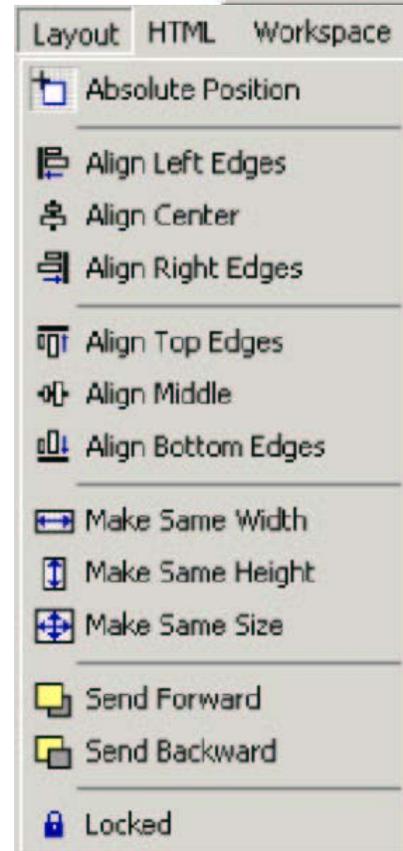
El menú **View** se puede utilizar para seleccionar una vista específica de la página o archivo actual (dependiendo del tipo de archivo). En la pantalla anterior del mismo IDE se pueden ver unas pestañas en la parte inferior de la ventana principal de edición, que corresponden a las cuatro vistas: **Design**, **HTML**, **Code** y **All** (las cuales se cubren más detalladamente cuando veamos la ventana de edición más adelante). El menú **View** también incluye comandos para iniciar la página ASP.NET actual que se está ejecutando en su explorador predeterminado, mostrar u ocultar la Caja de herramientas, cambiar la vista de **glyphs** (tal como los elementos `form` y `/form` visibles en la pantalla anterior del IDE de Web Matrix), mostrar u ocultar los bordes de los elementos, mostrar u ocultar la rejilla de edición y especificar si la función **Snap-to-Grid** está activada o desactivada.



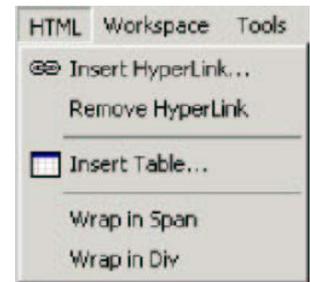
El menú Format le permite especificar la apariencia de los elementos en sus páginas ASP.NET y HTML, utilizando las combinaciones normales de formateo, es decir, negrillas, itálicas y tachado. También puede configurar los colores de frente y fondo, controlar la alineación horizontal, formatear los elementos, tal como una lista en orden o en desorden y ensamblarlos. El formato que se especifique aquí se agrega a la declaración de los elementos HTML como una mezcla de elementos de formateo estándar (, <i>, , etc.). La opción final abre un submenú de opciones de formateo en bloque, que agregan el importante elemento de "Heading" desde <h1> hasta <h6>.



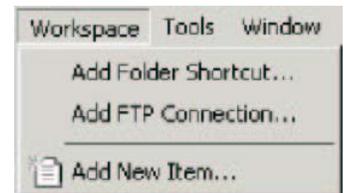
Otros elementos del menú en la parte superior menos familiares son Layout y HTML. El menú Layout contiene un comando para cambiar los controles seleccionados al modo Absolute Position. Web Matrix agrega luego a los elementos un selector de estilo CSS `position: absolute`, con los selectores de tamaño y posición apropiados. Luego, utilizando los comandos de este menú puede alinear los elementos en diversas formas, convertirlos a igual ancho, altura o tamaño, controlar su orden z, así como asegurar los elementos, para que no se muevan accidentalmente.



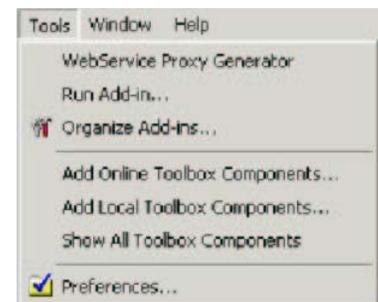
El menú HTML proporciona funciones para agregar hipervínculos a un elemento (puede especificar el URL y la descripción) y eliminarlos de un elemento. También le permite insertar una tabla HTML de tamaño fijo, en donde puede especificar todos los tipos de atributos para ésta, incluyendo el número de filas y columnas, los bordes y los colores, el espaciado de la celda y el *padding* de la misma. Los comandos del menú HTML también se pueden utilizar para envolver los elementos actualmente seleccionados en un elemento `` o `<div>`.



El menú Workspace se utiliza para manipular la ventana Workspace, que es uno de los elementos que aparecen al lado derecho del IDE (en la sección a la cual nos referimos como "ventanas de proyectos"). Le permite agregar un acceso directo a una carpeta correlacionada en la red, o establecer una conexión FTP a otro servidor. También se puede utilizar para agregar otros elementos al Workspace.

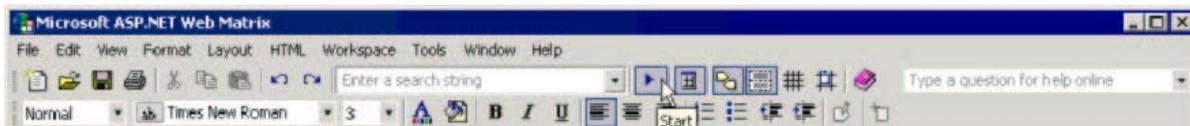


El menú Tools contiene comandos para ejecutar y manejar los diversos complementos que se presentan con Web Matrix, así como los que usted desarrolle y aquellos proporcionados por terceros. También puede utilizarlo para agregar componentes (ya sea almacenados en su máquina local o descargados del sitio de la comunidad ASP.NET) a la Caja de herramientas, que aparece en el lado izquierdo del IDE, o reconfigurar la Caja de herramientas para que muestre la lista predeterminada de componentes. El comando final de este menú abre el cuadro de diálogo Preferences, donde puede cambiar la manera en que trabaja Web Matrix para ajustarla a sus propios requerimientos (veremos esto más detalladamente al final de este documento).



Los dos últimos menús, que no aparecen aquí, son Windows y Help. El menú Windows contiene los comandos usuales Cascade, Tile y Close All, así como una lista de las ventanas abiertas, para que pueda intercambiar rápidamente entre ellas. El menú Help contiene los comandos usuales para trabajar con el archivo de ayuda local, así como vínculos a sitios web relacionados, tal como la documentación .NET en línea y los tutoriales ASP.NET QuickStart en MSDN. El menú Help también contiene un vínculo que se puede usar para enviar retroalimentación al equipo de Web Matrix en Microsoft, así como comandos que muestran información sobre la propia Web Matrix.

La mayoría de los comandos del menú comúnmente utilizados también están disponibles desde las dos barras de herramientas que aparecen de manera predeterminada en Web Matrix. La siguiente pantalla muestra estas barras de herramientas, con el comando Start (también disponible desde el menú View) que aparece seleccionado. A la izquierda de este icono aparece un cuadro de texto, en el cual puede registrar una cadena para buscar en el archivo actual (presione *Intro* para iniciar el proceso de "encontrar") y a la derecha aparecen los iconos para articular la presentación del Toolbox, los glifos en la ventana de edición, los bordes de los elementos y la rejilla de edición, así como un icono para activar o desactivar "Snap-to-Grid". La segunda barra de herramientas contiene un conjunto de comandos de formateo comunes. Como es usual, cada icono en la barra de herramientas muestra una sugerencia descriptiva de la herramienta cuando coloca el *mouse* sobre éste:



El extremo derecho de la barra de herramientas en la parte superior contiene una lista desplegable de cuadro combinado en la cual puede escribir una pregunta (o palabras clave importantes) y obtener ayuda directamente del sitio web ASP.NET en <http://www.asp.net/>. Esta ayuda se entrega en forma de una lista de artículos y recursos relacionados con su consulta (de nuevo, presione *Intro* para iniciar el proceso de "buscar").

La ventana Edit

La mayor parte del trabajo que realizará en Web Matrix involucra una instancia de la **ventana editar** (cada archivo que abre aparece en una ventana de editar por separado). Como es de esperar, la ventana editar muestra los contenidos de un archivo, tales como una página ASP.NET, un archivo de clase, un Servicio web, etc. Dependiendo del tipo de archivo que se carga en la ventana, usted puede seleccionar una de entre cuatro vistas de ese archivo:

- Design – que muestra la apariencia visible de la página (con glifos o sin ellos)
- HTML – que muestra el contenido real de HTML y texto de la página, pero no muestra las secciones de código
- Code – que muestra sólo el código en la página, sin HTML ni otro contenido
- All – que (como pensaría) muestra la página completa, incluyendo las directivas de la página y las secciones de código en línea

Para una página ASP.NET o un Control de usuario, recibe el conjunto completo de opciones (pestañas) que aparece en la siguiente pantalla. Para una página HTML, sólo cuenta con Design y HTML. Para todos los demás tipos de archivos, no existen pestañas y la única vista disponible consiste en el contenido del archivo como texto.



Los elementos que aparecen en la Caja de herramientas cambian, dependiendo de la vista (analizaremos esto más adelante, cuando veamos la Caja de herramientas con mayor detalle). En las vistas Design, HTML y All puede arrastrar los controles de la Caja de herramientas hacia la página y la HTML apropiada se insertará en el punto en donde suelte el control. En la vista Code, puede arrastrar los **desarrolladores de código** hacia la página. Estos son "mini-asistentes" que crean automáticamente el código para lograr tareas específicas. Web Matrix incluye un conjunto de desarrolladores de código extremadamente útiles, que crean métodos de acceso a datos utilizando instrucciones SQL, así como otro desarrollador de código que elabora el código para enviar un mensaje de correo electrónico.

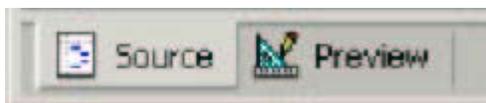
Conforme edita un archivo en cualquiera de las vistas, los cambios se reflejan también en las otras vistas. Así pues, puede arrastrar un elemento hacia una página ASP.NET en la vista Design y configurar su apariencia visible; cambiar a la vista HTML para agregarle atributos y cambiar a la vista Code para añadirle un manejador de eventos. Después, al cambiar a la vista All aparecerá la página completa, tal como la vería en un editor de texto normal, con la directiva Page, bloques de `<script>`, HTML y todos los demás contenidos visibles.

Veremos más detalladamente cómo se utiliza la ventana editar en la *Parte 2*, cuando empecemos a crear páginas ASP.NET y otros tipos de archivos.

Usar el modo Preview

Es posible modificar el comportamiento de la ventana editar de Web Matrix, cambiando al modo Preview. En este modo, Web Matrix no intenta interpretar y mostrar los controles y elementos HTML en una página en la vista Design (en donde normalmente aparecen como glifos). Este modo es útil si usted tiene contenido que no se creó dentro de Web Matrix, o contenido que no desea que Web Matrix interprete y quizá reformatee. Preview Mode se selecciona a través del cuadro de diálogo Preferences (como veremos más adelante) y da como resultado sólo dos vistas (pestañas), que están disponibles en la ventana editar:

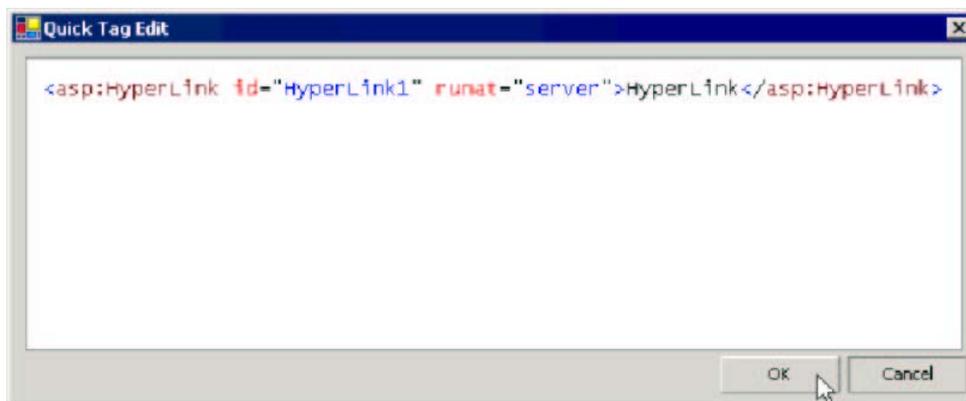
- ❑ Source – que muestra la página completa, incluyendo las directrices de la página y las secciones de código en línea (igual que en la vista All, cuando está en el modo editar predeterminado).
- ❑ Preview – que muestra una vista previa del contenido HTML y de texto de la página, pero sin intentar interpretar y reformatearla. En apariencia se asemeja a la vista Design, pero sin los glifos y sin cambiar la fuente de la página.



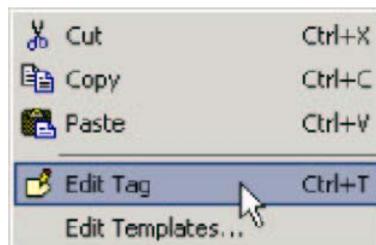
Preview Mode sólo afecta a las páginas ASP.NET y a los Controles de usuario (para los cuales las dos vistas se llaman Source y Preview) y las páginas HTML (para las cuales las dos vistas se llaman HTML y Preview).

Quick Tag Edit

Así como puede editar los contenidos de una página ASP.NET o los Controles de usuario directamente en la ventana Edit, usted puede utilizar la función Edit Tag, en el menú Edit, para cambiar rápidamente los atributos o los contenidos de un elemento cuando la página se muestra en la vista Design. Esto abre una ventana de diálogo que muestra la declaración HTML del elemento actualmente seleccionado en la página, que podrá entonces editar según se requiera:



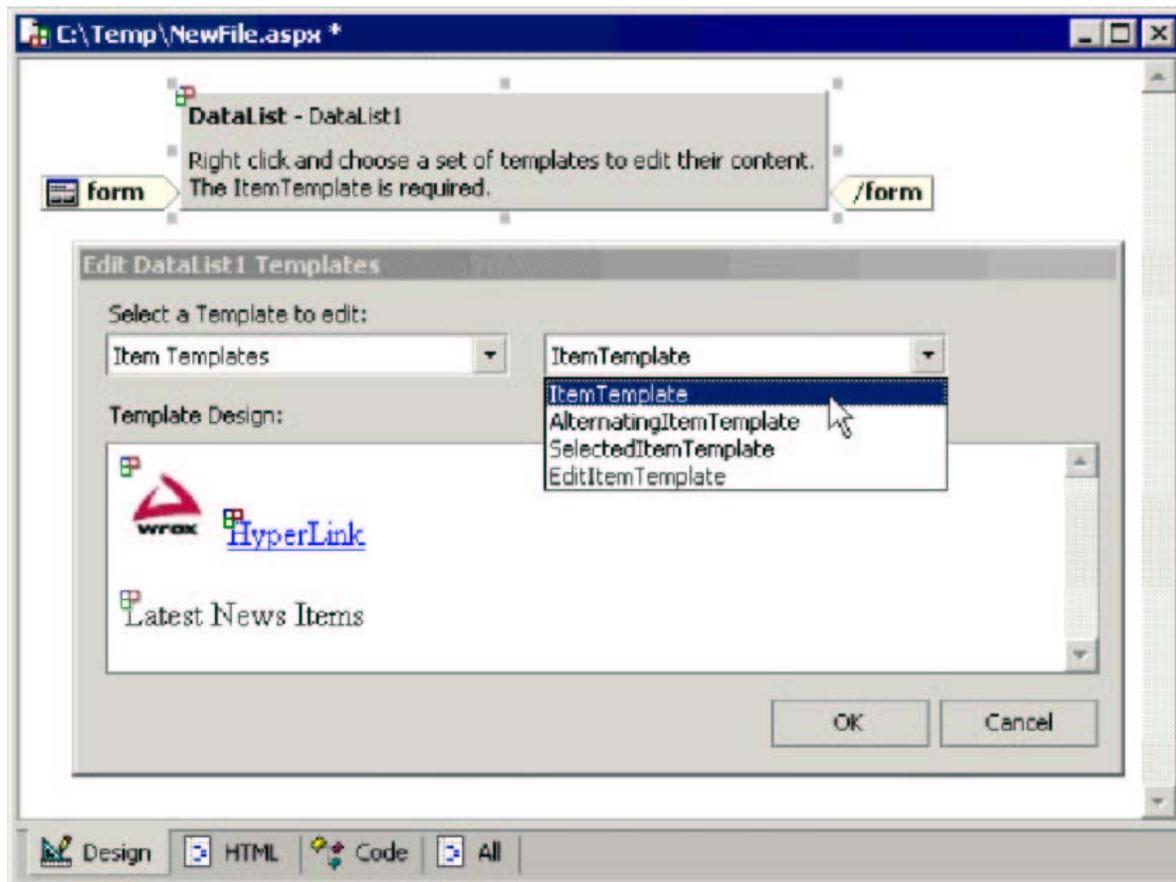
La ventana Quick Tag Edit también se puede abrir al hacer clic con el botón alterno sobre un elemento en la página de la vista Design, y seleccionar Edit Tag en el menú de contexto que aparece. Este menú también contiene los comandos Cut, Copy y Paste.



Editar plantillas de Listas de datos

Cuando el control actualmente seleccionado en la vista Design (para una página ASP.NET o un Control de usuario) es una `DataList`, puede editar las plantillas para dicho control mediante el comando `Edit Templates` en el menú `Edit`. De otra forma, puede hacer clic con el botón derecho en el control `DataList` dentro de la página en la vista `Design` para abrir el menú de contexto que analizamos en la sección anterior y seleccionar `Edit Templates`. Ambas acciones abren el cuadro de diálogo `Edit DataList Templates`.

Las plantillas definen la apariencia de las diversas secciones de la producción del control durante el tiempo de ejecución, y se dividen en tres grupos: `Header and Footer Templates`, `Item Templates` y `Separator Template`. A través de las dos listas desplegables puede especificar la plantilla que desea editar y después crear el contenido para la plantilla dentro del cuadro de diálogo:



La sección `Template Design` del cuadro de diálogo es una "superficie de diseñador", justo como la ventana `Edit`, en la vista `Design`. Usted arrastra y suelta los controles sobre esta superficie para crear la apariencia, exactamente de la misma forma en que lo haría en la ventana `Edit`.

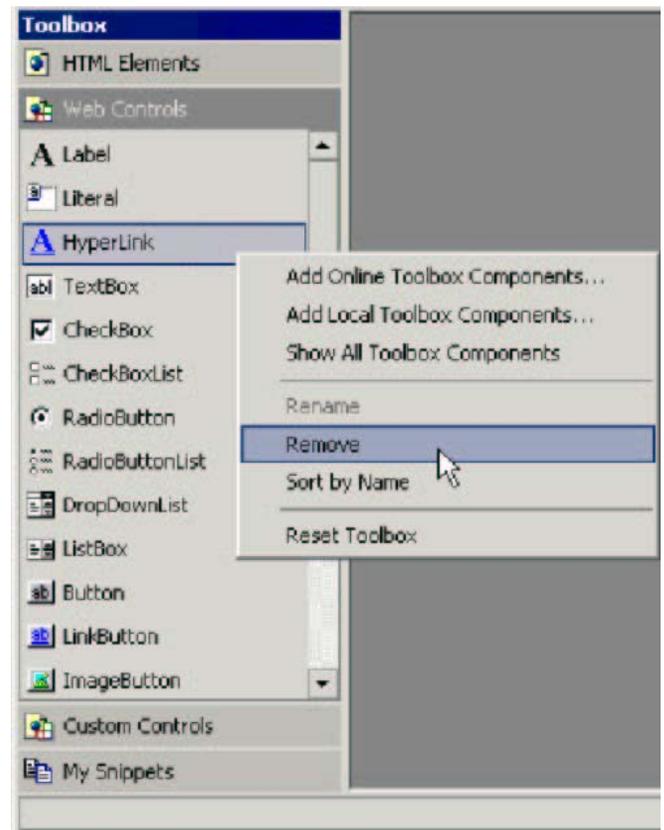
Las plantillas para un control `DataGrid` se editan utilizando la ventana `Properties`, y no a través del comando `Edit Templates`. Esto se explicará más adelante en este documento. Las plantillas para el control `Repeater` se deben definir manualmente dentro de la ventana `Edit`, en la vista `HTML`.

La Caja de herramientas

La Caja de herramientas o Toolkit de Web Matrix se ve y funciona en gran medida como la de Visual Studio y otras herramientas de terceros. Cuando una página ASP.NET o una página de Control de usuario se abre en la vista Design, en la vista HTML, o en la vista All dentro de la ventana editar, la Toolbox muestra los diversos elementos y controles que se pueden agregar a la página. Los controles disponibles se dividen en tres secciones: HTML Elements (controles de servidor del espacio de nombre `System.Web.UI.HtmlControls`); Web Controls (controles de servidor del espacio de nombre `System.Web.UI.WebControls`) y Custom Controls (un área en donde puede agregar sus propios controles). Para colocar cualquiera de estos controles en la página actual, simplemente los arrastra de la Toolbox y los suelta en la posición requerida en la página.

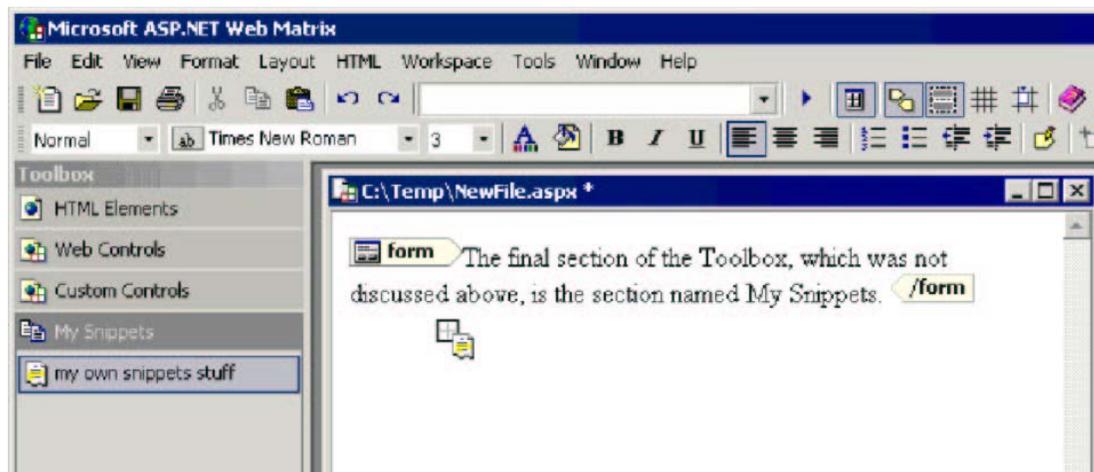
Cuando la ventana de edición actual está en la vista Code, o en la vista All, la Toolbox contiene una sección llamada Code Builders (mencionada brevemente antes). La sección Code Builders contiene pequeños Asistentes que se pueden utilizar para automatizar la creación de bloques específicos de código en la página actual. Para iniciar un Asistente de Code Builders, simplemente lo arrastra de la Toolbox hacia la página. En caso de requerir información adicional, el Asistente la solicitará.

Para manejar los contenidos de la Toolbox haga clic con el botón alterno sobre ésta (o sobre uno de los controles que contiene) y aparecerá un menú de contexto que proporciona opciones para agregar, eliminar y renombrar los controles – así como opciones para ordenar los controles por nombre y reconfigurar la Toolbox, de manera que muestre sus contenidos predeterminados.



Complementos de la Caja de herramientas

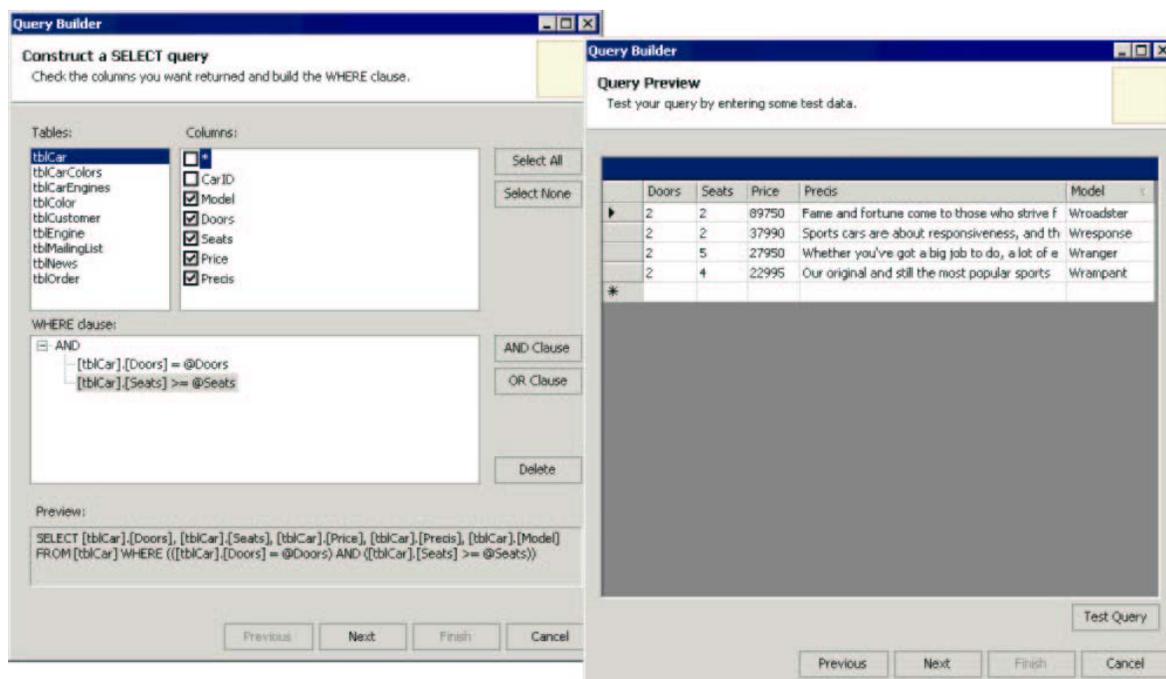
La última sección de la Caja de herramientas se conoce como My Snippets (para continuar con la tradición de Microsoft de crear nombres que suenen agradables). Puede crear un complemento al seleccionar cierto texto o código en la ventana editar y luego simplemente arrastrarlo hasta la Toolbox. Parte de la primera línea del texto aparecerá en la Toolbox (aunque usted puede renombrarla, como se muestra en la siguiente pantalla) y el complemento completo aparece en una sugerencia de herramienta, al colocar sobre éste el señalizador del mouse. Después, este complemento se puede insertar en cualquier otra página o archivo, al arrastrarlo desde la Toolbox:



Si hace clic con el botón alterno sobre la sección My Snippets de la Toolbox, o sobre un complemento existente, aparece un menú de contexto que le permite renombrar o eliminar los complementos. También puede exportar los complementos a un archivo XML, así como importar los complementos como XML. Esta es una forma extremadamente rápida y útil de compartir código con otros – incluyendo la comunidad de Web Matrix ASP.NET. Más adelante encontrará información acerca de esta comunidad.

Desarrolladores de código de la Caja de herramientas

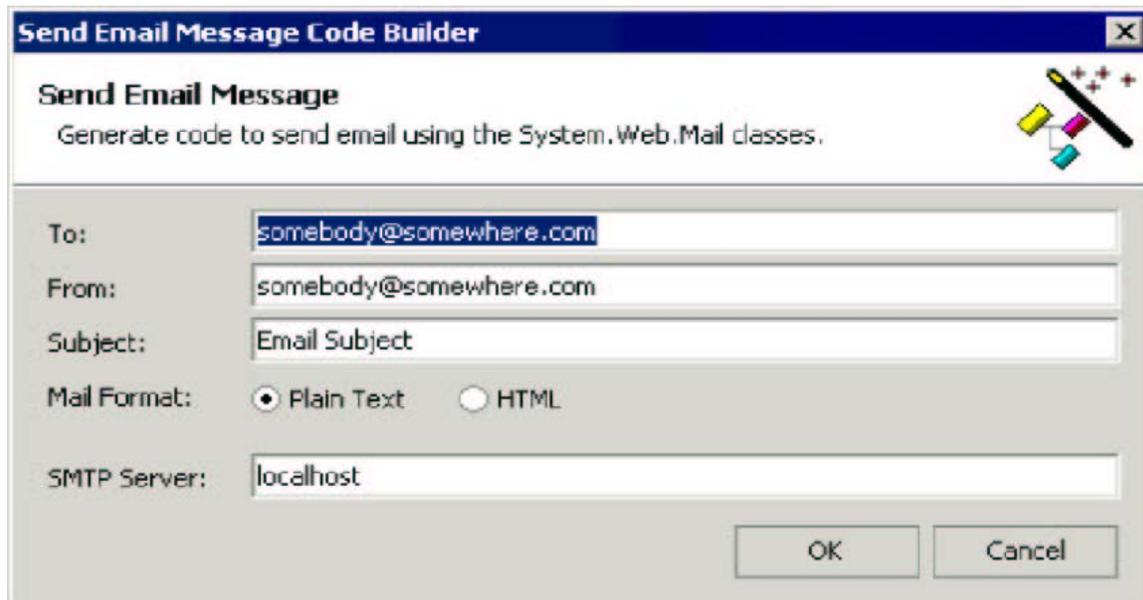
Veamos un ejemplo de los Desarrolladores de código (Code Builders) proporcionados con Web Matrix. La siguiente pantalla muestra los resultados de arrastrar el Desarrollador de código SELECT Data Method hasta una página ASP.NET. Después de mostrar el cuadro de diálogo en el cual usted especifica a cuál base de datos conectarse, el Query Builder abre un cuadro de diálogo en el que puede crear la instrucción SQL requerida. Cuando esta declaración SQL está completa, el Query Builder muestra un cuadro de diálogo en donde usted podrá revisar la instrucción, al registrar los valores para los parámetros de la cláusula WHERE y luego ver los resultados:



Cuando usted esté satisfecho con la declaración SQL, el Query Builder le pide que especifique el nombre para el método de acceso a datos que creará, y que defina si el método utilizará un DataSet o un DataReader para acceder al almacén de datos. Si elige un DataSet se crea un código que se adjunta a las líneas siguientes. El código se inserta automáticamente en la página actual. Observe que la cláusula WHERE contiene argumentos de tipo seguro. Esto se logra mediante parámetros escritos en la función:

```
Function MyQueryMethod(ByVal doors As Short, ByVal seats As Short) As System.Data.DataSet
    Dim connectionString As String = "server=localhost'; user id='userid'; " _
        & "password='password'; Database='WroxCars'"
    Dim sqlConnection As System.Data.SqlClient.SqlConnection _
        = New System.Data.SqlClient.SqlConnection(connectionString)
    Dim queryString As String = "SELECT [tblCar].[Doors], [tblCar].[Seats], " _
        & "[tblCar].[Price], " _
        & "[tblCar].[Model] " _
        & "FROM [tblCar] WHERE (([tblCar].[Doors] = @Doors) " _
        & "AND ([tblCar].[Seats] >= @Seats))"
    Dim sqlCommand As System.Data.SqlClient.SqlCommand _
        = New System.Data.SqlClient.SqlCommand(queryString, sqlConnection)
    sqlCommand.Parameters.Add("@Doors", System.Data.SqlDbType.SmallInt).Value = doors
    sqlCommand.Parameters.Add("@Seats", System.Data.SqlDbType.SmallInt).Value = seats
    Dim dataAdapter As System.Data.SqlClient.SqlDataAdapter _
        = New System.Data.SqlClient.SqlDataAdapter(sqlCommand)
    Dim dataSet As System.Data.DataSet = New System.Data.DataSet
    dataAdapter.Fill(dataSet)
    Return dataSet
End Function
```

El elemento Desarrollador de código Send Email también se puede arrastrar hasta una página, en cuyo caso aparece el cuadro de diálogo mostrado en la siguiente pantalla. En este cuadro de diálogo usted puede especificar las direcciones To y From, el Subject, el formato de correo y el Servidor SMTP que se utilizará para enviar el mensaje:



Entonces se crea el siguiente código. Lo único que resta por hacer es establecer el texto del mensaje, utilizando la propiedad `mailMessage.Body`:

```
' Build a MailMessage
Dim mailMessage As System.Web.Mail.MailMessage = New System.Web.Mail.MailMessage
mailMessage.From = "somebody@somewhere.com"
mailMessage.To = "somebody@somewhere.com"
mailMessage.Subject = "Email Subject"
mailMessage.BodyFormat = System.Web.Mail.MailFormat.Text

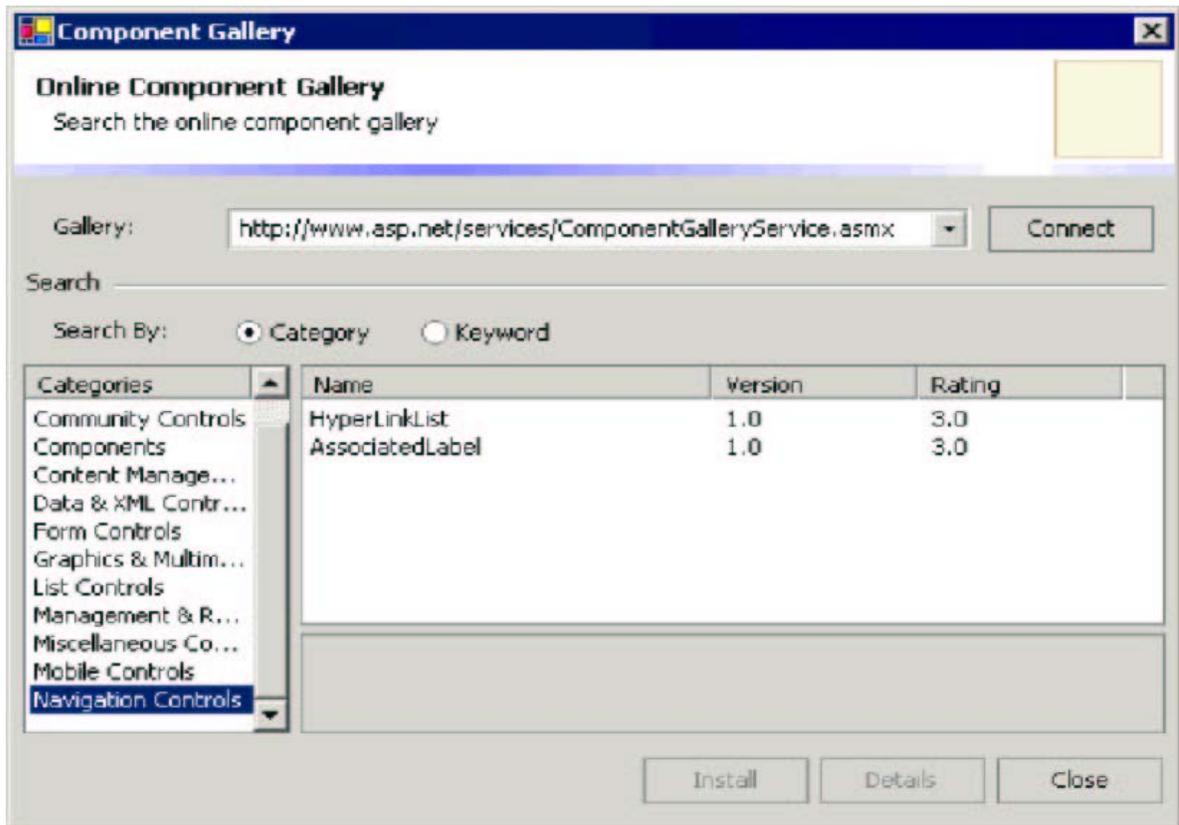
' TODO: Set the mailMessage.Body property

System.Web.Mail.SmtpMail.SmtpServer = "localhost"
System.Web.Mail.SmtpMail.Send(mailMessage)
```

Agregar controles a la Caja de herramientas

Usted puede agregar cualquier tipo de control .NET a la Toolbox – los controles del Kit de herramientas de Microsoft Mobile Internet (MIT), los Controles Web de Internet Explorer, o los controles que instala desde el sitio Web de la comunidad de Web Matrix. Para instalar los controles dentro de la sección Custom Controls de la Toolbox, seleccione Add Online Toolbox Components o Add Local Toolbox Components, desde el menú principal Tools, o haga clic con el botón alterno sobre la Toolbox misma.

Si selecciona Add Online Toolbox Components, Web Matrix se conecta al sitio Web de la comunidad y muestra los controles que están disponibles para su descarga. En la siguiente pantalla, hemos seleccionado la sección de HTML Controls, en donde puede ver que contiene los Controles Web de IE (observe que el rango de controles de Online Component Gallery que se muestra aquí, se ampliará y cambiará regularmente con el tiempo):

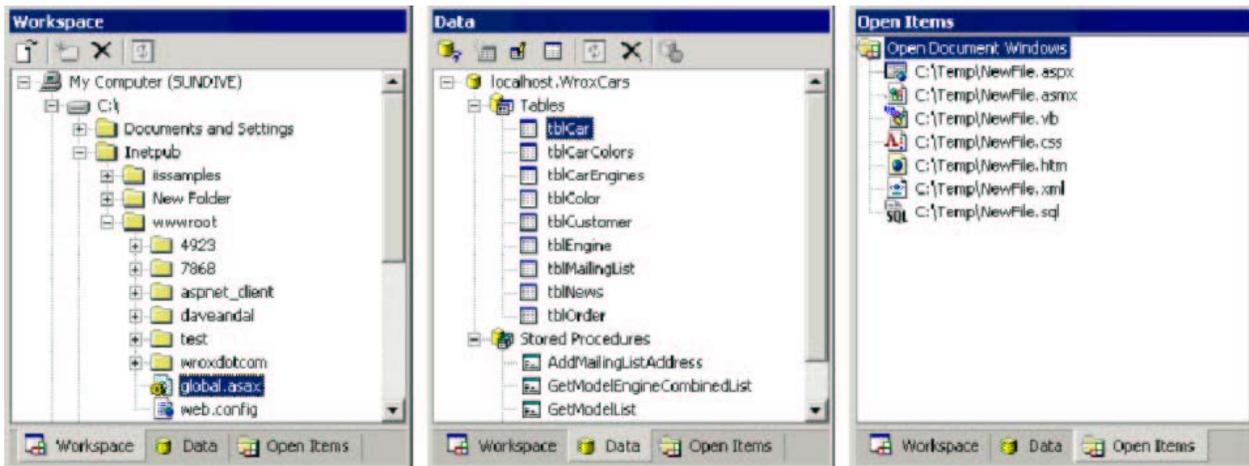


Si selecciona Add Local Toolbox Components, se abre un cuadro de diálogo que le permite seleccionar un ensamble desde la carpeta `%WINDOWS%\Microsoft.Net\Framework\version\` en su sistema predeterminado, o desde cualquier otra carpeta que contenga un DLL de la Biblioteca de componentes adecuado.

La ventana del "Proyecto" (mitad superior)

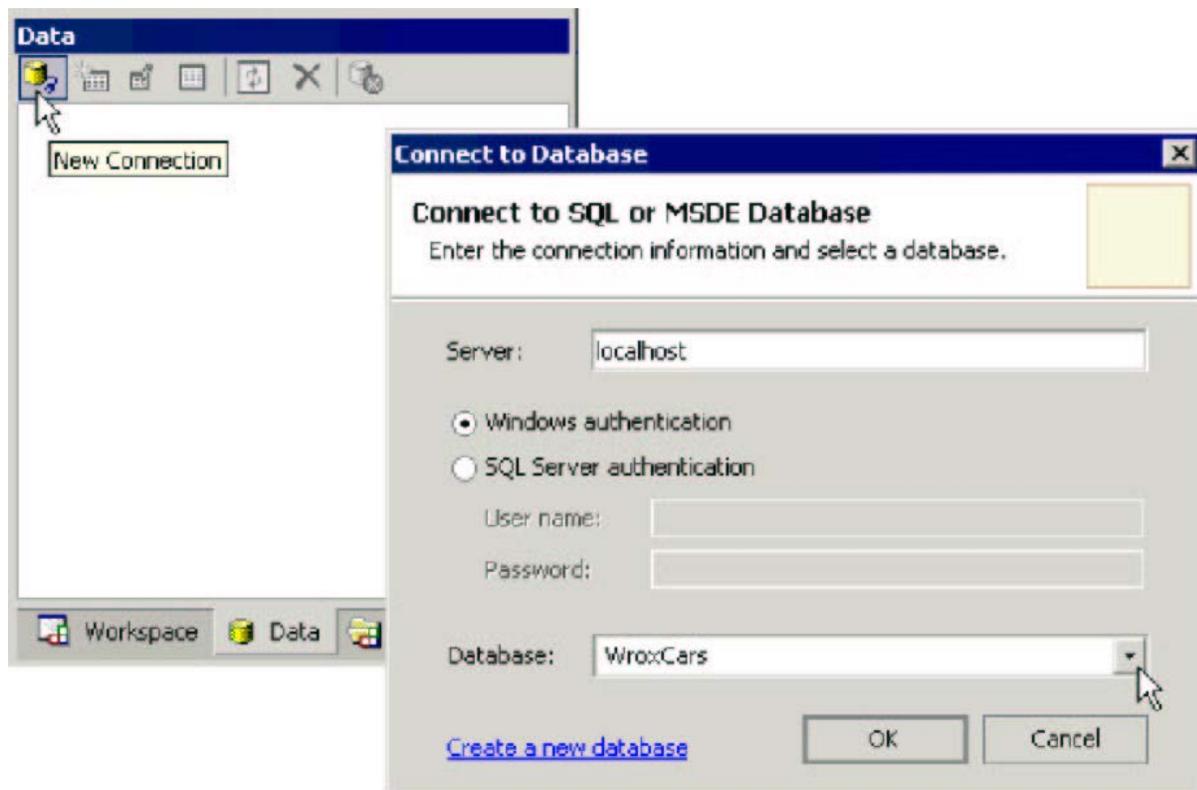
En el extremo inferior derecho de la pantalla, aparecen dos ventanas que forman conjuntamente lo que llamamos aquí "ventana del proyecto". La mitad superior proporciona tres opciones:

- **Workspace** – que muestra los controladores y archivos en su máquina local y, por predeterminación, cualesquiera carpetas correlacionadas. Asimismo, usted puede agregar accesos directos a otras carpetas locales o de red, así como a otros servidores a través de FTP, usando los comandos en el menú **Workspace**. Al hacer doble clic en un archivo aquí, éste se abre en una nueva ventana de edición. Al hacer clic con el botón alterno en la ventana **Workspace** puede agregar un acceso directo, crear una nueva carpeta, o crear un nuevo archivo con base en uno de los tipos estándar disponibles en el cuadro de diálogo **New File**. Más adelante veremos los tipos de archivo disponibles en *Tipos de archivos y asistentes*.
- **Data** – que le permiten conectarse a un SQL Server o a una base de datos MSDE, para poder ver, crear y editar tablas y procedimientos almacenados dentro de esa base de datos. La siguiente pantalla muestra una conexión a una base de datos llamada **WroxCars**, en donde aparecen desplegadas las tablas que contiene dicha base de datos.
- **Open Items** – que muestra las ventanas de edición que se abren actualmente dentro del IDE. Simplemente usted puede hacer clic sobre una de ellas y traerla al frente para editar.



Usar la ventana Data

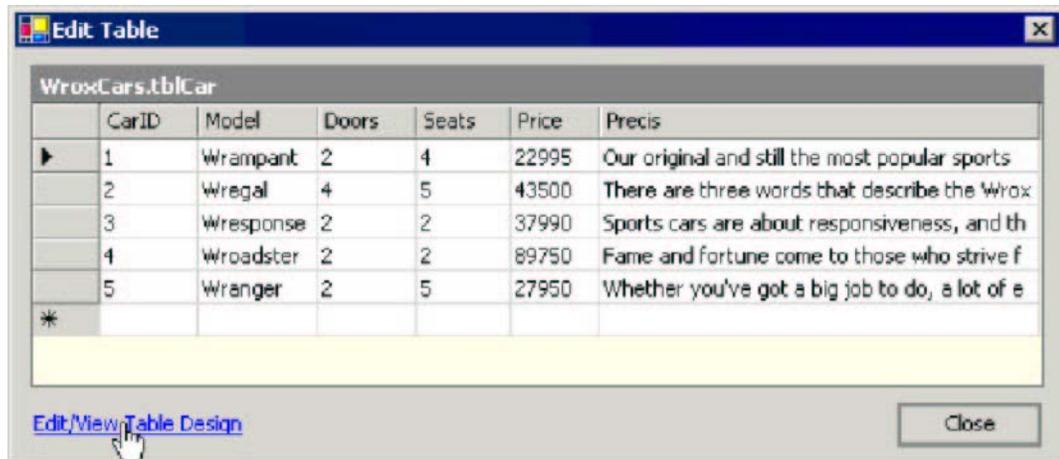
Para conectarse a una fuente de datos, haga clic en el icono en la parte superior de la ventana Data, como se muestra en la siguiente pantalla, y registre los detalles de conexión para la base de datos. Generalmente puede utilizar la autenticación de Windows para el SQL Server local o para la base de datos MSDE, o si lo prefiere, puede especificar el nombre del usuario y la contraseña para autenticación SQL Server. Después, seleccione la base de datos en la lista desplegable:



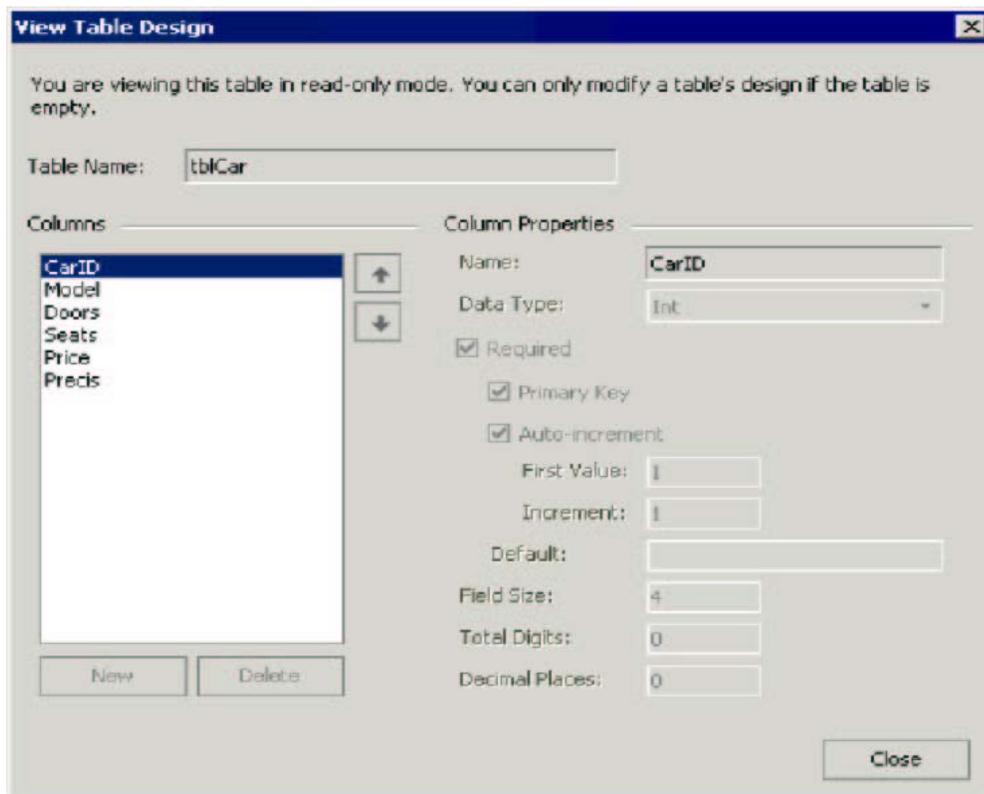
También existe una opción para crear una nueva base de datos en el servidor especificado. Todo lo que debe hacer es registrar el nombre para la nueva base de datos en el cuadro de diálogo que aparece, y la base de datos se crea y se muestra en la ventana Data.

Trabajar con una fuente de datos

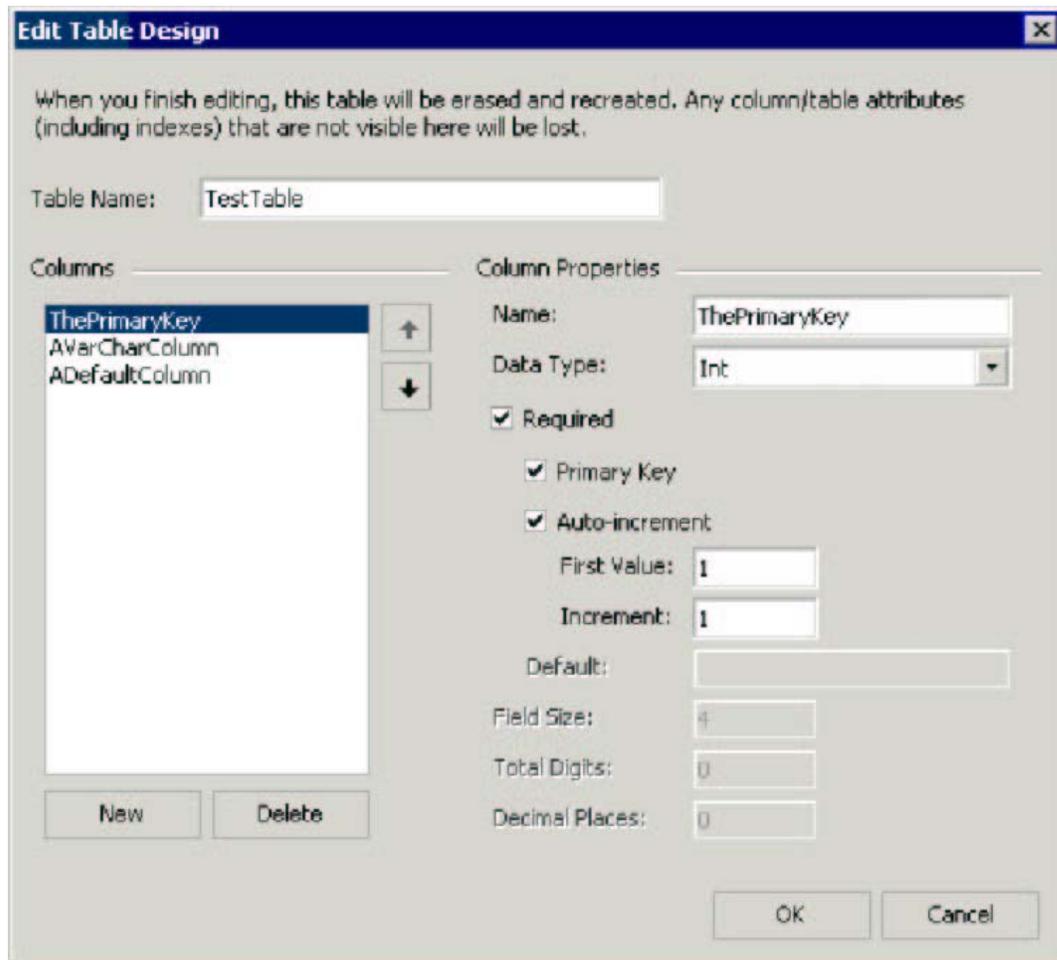
Una vez que tiene por lo menos una base de datos abierta en la ventana Data (si lo desea, se puede conectar a más de una a la vez), se habilitan los iconos restantes en la parte superior de esta ventana. Puede utilizar estos iconos para trabajar con la base de datos – puede crear una nueva tabla o un procedimiento almacenado (mediante el segundo icono), o editar los existentes (mediante el tercer icono). Por ejemplo, en la siguiente pantalla, seleccionamos nuestra tabla tblCars y hacemos clic en el icono Edit. Puede ver los contenidos de la tabla, así como editar directamente los valores de las filas. No obstante, observe que esto funciona sólo si la tabla tiene una clave primaria definida:



Además de editar los contenidos de la tabla, también puede ver la estructura de la tabla haciendo clic en el vínculo que aparece en la parte inferior del cuadro de diálogo mostrado en la pantalla anterior:

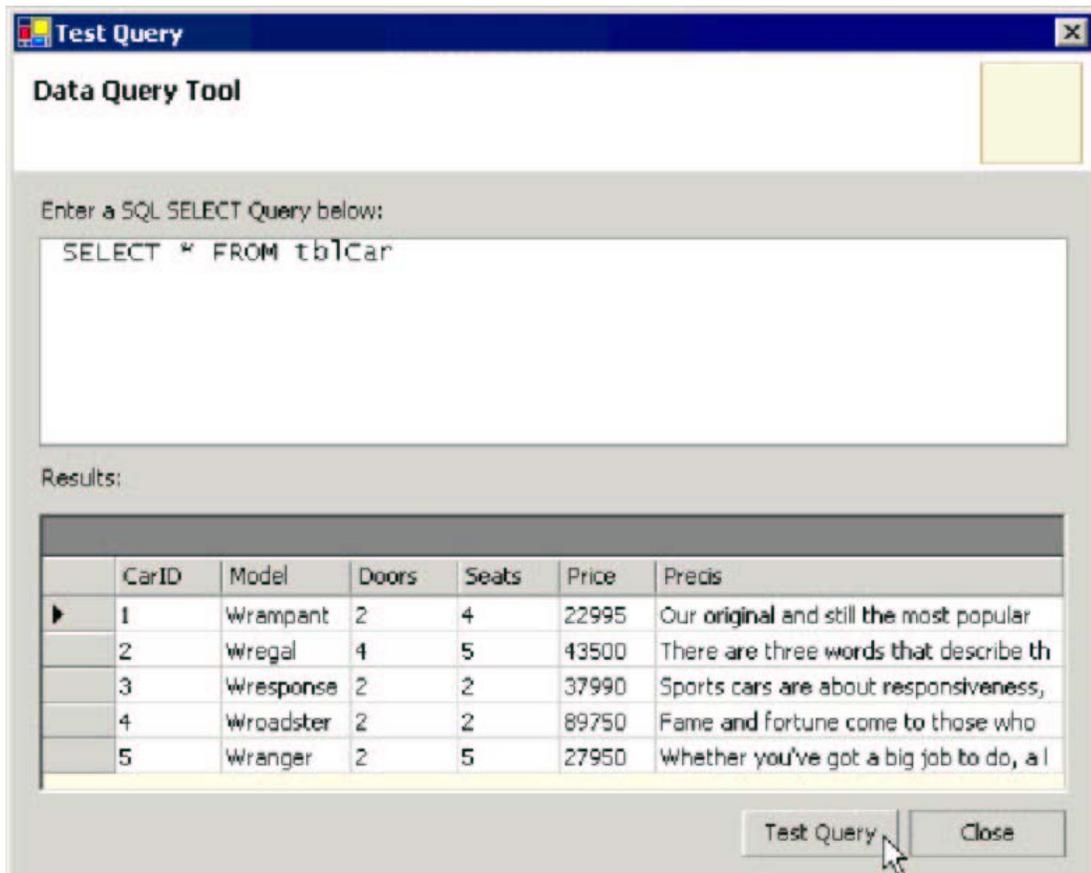


Sin embargo, sólo puede editar la estructura de la tabla si ésta se encuentra vacía. La siguiente pantalla muestra una tabla nueva, creada al hacer clic en el segundo icono (New Item) en la ventana de Data. Se le han agregado tres columnas, configurando las propiedades mediante los controles de la sección Column Properties del cuadro de diálogo:



Consultar una fuente de datos

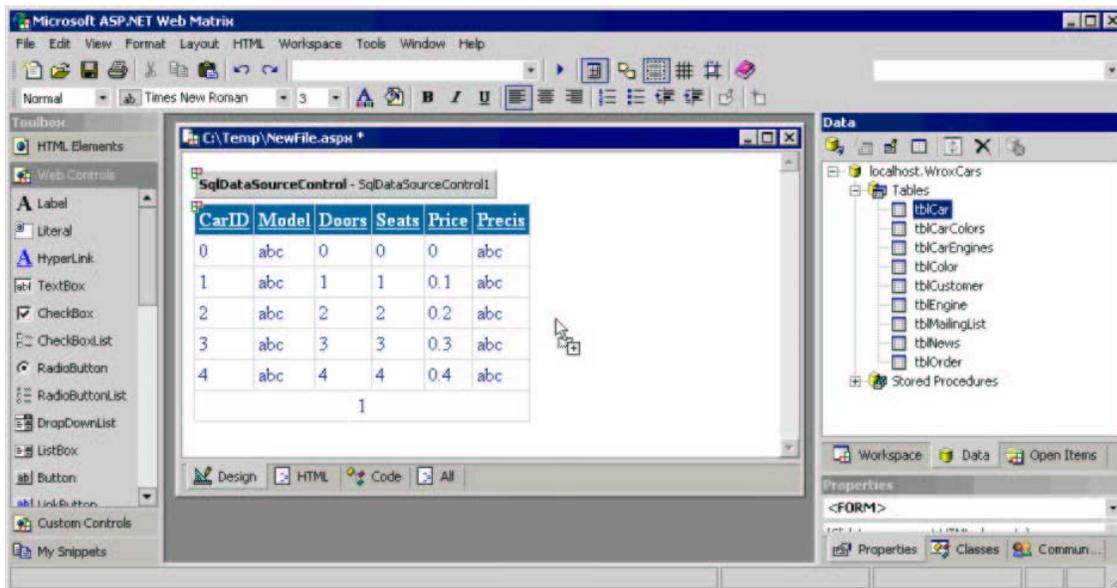
El cuarto icono en la ventana Data se utiliza para consultar una fuente de datos. Si primero selecciona una tabla existente en la ventana Data, Web Matrix crea una declaración SQL simple, la cual consultará y mostrará todos los contenidos de dicha tabla. Por supuesto, puede editar la declaración SQL para realizar consultas más específicas si lo requiere:



Los otros tres iconos en la parte superior de la ventana Data se pueden usar para actualizar los contenidos de la ventana, eliminar una tabla o un procedimiento almacenado de una base de datos (o incluso eliminar una base de datos completa), así como desconectar una base de datos de la ventana Data. Esto significa que la ventana Data le proporciona un ambiente completo para trabajar con las bases de datos, lo cual es especialmente útil si está conectando a MSDE (que no proporciona interfaz, a diferencia de la versión completa de SQL Server). ¡Sólo tenga cuidado de no seleccionar el icono Delete cuando sólo quiera desconectar!

Tablas de datos, controles de datos y controles de rejillas

El acceso a los datos es un elemento fundamental de muchos sitios Web y aplicaciones Web, por lo cual Web Matrix facilita las tareas comunes que interactúan con una base de datos. Por ejemplo, usted puede arrastrar una tabla de la lista que aparece en la ventana Data y soltarla sobre una página – en donde Web Matrix crea automáticamente una página basada en DataGrid, que muestra los datos de esa tabla:



Si cambia a la vista Code después, puede ver el código que Web Matrix creó dentro de la página (la siguiente es una versión sin puente).

```
<wmx:SqlDataSourceControl id="SqlDataSourceControl1" runat="server"
  UpdateCommand="UPDATE [tblCar] SET [Model]=@Model, [Doors]=@Doors, [Seats]=@Seats,
  [Price]=@Price, [Precis]=@Precis WHERE [CarID]=@CarID"
  SelectCommand="SELECT * FROM [tblCar]" AutoGenerateUpdateCommand="False"
  ConnectionString="server='localhost'; trusted_connection=true; Database='WroxCars'"
  DeleteCommand="" >
</wmx:SqlDataSourceControl>
<wmx:MxDataGrid id="MxDataGrid1" runat="server"
  DataSourceControlID="SqlDataSourceControl1"
  BorderColor="#CCCCCC" AllowSorting="True" AutoGenerateFields="False"
  DataMember="tblCar"
  AllowPaging="True" BackColor="White" CellPadding="3" DataKeyField="CarID"
  BorderWidth="1px" BorderStyle="None">
  <PagerStyle horizontalalign="Center"
  forecolor="#000066" backcolor="White"
  mode="NumericPages"></PagerStyle>
  <FooterStyle forecolor="#000066" backcolor="White"></FooterStyle>
  <SelectedItemStyle font-bold="True" forecolor="White"
  backcolor="#669999"></SelectedItemStyle>
  <ItemStyle forecolor="#000066"></ItemStyle>
  <Fields>
  <wmx:BoundField DataField="CarID" SortExpression="CarID"
  HeaderText="CarID"></wmx:BoundField>
  <wmx:BoundField DataField="Model" SortExpression="Model"
  HeaderText="Model"></wmx:BoundField>
  <wmx:BoundField DataField="Doors" SortExpression="Doors"
  HeaderText="Doors"></wmx:BoundField>
```

```

<wmx:BoundField DataField="Seats" SortExpression="Seats"
  HeaderText="Seats"></wmx:BoundField>
<wmx:BoundField DataField="Price" SortExpression="Price"
  HeaderText="Price"></wmx:BoundField>
<wmx:BoundField DataField="Precis" SortExpression="Precis"
  HeaderText="Precis"></wmx:BoundField>
</Fields>
<HeaderStyle font-bold="True" forecolor="White" backcolor="#006699"></HeaderStyle>
</wmx:MxDataGrid>

```

El código utiliza varios nuevos controles del servidor, diseñados especialmente para ser usados en Web Matrix - los elementos `MxDataGrid` y `BoundField` asociado, que crean la parte visible de la pantalla, así como `SqlDataSourceControl`, que se conecta a la fuente de datos y expone los mismos. Analizaremos esto detalladamente más adelante.

La ventana del "Proyecto" (mitad inferior)

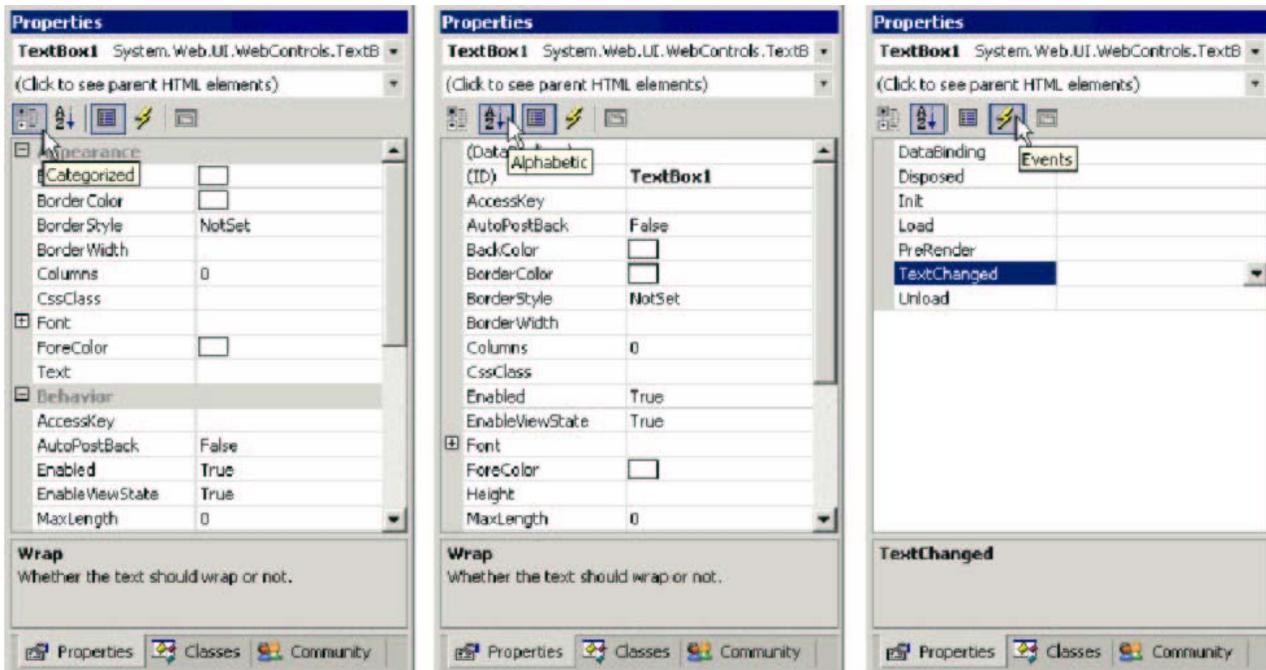
La sección inferior de la ventana del "proyecto", proporciona también tres funciones principales:

- ❑ **Properties (y Events)** – que le permite ver y editar las propiedades del elemento actualmente seleccionado en la ventana de edición. Para una página ASP.NET o un Control de usuario en la vista **Design**, usted puede seleccionar un elemento (o incluso la página misma seleccionando el elemento `<body>`) y acceder a sus propiedades. En todas las demás vistas, la ventana **Properties** muestra las propiedades relacionadas con ASP.NET para el documento mismo. Para todos los demás tipos de archivos, la ventana **Properties** simplemente muestra la ruta hacia el documento.
- ❑ **Classes** – que proporciona una lista de todos los espacios de nombre de .NET Framework (los cuales aparecen ya sea por tipo o por nombre) utilizados comúnmente en los proyectos ASP.NET. Cada espacio de nombre se puede ampliar para mostrar las clases que contiene, junto con la clase de base y las interfaces que implementa. Al hacer doble clic en cualquier entrada de esta lista se abre el **Class Browser** de Web Matrix, que muestra los detalles de ese elemento.
- ❑ **Community** – que proporciona vínculos a recursos útiles, grupos de noticias, servidores de listas y otros sitios que proporcionan soporte y materiales de referencia para Web Matrix.

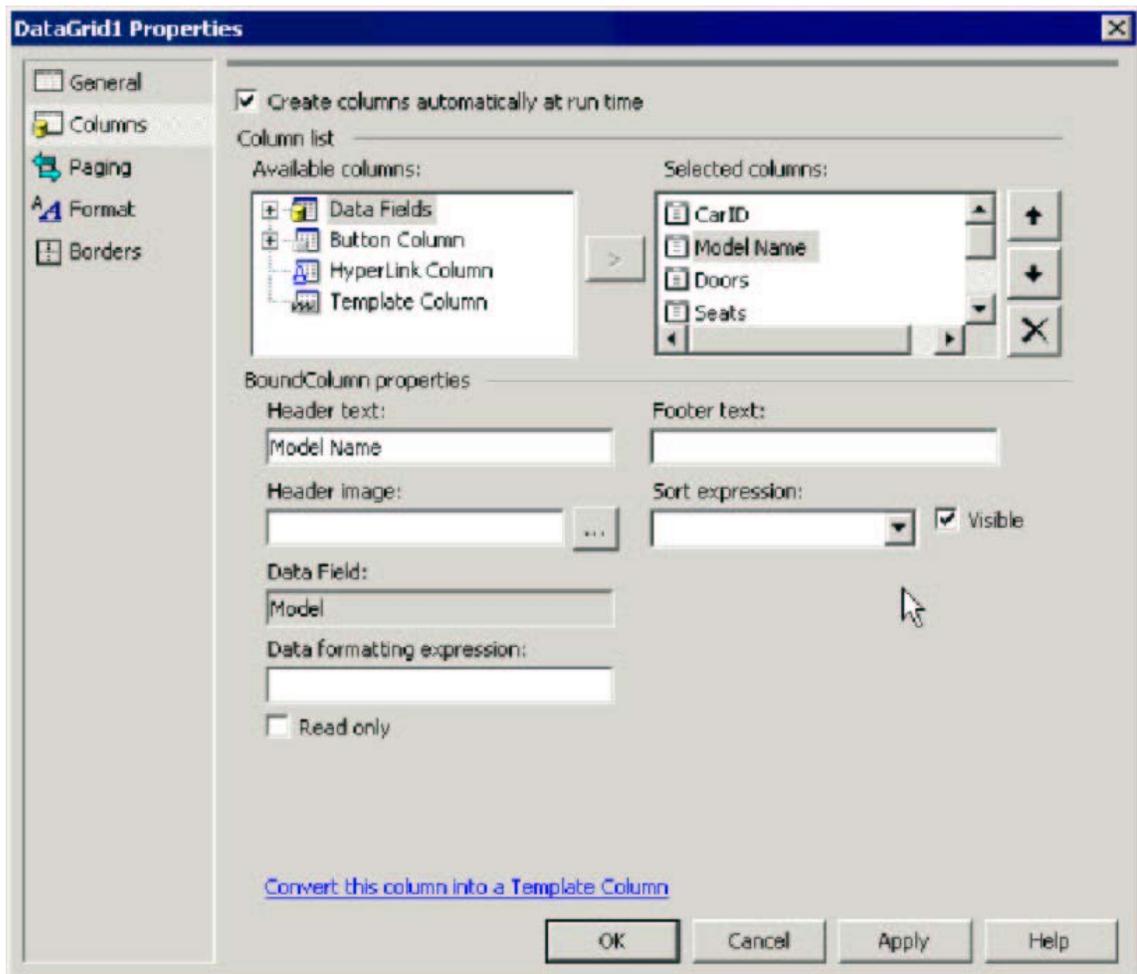
La ventana Properties

La ventana **Properties** ofrece tres vistas de las propiedades y eventos para un elemento u otro objeto. La vista predeterminada es **Categorized** (como se ve en la primera de las siguientes pantallas), que muestra las propiedades dentro de las categorías expandibles, tal como **Appearance**, **Data** y **Design**. Los iconos en la ventana **Properties** le permiten cambiar entre esta vista y la vista **Alphabetic**, que es útil si está familiarizado con los nombres de las propiedades del objeto seleccionado. En ambos casos, se pueden configurar los valores de las propiedades al escribir un valor nuevo, seleccionándolo de una lista de valores predefinidos (tal como **True** o **False**), o mediante un "seleccionador" (por ejemplo, para elegir un color) o algún otro cuadro de diálogo.

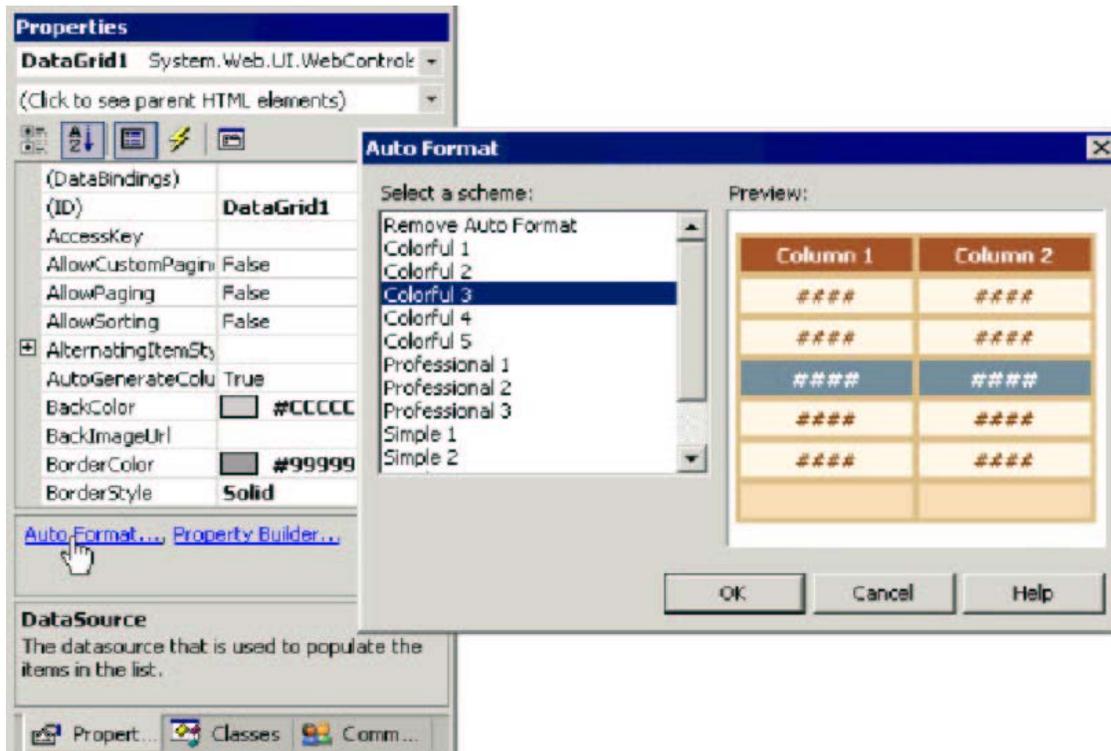
También existe un icono Events en la ventana Properties, en el cual puede hacer clic para cambiar la vista de forma que muestre los eventos ASP.NET del lado del servidor que están disponibles para el objeto seleccionado. Al hacer doble clic sobre un nombre de evento, automáticamente se inserta el código para un manejador de eventos vacío en la página, se muestra la página en la vista Code y se coloca el cursor en la posición justa para que escriba el código para el evento. También puede utilizar la lista desplegable de cada evento para conectarlo a un manejador de eventos existente en la página. Veremos este proceso en acción en la *Parte 2*.



Al hacer clic en el quinto icono (final) se abren cualesquiera Property Pages asociadas con el control seleccionado. Por ejemplo, con un control `DataGrid`, aparece un cuadro de diálogo que permite definir las plantillas para el control:



Cuando el control actualmente seleccionado es un `DataGrid` o un `DataList`, la sección inferior de la ventana de **Propiedades** contiene dos hipervínculos. El segundo de ellos, **Property Builder**, abre el cuadro de diálogo “página de propiedades” que aparece en la pantalla anterior. El otro vínculo, **Auto-Format**, presenta diversos colores y estilos predefinidos para la rejilla o la lista, así como para sus contenidos. El estilo se aplica simplemente al seleccionarlo en el cuadro de diálogo **Auto-Format**:



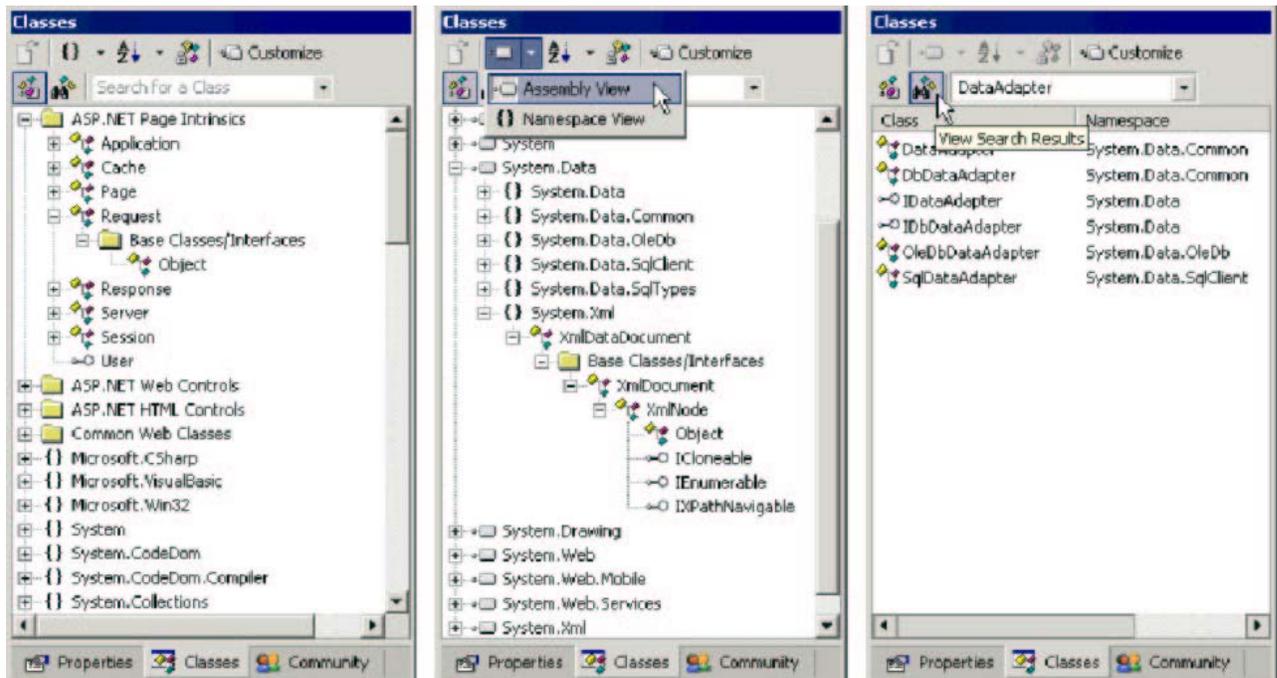
Por último, sobre los iconos en la ventana Properties, hay dos útiles listas desplegables que facilitan navegar a través de los controles en una página. La lista superior muestra el control actualmente seleccionado, además de enumerar todos los demás controles en la página, para permitirle seleccionar uno directamente. La lista desplegable que aparece debajo enumera todos los elementos para los cuales el elemento actualmente seleccionado es hijo o descendiente, para permitirle encontrar y elegir fácilmente cualquier elemento ancestro o padre (anexo).

La ventana Classes

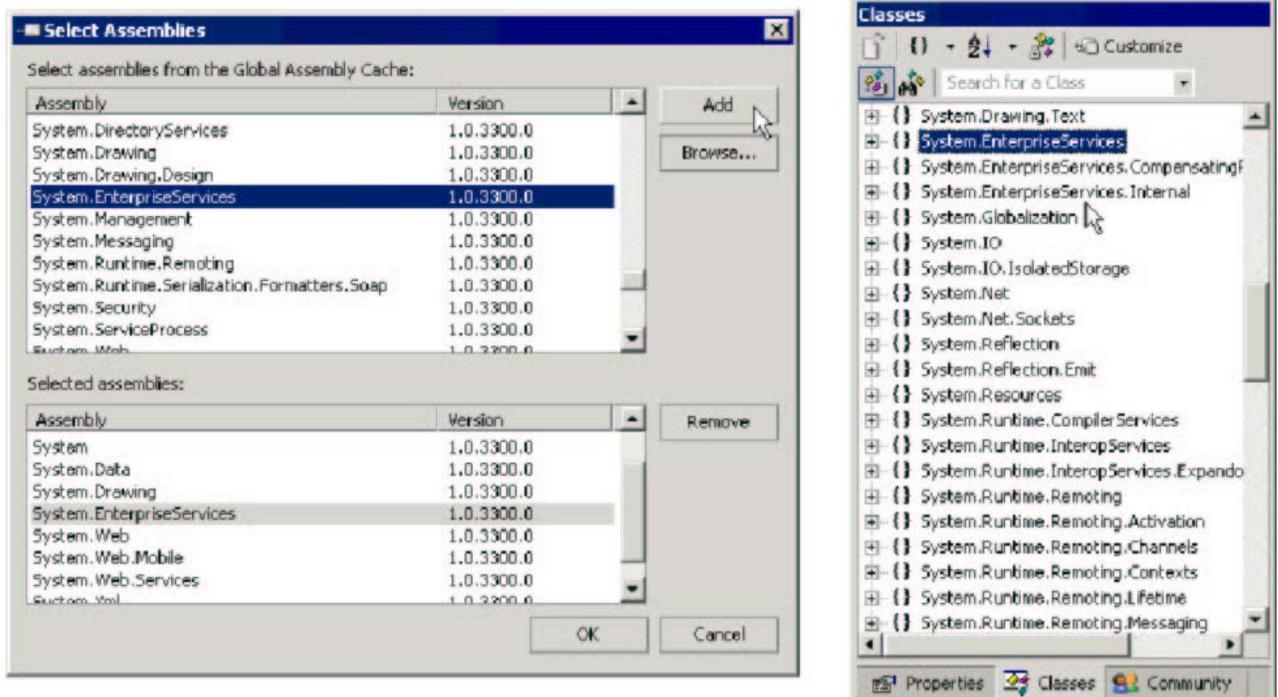
Al trabajar con .NET Framework, es esencial la información sobre la variedad de clases incluidas en la Biblioteca de clases .NET Framework. Para facilitar esto, Web Matrix incluye funciones que le permiten acceder a una ayuda detallada sobre cualquier clase dentro del IDE. De forma predeterminada, la ventana Classes muestra cuatro carpetas que contienen los Intrínsecos de página ASP.NET, el rango de los Controles Web, el rango de los Controles HTML y otras clases comunes para aplicaciones Web. Debajo de éstas se encuentran las otras clases comúnmente utilizadas, enumeradas por espacio de nombre.

Se pueden seleccionar otras vistas; por ejemplo, usted puede utilizar los iconos en la parte superior de la ventana Classes para mostrar los detalles en la vista Assembly (que enumera las clases por el DLL del ensamble que las contiene), o para clasificar la lista por orden alfabético, por tipo de clase o por visibilidad (esto es, si son públicas o privadas).

Por predeterminación, las clases que no son públicas no aparecen en la lista, aunque el cuarto icono en esta ventana se puede usar para mostrarlas además de las clases públicas. La ventana Classes también contiene un cuadro de texto de búsqueda, en el que puede registrar una cadena de búsqueda. Entonces sólo aparecerán las clases que contienen dicha cadena en su nombre (presione *Intro* para iniciar la búsqueda y luego haga clic en el icono View Search Results, para poder alternar entre los resultados de búsqueda y la lista de espacios de nombre):

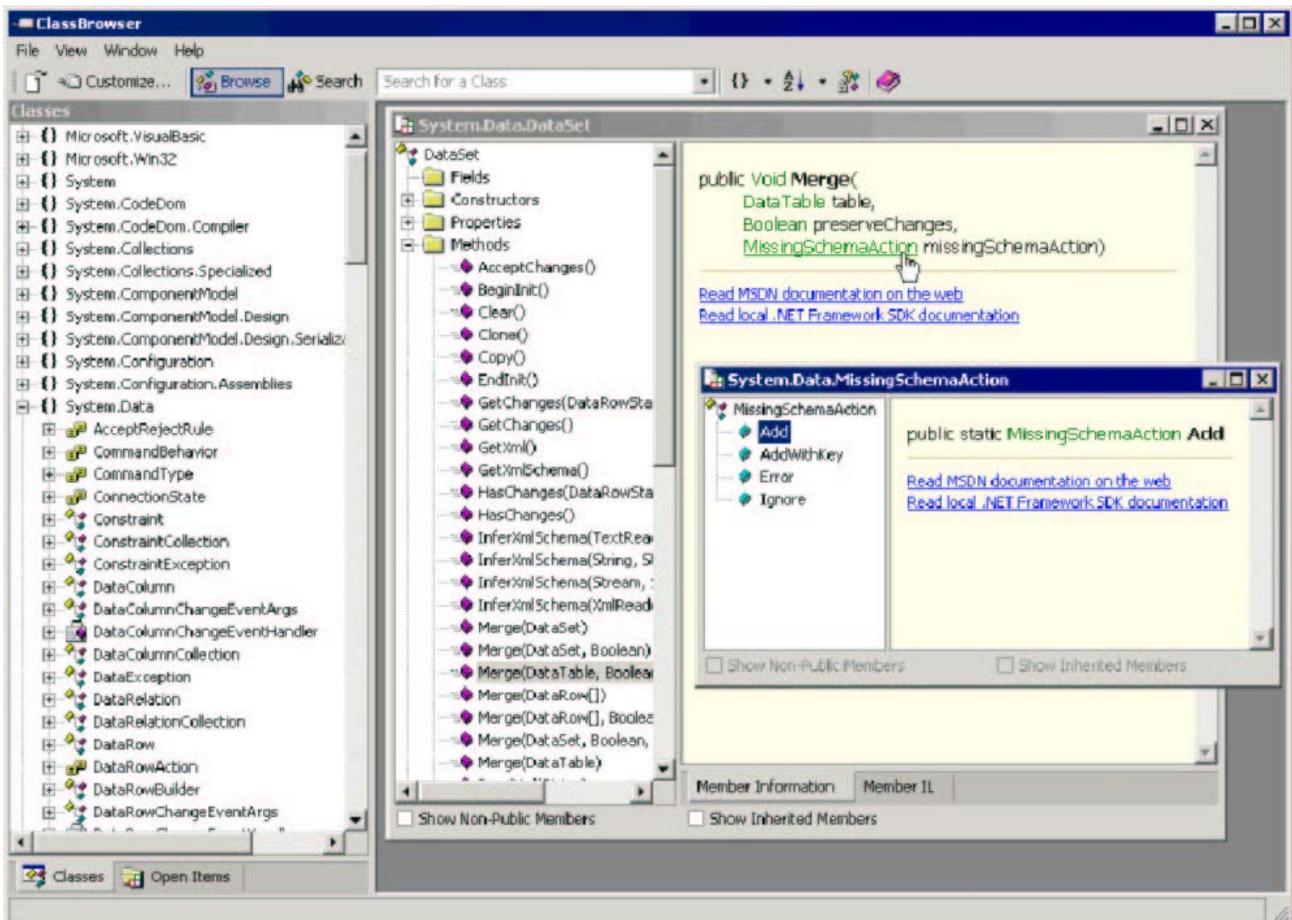


El icono Customize, en la parte superior derecha de la ventana Classes, se usa para modificar la lista de ensamblajes que aparecen en la ventana Classes. La siguiente pantalla muestra cómo agregar el ensamblaje System.EnterpriseServices a la lista en la ventana Classes, en la que luego aparecerán todos los espacios de nombre dentro de este ensamblaje:



El explorador de clases

Web Matrix incluye una herramienta completa – el ClassBrowser – que se puede abrir independientemente del menú Start, o desde dentro de Web Matrix, haciendo doble clic sobre una clase que aparezca en la ventana Classes. Cuando se abre desde el menú Start, tiene la apariencia de la siguiente pantalla. La ventana del lado izquierdo enumera los espacios de nombre de .NET Framework, y para cada uno de ellos, las clases e interfaces implementadas dentro del espacio de nombre enumerado. Al hacer doble clic en una clase, se abre una lista de todos los miembros de esa clase en una nueva ventana en el área derecha del Class Browser. Haciendo clic en uno de los miembros, aparecen entonces los detalles de dicho miembro en el área derecha de esta ventana – incluyendo los campos estáticos, los constructores, las propiedades, los métodos y los eventos:



El Class Browser utiliza la reflexión para obtener detalles acerca de la clase y de sus miembros, de tal forma que la información se limite sólo a las definiciones del miembro. Cada parámetro o enumeración que aparece en el panel derecho actúa como un hipervínculo, por lo que, al hacer clic sobre alguno de ellos muestra información sobre el objeto o enumeración en otra ventana. Esto facilita seguir la jerarquía y obtener detalles acerca de los parámetros del método o de los tipos de valores de las propiedades/campos que se requieren. También existen vínculos que abren la documentación local del SDK de .NET Framework (en caso de estar instalada), o la Biblioteca MSDN en línea en la página relevante, en donde se pueden encontrar más detalles sobre la clase y sus miembros.

La barra del menú y la barra de herramientas en el **Class Browser** ofrecen el mismo conjunto de funciones que la ventana **Classes** que analizamos antes. Usted puede cambiar el orden de clasificación y la organización de los espacios de nombre y de las clases en la ventana izquierda, buscar las clases por nombre y mostrar u ocultar las clases que no son públicas. También aparece el mismo botón **Customize**, que abre el cuadro de diálogo **Select Assemblies**, en el cual puede agregar y eliminar ensamblajes desde la lista derecha. El menú **Window** se puede utilizar para colocar las ventanas en mosaico o en cascada, o sólo para cambiar entre una y otra.

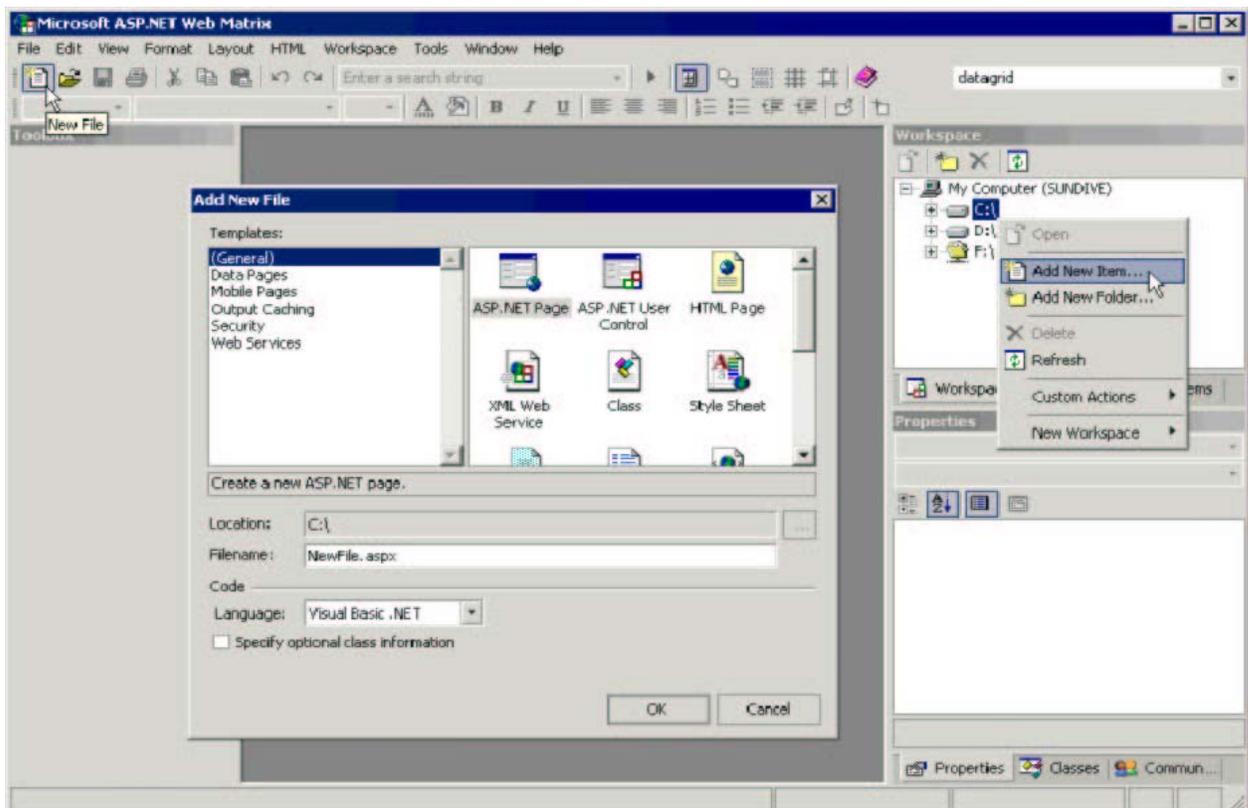
Cuando la información sobre una clase se abre desde dentro del IDE de Web Matrix (haciendo doble clic sobre una clase en la ventana **Classes**) aparecen dentro del IDE las mismas secciones del lado derecho de la página mostradas en la pantalla anterior, pero no aparece la lista de espacios de nombre del lado izquierdo.

Tipos de archivos y asistentes

Existen tres formas de abrir un nuevo archivo en Web Matrix. Puede:

- ❑ Seleccionar **New** desde el menú **File**
- ❑ Hacer clic en el icono **New File** en la barra de herramientas
- ❑ Hacer clic con el botón alterno en la carpeta objetivo en la ventana **Workspace** y seleccionar **Add New Item**

Las dos últimas de estas técnicas aparecen en la siguiente pantalla compuesta, junto con el cuadro de diálogo **New File** que aparece. Observe que la ruta es la seleccionada en la ventana **Workspace**:



El cuadro de diálogo **New File** enumera los diversos tipos de archivos que usted puede crear y editar dentro del IDE de Web Matrix. Cada uno es creado desde una plantilla almacenada en carpetas, dentro de la subcarpeta `Templates` de la carpeta `Program Files\Microsoft ASP.NET Web Matrix\version\`. Cada una de ellas se describe a continuación.

Tipos de archivo en la sección (General)

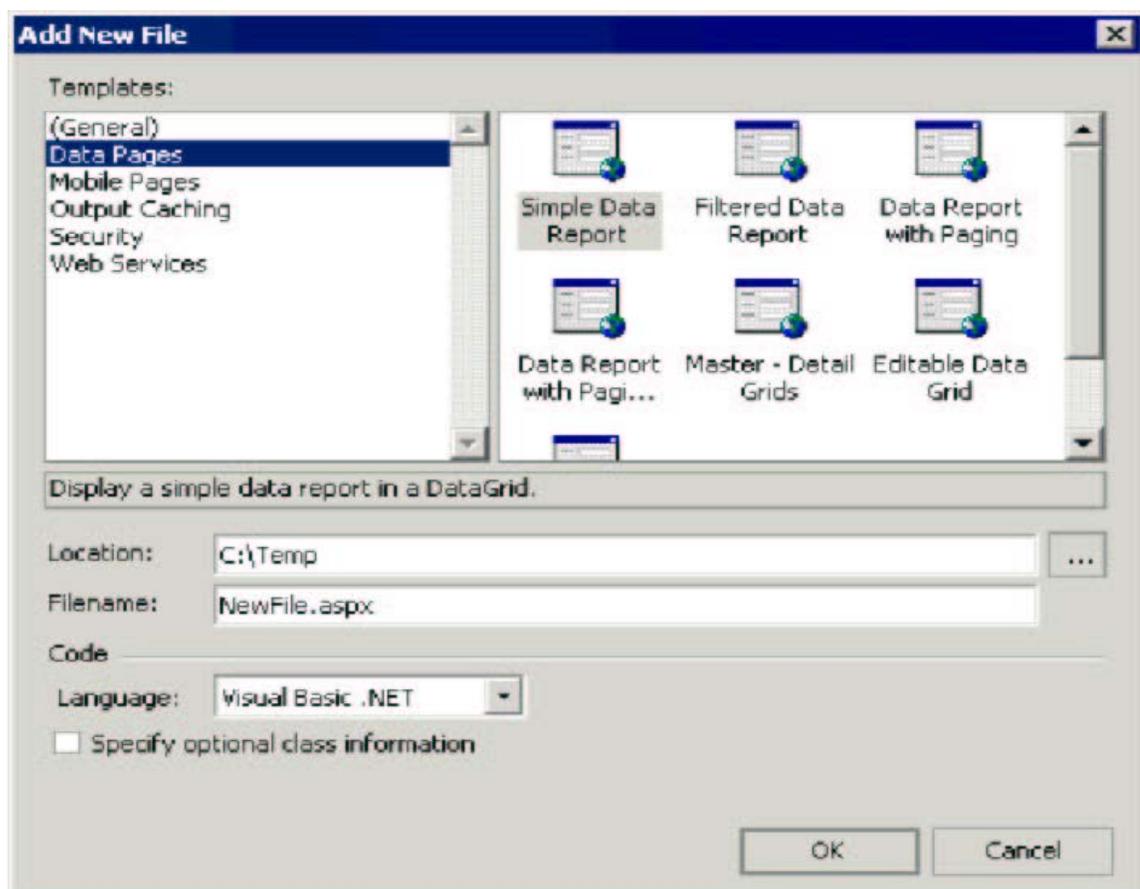
Existen 14 diferentes tipos de archivos que usted puede crear desde la sección (**General**) del cuadro de diálogo **New File**. Estos son:

- ❑ **ASP.NET Page** – esto crea un archivo con la extensión `.aspx`. El archivo contiene la directiva `@Page`, las etiquetas para abrir y cerrar `<html>`, una sección `<head>` vacía y una sección `<body>` que contiene una `<form>` del lado del servidor.
- ❑ **ASP.NET User Control** – esto crea un archivo con la extensión `.ascx`. El archivo contiene sólo la directiva `@Control`.
- ❑ **HTML Page** – esto crea un archivo con la extensión `.htm`. El archivo contiene las etiquetas para abrir y cerrar `<html>`, más las secciones `<head>` y `<body>` vacías.
- ❑ **XML Web Service** – esto crea un archivo con la extensión `.asmx`. El archivo contiene la directiva `@WebService`, declaraciones `Imports` o `using` para los espacios de nombre que se requieren del servicio Web, una definición de `Class`, y un ejemplo de función pública que muestra lo que puede modificar. Debe especificar el nombre de clase y el espacio de nombre antes de poder crear este tipo de archivo.
- ❑ **Class File** – esto crea un archivo con la extensión `.vb` o `.cs` (dependiendo del lenguaje que especifique), contiene una declaración `Imports` o `using` para el espacio de nombre `System`, la definición `namespace`, una definición de `Class`, y una `Sub` o `function` pública vacía. Debe especificar el nombre de clase y el espacio de nombre antes de crear este tipo de archivo.
- ❑ **Style Sheet** – esto crea un archivo con la extensión `.css`. El archivo contiene sólo una definición de selector `BODY{ }` vacía.
- ❑ **Global.asax** – esto crea un archivo con la extensión `.asax`. El archivo contiene la directiva `@Application` y una sección `<script>` que contiene manejadores de eventos vacíos para los eventos `Application_Start`, `Application_End`, `Application_Error`, `Session_Start` y `Session_End`.
- ❑ **Web.Config** – esto crea un archivo `Web.config` que contiene las secciones `<configuration>`, `<appSettings>` y `<system.Web>`. Dentro de la sección `<system.Web>` están los elementos `<sessionState>`, `<customErrors>`, `<authentication>` y `<authorization>`. Todos los elementos se comentan por predeterminación, y contienen una descripción de su uso, así como los valores válidos para los atributos comunes.
- ❑ **XML File** – esto crea un archivo con la extensión `.xml`. El archivo contiene sólo la instrucción de procesamiento `<?xml ... ?>` que define la versión y la codificación del archivo.
- ❑ **XSL Transform** – esto crea un archivo de hoja de estilo XSLT con la extensión `.xslt`. El archivo contiene la instrucción de procesamiento `<?xml ... ?>` y el elemento raíz `<stylesheet>`.

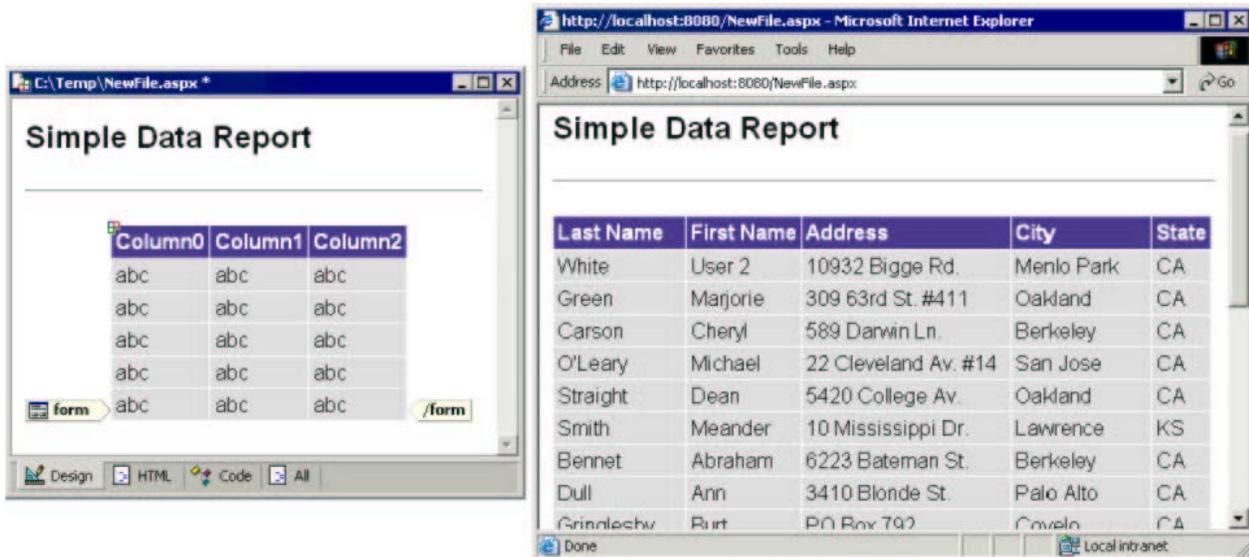
- ❑ **XML Scheme** – esto crea un archivo de esquema XML (XSD) con la extensión `.xsd`. El archivo contiene la instrucción de procesamiento `<?xml ... ?>` y el elemento raíz `<xsd:schema>`.
- ❑ **ASP.NET HTTP Handler** – esto crea un archivo con la extensión `.ashx`. El archivo contiene la directiva `@WebHandler`, las declaraciones `Imports` o `using` para los espacios de nombre `System` y `System.Web` que se requieren, una definición de `Namespace` y una definición de `Public Class` que implementa la interfaz `IHttpHandler`. Dentro de la definición `Clases` aparecen dos definiciones de miembro que se requieren para el evento `ProcessRequest` y la propiedad `IsReusable`. Debe especificar el nombre de clase y el espacio de nombre antes de poder crear este tipo de archivo.
- ❑ **Text File** – esto crea un archivo de texto vacío con la extensión `.txt`.
- ❑ **SQL Script** – esto crea un archivo de texto que contiene sólo `"/ * New SQL script */`. El archivo tiene una extensión de `.sql`.

Tipos de archivo en la sección de páginas de datos

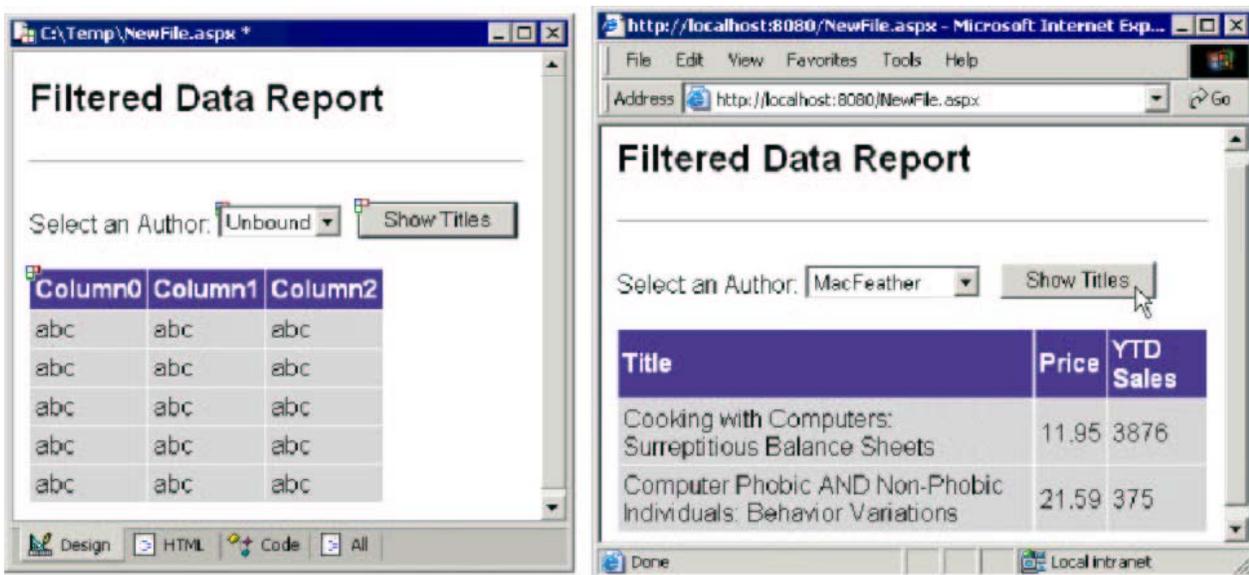
Mientras que los tipos de archivo que se enumeran en la sección (General) son predominantemente archivos “para delinear” vacíos, los tipos de archivo enumerados en otras secciones tienden más a ser como Asistentes, pero sin ningún cuadro de diálogo paso por paso. Las plantillas que utilizan para crear el archivo nuevo contienen códigos y (en algunos casos), controles de servidor ASP.NET para implementar una página de trabajo que puede utilizar como un punto de inicio para desarrollar sus propias páginas. Los tipos de archivo disponibles en la sección `Data Pages` se muestran en la siguiente pantalla, y luego se describen:



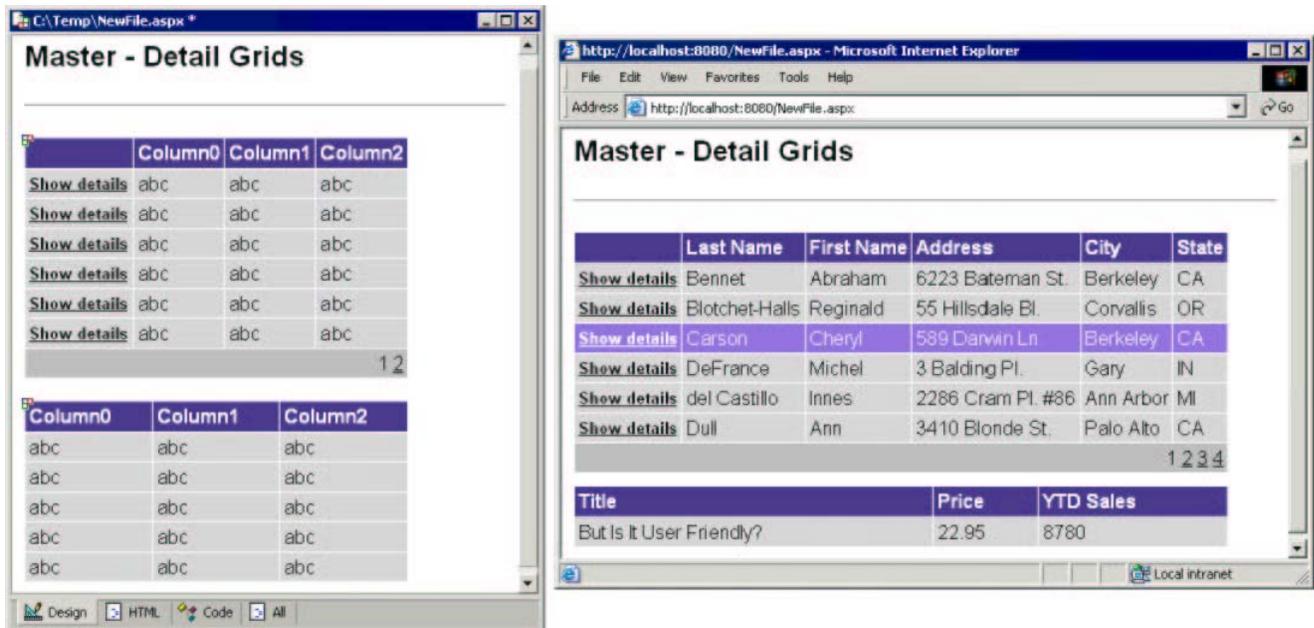
- **Simple Data Report** – esto crea una página que accede a SQL Server local o a la base de datos MSDE y muestra detalles de la tabla Authors de la base de datos Public de ejemplo, utilizando un `DataReader` como la fuente de datos para un control `DataGrid` de ASP.NET. La siguiente pantalla muestra la página en la vista Design, y cuando se abre en un explorador:



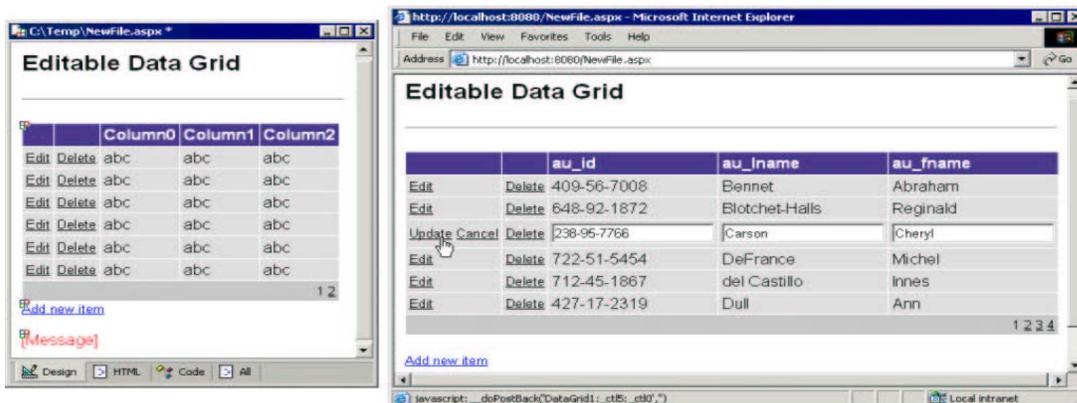
- **Filtered Data Report** – esto crea una página similar al ejemplo previo, pero esta vez contiene controles en donde puede seleccionar un autor y ver una lista de sus libros. Se utiliza un `DataReader` cuando la página se carga primero para llenar la lista desplegable. Luego, cuando se hace clic en `Show Titles`, se construye una declaración SQL apropiada y se utiliza con un `DataReader` para extraer la información de la vista SQL `titleview` en la base de datos pubs. La siguiente pantalla muestra la página en la vista Design, y cuando se abre en un explorador:



- ❑ **Data Report with Paging** – esta página llena un `DataSet` con datos de la tabla `Authors` y luego la une a un control `DataGrid` de ASP.NET. Sin embargo, esta vez utiliza las funciones de paginación integradas de la `DataGrid` para mostrar los resultados en páginas por separado, junto con los vínculos que permiten a un usuario navegar a través de las mismas.
- ❑ **Data Report with Paging and Sorting** – esta página amplía las técnicas desarrolladas en el tipo de página previo, al agregar una facilidad de clasificación. Lo hace al establecer los atributos del control `DataGrid`, y al agregar un manejador de eventos sencillo para responder al evento `Sort` de la rejilla.
- ❑ **Master- Detail Grids** – esta página muestra lo fácil que es mostrar datos a partir de dos tablas relacionadas. Los datos de la tabla `Autores` en la base de datos `pubs` de ejemplo, se cargan en un `DataSet` y aparecen en un control `DataGrid` que tiene activada la paginación, y que contiene un `ButtonColumn` con el texto Show details. El hacer clic en uno de estos vínculos del botón, causa que un `DataReader` relacione los datos que corresponden de la vista SQL `titleview` en la base de datos `pubs` y ésta aparece en el segundo control `DataGrid`. La siguiente pantalla muestra la página en la vista `Design`, y cuando se abre en un explorador:



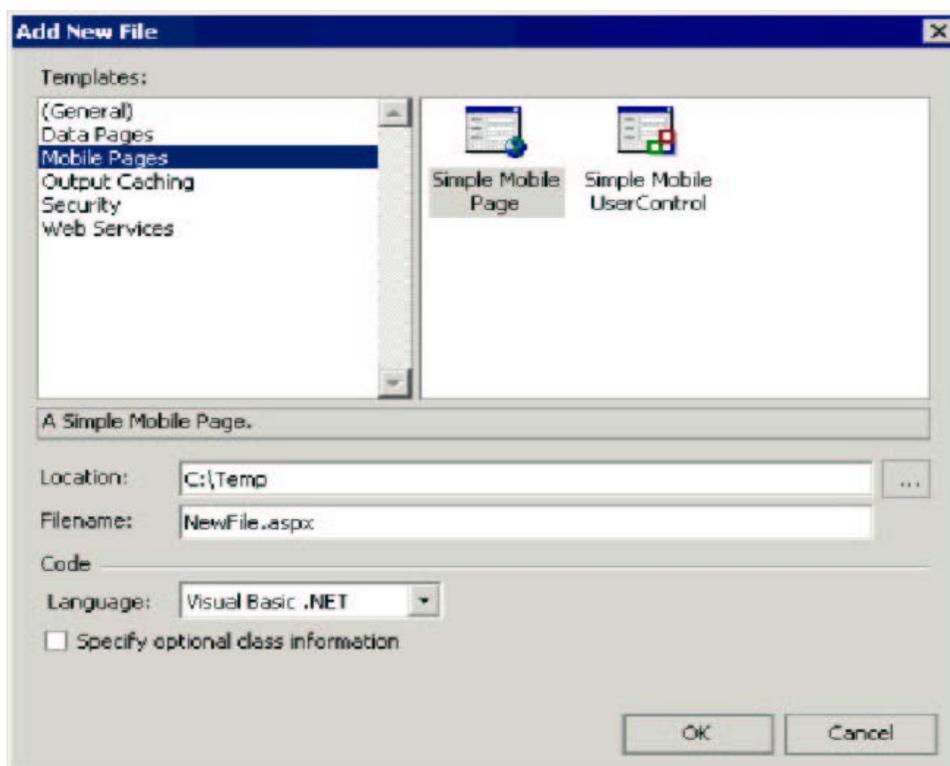
- ❑ **Editable Data Grid** – este ejemplo muestra la técnica básica para editar datos utilizando las funciones integradas del control `DataGrid` de ASP.NET. Un `DataSet` se llena con datos de la tabla `Authors`. Un `EditCommandColumn`, que implementa los botones del vínculo `Edit/Update/Cancel` y un `ButtonColumn`, que implementa los botones del vínculo `Delete`, se agregan luego a esta `DataGrid`. El código en la página reacciona a los diversos eventos que surgen de estos botones de vínculos; el código ejecuta las declaraciones SQL que actualizan los contenidos de la tabla de la base de datos original. La siguiente pantalla muestra la página en la vista `Design`, y cuando se abre en un explorador:



- **Simple Storing Procedure** – este ejemplo es similar al primero de las Data Pages que analizamos. La única diferencia es que invoca al procedimiento almacenado nombrado `CustOrdersDetail` dentro de la base de datos pubs, en lugar de utilizar una declaración SQL almacenada como una cadena. El resultado se regresa como un `DataReader`, que se une a un control `DataGrid` ASP.NET en la página.

Tipos de archivo en la sección de páginas móviles

Web Matrix contiene plantillas que le permiten crear páginas “móviles” y controles de usuario. Estas páginas se basan en las clases expuestas por el kit de herramientas de Microsoft Mobile Internet (MMIT). El MMIT se puede utilizar para crear páginas que se adaptan automáticamente a una amplia gama de dispositivos. Estas páginas, y los controles que contienen, producen resultados HTML o WML (Lenguaje de marcación inalámbrico), ajustándolos para el dispositivo en particular que accede a la página:



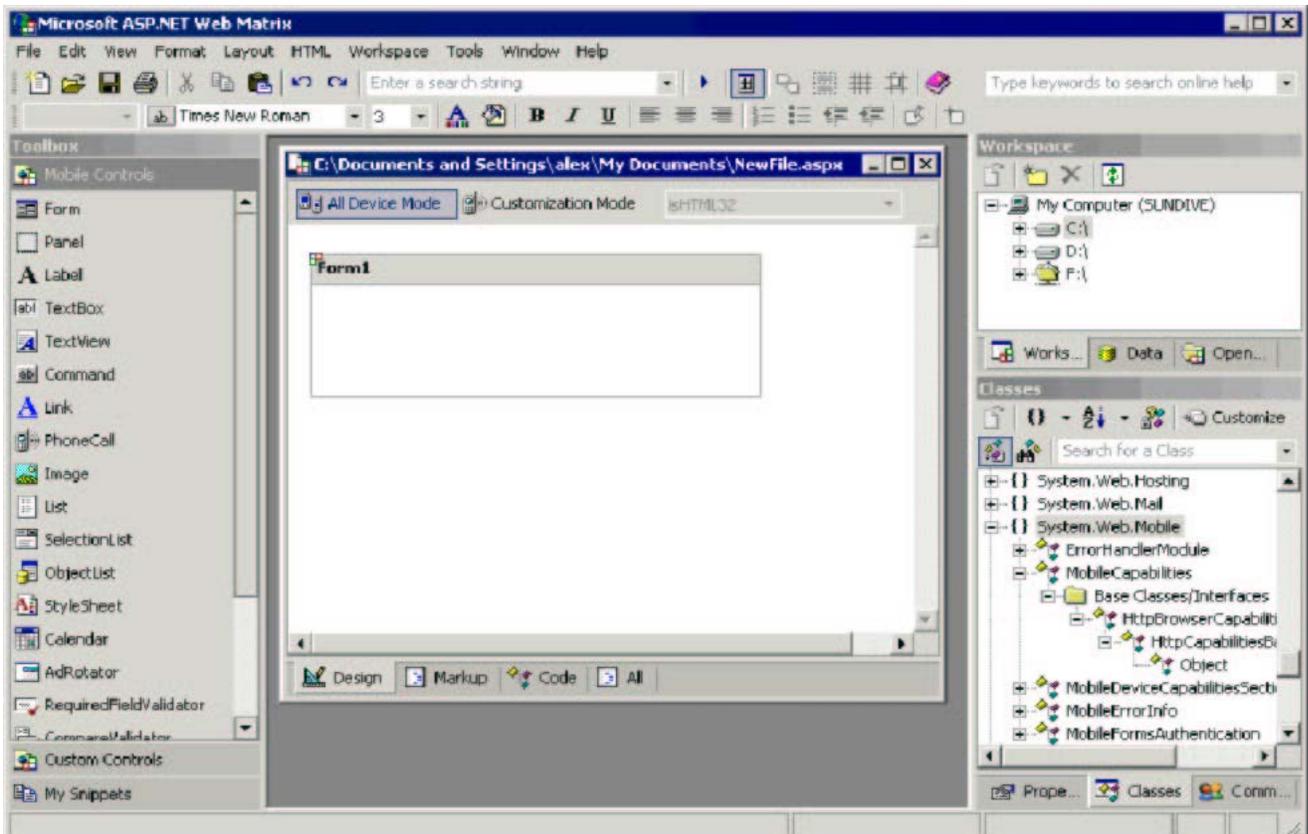
Existen dos tipos de archivos en la sección Mobile Pages del cuadro de diálogo Add New File:

- **Simple Mobile Page** – esto crea un archivo con la extensión `.aspx`. El archivo contiene la directiva `@Page` y la hereda de la clase móvil `MobilePage` especial que se implementa dentro del Kit de herramientas de Microsoft Mobile Internet (MMIT). El archivo también incluye la directiva `@Register` para los Mobile Controles y una `<mobile:Form>` vacía del lado del servidor.
- **Simple Mobile User Control** – esto crea un archivo con la extensión `.ascx`. El archivo contiene la directiva `@Control` pero la hereda de la clase especial `MobileUserControl` implementada dentro del Kit de herramientas de Microsoft Mobile Internet (MMIT). El archivo también incluye la directiva `@Register` para los Controles móviles.

El ambiente para las páginas móviles

Cuando la página que se está editando actualmente dentro de Web Matrix es una página móvil, el ambiente cambia para proporcionar las funciones especiales que se requieren para este tipo de página. La barra de herramientas muestra ahora la sección Mobile Controls, que contiene los controles del MMIT. Estos son los únicos controles que se deben utilizar en páginas móviles, ya que los controles HTML y Web Forms estándar no pueden generar el contenido correcto en todas las circunstancias (debido a que no pueden producir WML).

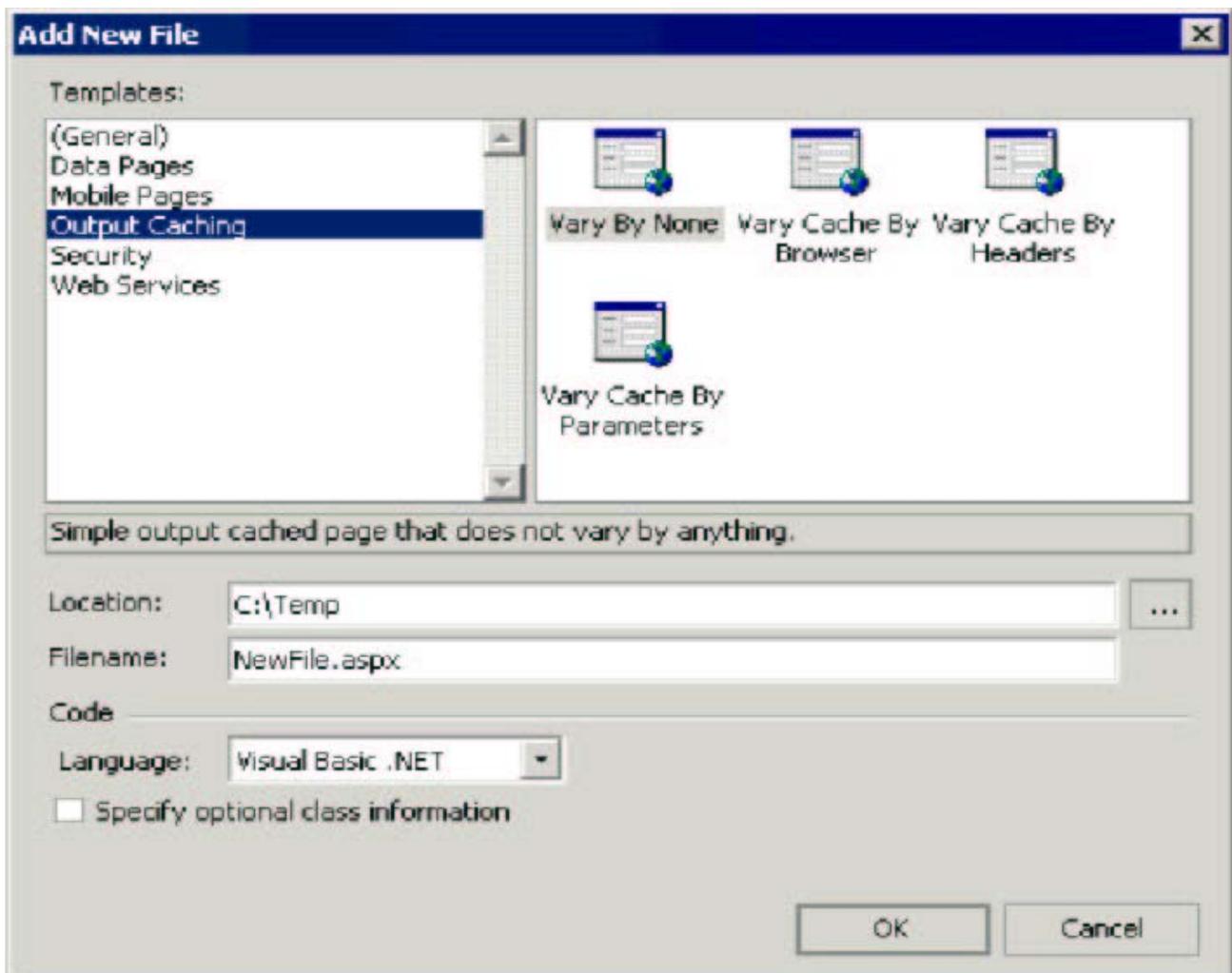
La siguiente pantalla muestra que la ventana Edit cambia también. Gana controles para especificar cómo se filtrará la página y reaccionará a diferentes dispositivos. En el MMIT, es posible establecer filtros de dispositivos, para que las secciones de la salida se puedan modificar para dispositivos específicos. Estos controles se utilizan para configurar ese filtro.



En la parte inferior de la ventana Edit, existe un cambio menor en las pestañas para las cuatro diferentes vistas de la página. Markup reemplaza a HTML en la segunda pestaña, ya que la salida de una página móvil puede ser WML en lugar de HTML. Además, los controles y las clases del Kit de herramientas de Internet Mobile se incluyen en la lista predeterminada de clases que aparecen en la ventana Classes, y en la herramienta ClassBrowser por separado.

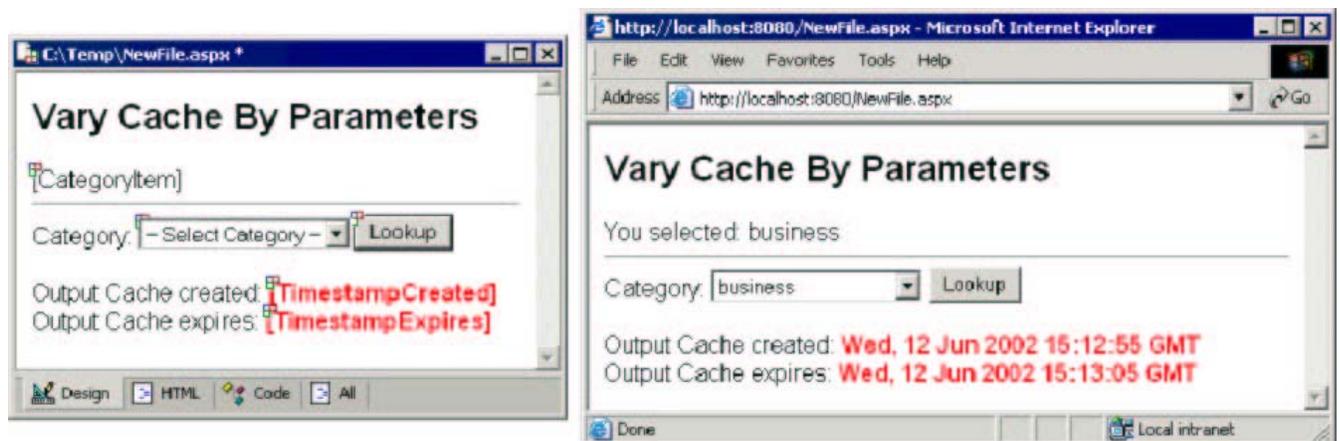
Tipos de archivo en la sección de memoria caché de salida

La sección Output Caching del cuadro de diálogo Add New File contiene ejemplos de cómo puede configurar páginas que utilizan memoria caché de salida para mejorar el rendimiento, minimizar los gastos del servidor y reducir los tiempos de respuesta. Los cuatro tipos de archivos disponibles son fundamentalmente similares, y demuestran cómo se puede configurar la memoria caché de salida para detectar automáticamente las diferentes funciones de cada solicitud:



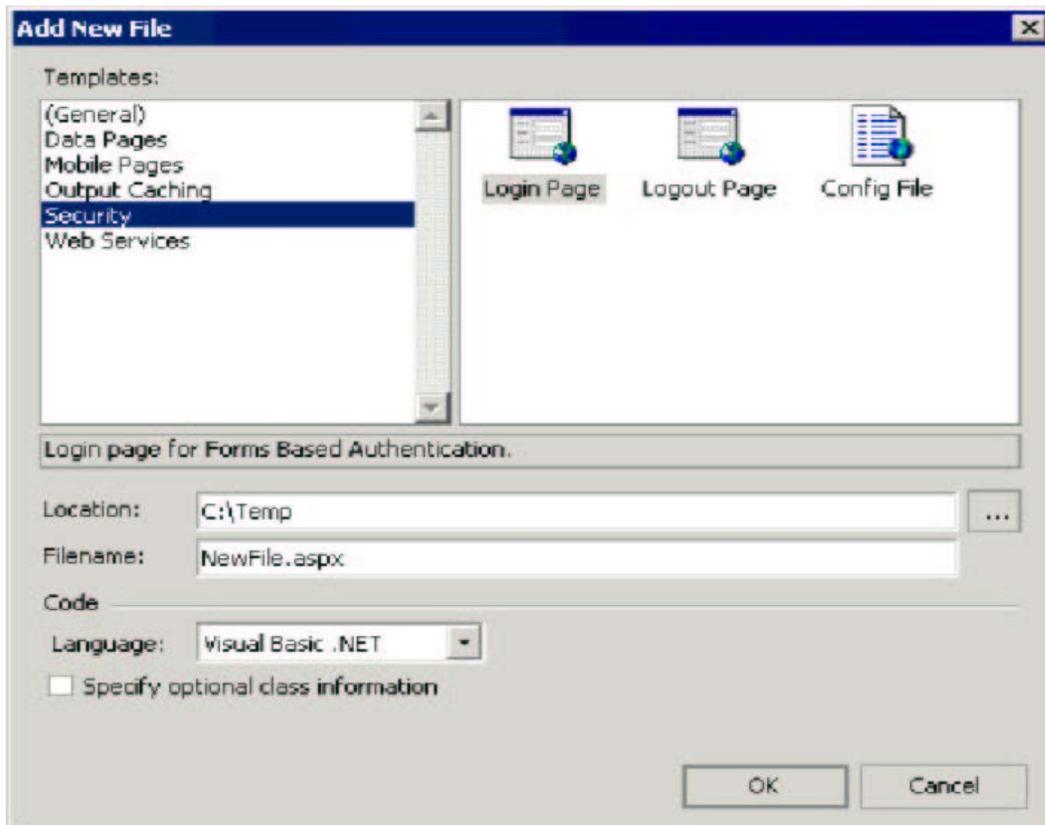
- ❑ **Vary by None** – esto demuestra la memoria caché de salida “total”, en donde a cada cliente se le envía la misma página en memoria caché hasta que expira. El ejemplo contiene una directiva `@OutputCache` que especifica una duración de memoria caché de 10 segundos, y contiene el atributo `VaryByParam="none"`. Se configuran dos controles de etiqueta en la página a la hora actual y a la hora en que expira la memoria caché. Al actualizar la página en el explorador, puede ver el efecto de la memoria caché de salida.

- ❑ **Vary Cache by Browser** – esto demuestra la memoria caché de salida que envía diferentes páginas a cada tipo de explorador, con base en la detección del explorador que realiza el objeto `Request.Browser`. El ejemplo contiene una directiva `@OutputCache` que especifica la duración de una memoria caché de 10 segundos, y contiene los atributos `VaryByParam="none"` y `VaryByCustom="browser"`. Esta vez existen tres controles de etiqueta en la página, que se establecen para el nombre del explorador, la hora actual y la hora en que expira la memoria caché. Al actualizar la página en el explorador, y cargarla en diferentes tipos de exploradores, puede ver el efecto de la memoria caché de salida.
- ❑ **Vary Cache by Headers** – esto demuestra la memoria caché de salida que envía diferentes páginas dependiendo de un encabezado HTTP específico enviado en la solicitud. Este ejemplo contiene una directiva `@OutputCache` que nuevamente especifica una duración de memoria caché de 10 segundos, con los atributos `VaryByParam="none"` y `VaryByHeader="Accept-Language"`. La misma página sólo se enviará en respuesta a las solicitudes en donde el encabezado `Accept-Language` es el mismo. Se establecen tres controles de etiqueta en la página al valor del encabezado `Accept-Language`, la hora actual y la hora en que expira la memoria caché.
- ❑ **Vary Cache by Parameters** – esto demuestra la memoria caché de salida que envía diferentes páginas dependiendo del valor enviado como parámetro desde el cliente – en este caso, un valor colocado desde el control de la lista desplegable en una `<form>`. Este ejemplo contiene una directiva `@OutputCache` que especifica una duración de la memoria caché de 120 segundos, con el atributo `VaryByParam="Category"` (el id y nombre de la lista desplegable). Se configuran tres controles de etiqueta en la página al valor seleccionado en la lista desplegable, la hora actual y la hora en que expira la memoria caché. Al seleccionar una categoría diferente y al hacer clic en el botón `Lookup` genera que se elabore una página y se coloque en la memoria caché para esa categoría, sólo si no hay ya una disponible en la memoria caché. Las siguientes pantallas muestran esta página en la vista de Diseño, y cuando se abren en un explorador:

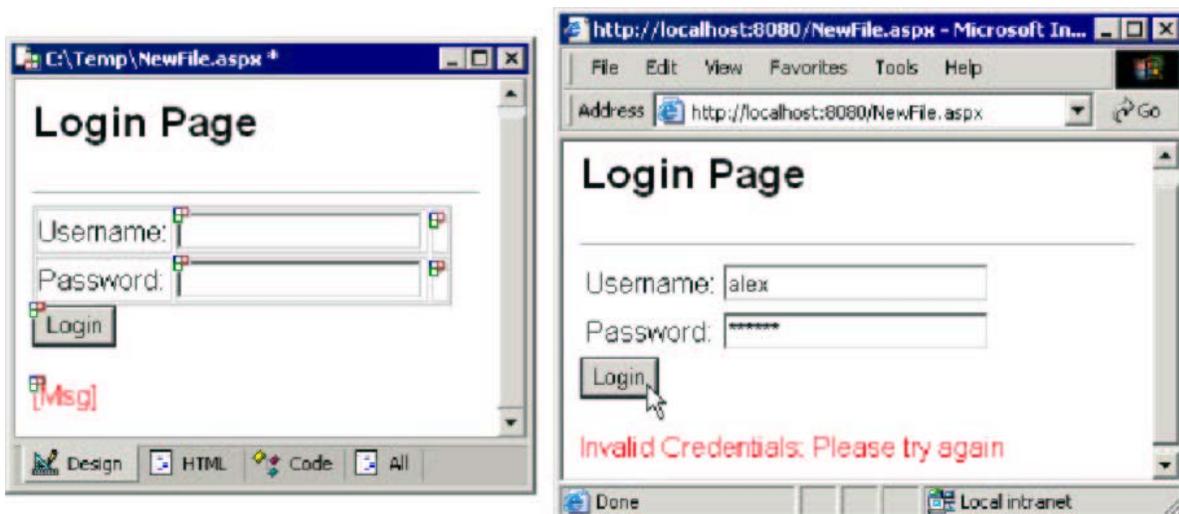


Tipos de archivo en la sección de seguridad

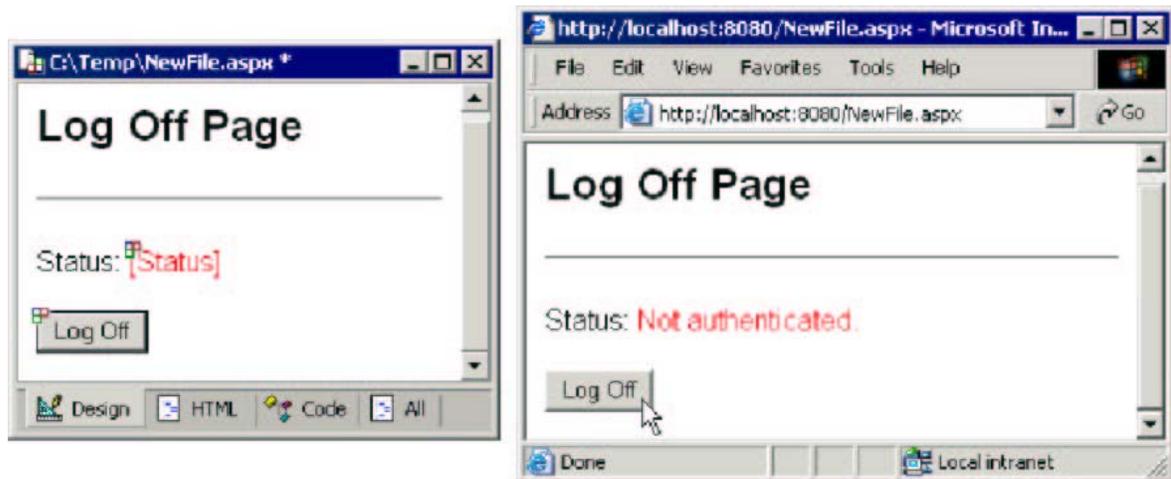
Existen tres ejemplos en la sección Security del cuadro de diálogo Add New File, que demuestra las técnicas comunes para crear páginas de autenticación (registro) para una sección segura de un sitio Web:



- **Login Page** – esto crea una página de “registro” estándar que contiene dos cuadros de texto con controles `RequiredFieldValidator` correspondientes a nexos, un botón de Login y una etiqueta en donde se pueden mostrar cualesquiera errores. El código en la página utiliza una verificación codificada sencilla de los valores que registra, y luego muestra cómo ejecutar el método `RedirectFromLoginPage` para cargar la página que se le solicitó originalmente. Las siguientes pantallas muestran esta página en la vista Design, y cuando se abre en un explorador:



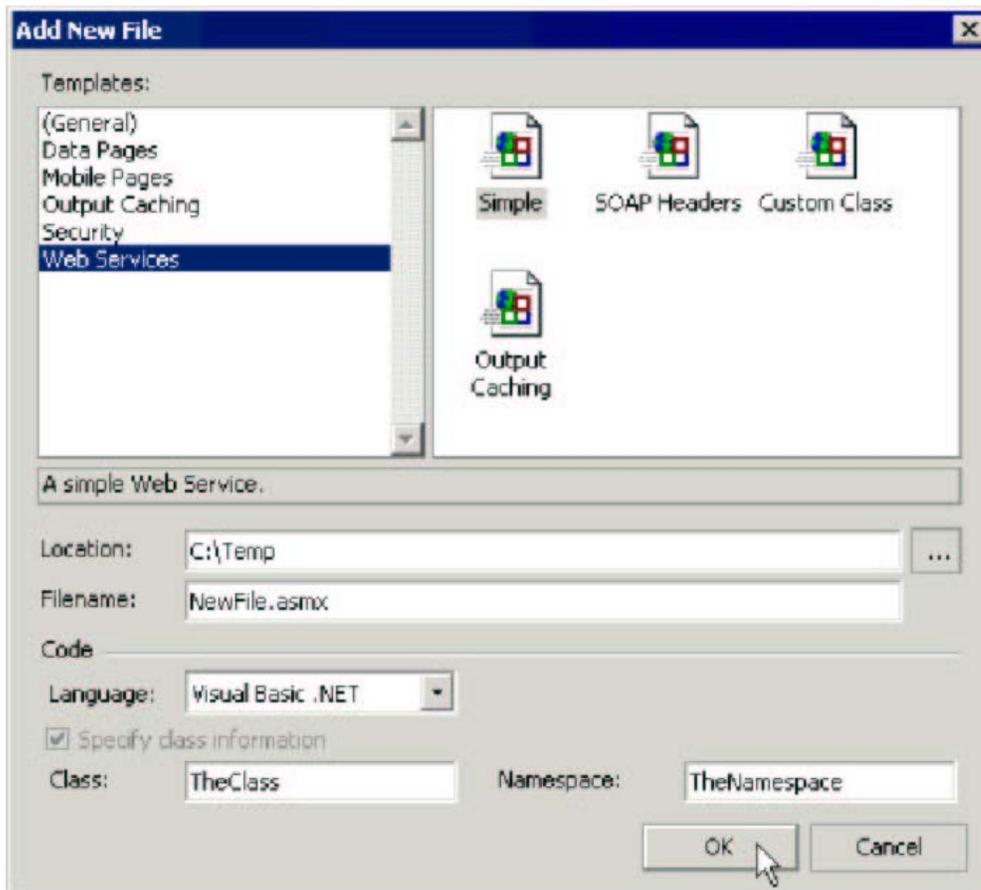
- **Logout Page** – esto crea la página de “salida” correspondiente, con una etiqueta de Status y un botón de Log Off individual. La etiqueta muestra el nombre del usuario actualmente registrado, cuando está disponible. Al hacer clic en el botón se invoca el método `SignOut` y aparece un mensaje que indica que el usuario ya no está autenticado. Las siguientes pantallas muestran la página de Logout en la vista Design, y cuando se abre en un explorador:



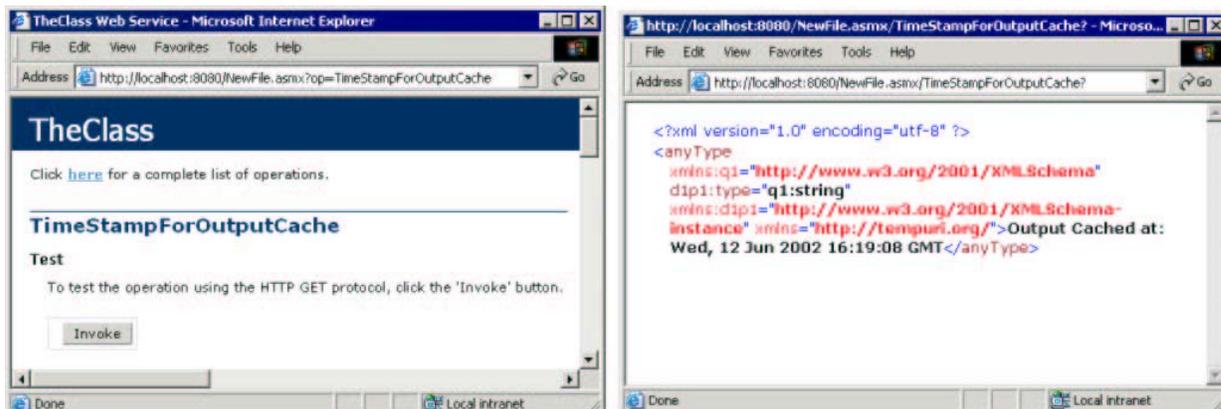
- **Config File** – este ejemplo crea un archivo `Web.config` adecuado para utilizarlo con los ejemplos de seguridad anteriores. El archivo contiene un elemento `<configuration>` con un elemento `<system.Web>` hijo. El elemento `<system.Web>` contiene los elementos `<authentication>` y `<authorization>` que especifica la autenticación de las Forms, y rechaza a los usuarios anónimos.

Tipos de archivo en la sección de servidores Web

La sección final del cuadro de diálogo Add New File es la sección Web Services. Esta incluye cuatro páginas de ejemplo que implementan diferentes funciones de los Servicios Web. Para cada una de ellas, debe registrar el nombre de clase y el espacio de nombre antes de poder crear el archivo:



- ❑ **Simple** – este ejemplo crea el tipo más sencillo de Servicio Web, básicamente el mismo que la opción XML Web Service en la sección (General) del cuadro de diálogo New File. El archivo contiene una directiva `@WebService`, declaraciones `Imports` o `using` para los espacios de nombre del Servicio Web que se requieren, una definición de `Class` y un ejemplo de función pública que muestra lo que puede modificar.
- ❑ **SOAP Headers** – este ejemplo crea un servicio Web que lee un valor personalizado a partir de los encabezados SOAP de la solicitud, y muestra el resultado.
- ❑ **Custom Class** – este ejemplo muestra cómo se puede regresar una clase personalizada desde un Servicio Web. El código crea una instancia de una clase personalizada nombrada `OrderDetails` (que en realidad es un arreglo de otra clase personalizada que recibe el nombre de `OrderItem`), establece algunos valores para los miembros de la clase y luego devuelve esta instancia.
- ❑ **Output Caching** – este ejemplo demuestra cómo se puede colocar en la memoria caché la salida de un Servicio Web, de forma muy parecida a los ejemplos mostrados antes que utilizaron memoria caché de salida. Sencillamente define una función pública que se implementa como un `WebMethod`, y agrega un atributo `CacheDuration` con un valor de 30 para la función, de manera que el resultado se incluya en la memoria caché automáticamente durante 30 segundos. Las siguientes pantallas muestran la página que se abrió en un explorador, y el resultado:



Lenguaje, nombres de clases y espacios de nombre

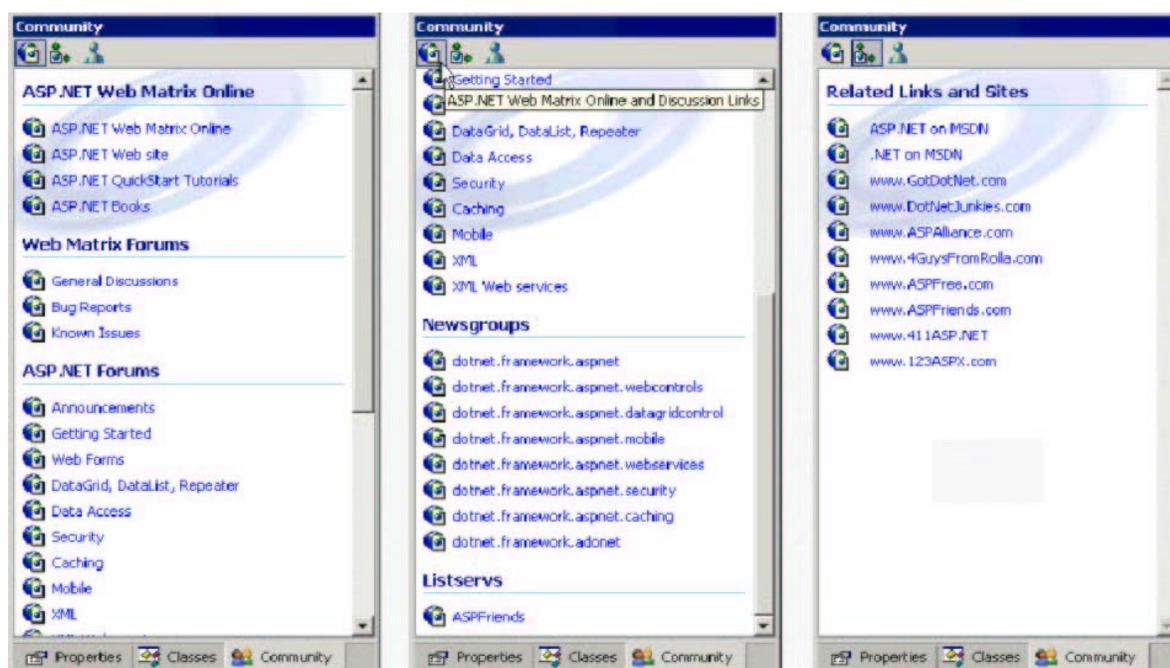
Recuerde que, para cada tipo de archivo seleccionado en el cuadro de diálogo New File, cualquier código que se incluya automáticamente en el archivo se encuentra en el lenguaje que especificó en ese cuadro de diálogo; la opción está entre Visual Basic .NET y C#. Dependiendo del tipo de archivo que seleccionó, el cuadro de diálogo también contendrá controles en los cuales se registra cualquier otra información requerida – tal como nombre de clase y espacio de nombre (en algunos casos esto es opcional, mientras que en otros, como en el Web Service o en el archivo Class, es obligatorio). Más adelante crearemos algunas páginas de ejemplo, para demostrar estas técnicas generales.

Ayuda, soporte e información de referencia

Hemos visto cómo Web Matrix proporciona acceso a los materiales de referencia y a la ayuda en línea de diversas formas. Existen planes futuros para que Web Matrix incluya sus propios archivos de ayuda completos que describan las funcionalidades del IDE, y cómo obtener los mejores resultados del producto. Actualmente sólo se implementan las funciones mínimas de ayuda integradas, tales como los vínculos a diversos recursos y ejemplos en <http://www.asp.net/> y <http://www.getdotnet.com/>. Sin embargo, si coloca el cursor sobre un nombre de clase en la ventana Edit y presiona la tecla *F1*, se abre una nueva ventana ClassBrowser con detalles de referencias sobre esa clase.

Otros lugares dentro del IDE de Web Matrix también proporcionan acceso a la ayuda y al soporte en línea. La ventana Comunidad (en la parte inferior de la ventana “proyecto”) contiene vínculos hacia el sitio de Web Matrix ASP.NET, así como vínculos a varios grupos de noticias .NET ejecutados por Microsoft, y enumera servidores proporcionados por otros miembros de la comunidad de Web Matrix.

El sitio ASP.NET de Web Matrix forma parte del sitio ASP.NET principal en <http://www.asp.net/WebMatrix>, que también contiene una gran cantidad de información útil y de vínculos a otros sitios relacionados con ASP.NET. También es la fuente principal para complementos descargables, bibliotecas de control y otros recursos para Web Matrix – incluyendo el acceso a la versión más reciente del producto. A continuación se presentan dos vistas de la primera página, para que pueda ver el rango de recursos que se proporcionan:



La segunda página (que se abre desde el segundo icono en la parte superior de la ventana Community) contiene vínculos a sitios Web relacionados y a otros recursos, mientras que la tercera página (que se abre desde el tercer icono) accede a MSN Messenger (si está instalado en su máquina), para que pueda chatear en tiempo real con otros usuarios de Web Matrix.

No olvide que la barra de herramientas principal en la parte superior de la ventana de Web Matrix contiene una lista combinada desplegable en la cual puede escribir una pregunta o una serie de palabras clave. Al presionar *Intro* se abre el sitio ASP.NET de Web Matrix en su explorador predeterminado, y muestra una lista de los artículos y recursos relacionados con su consulta.

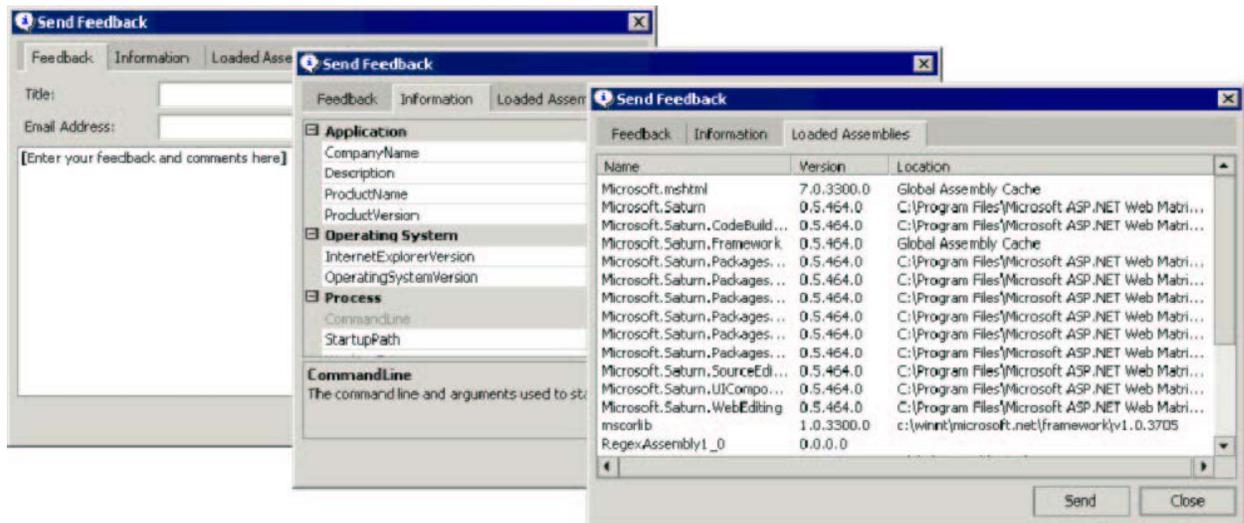
También recordará, de nuestro análisis anterior, que la ventana ClassBrowser contiene vínculos para cada miembro de la Biblioteca de clases de .NET Framework que abre las páginas de ayuda y de referencia correspondientes, ya sea localmente desde su propia máquina, o desde el sitio de la biblioteca MSDN Online.

Enviar retroalimentación a Microsoft

El menú Help en el IDE de Web Matrix contiene una entrada para enviar a Microsoft retroalimentación sobre el producto. Dicha retroalimentación puede consistir en reportes de errores, solicitudes de funciones, o sólo información general y comentarios.

Web Matrix es un “producto de la comunidad”, y, como tal, su desarrollo futuro será guiado, en gran medida, por la retroalimentación que Microsoft reciba de los usuarios. Así pues, siéntase libre de enviar sus opiniones, ¡el equipo de desarrollo está deseoso de escuchar lo que piensa!

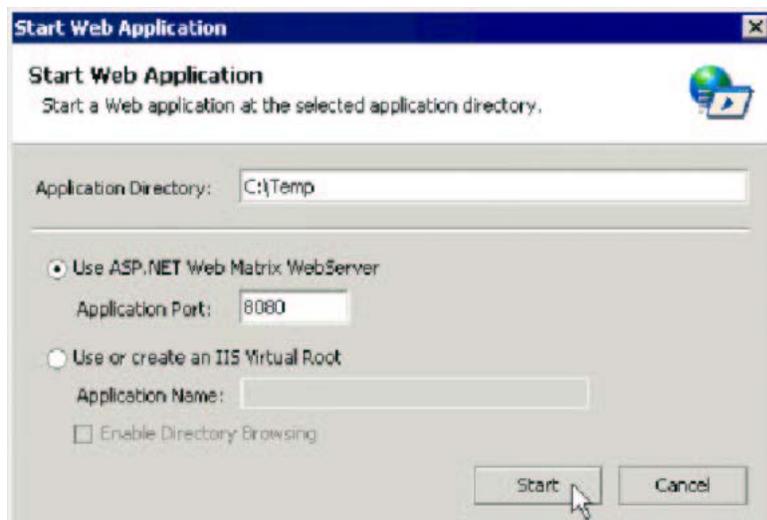
La ventana Send Feedback es un cuadro de diálogo tabulado de tres páginas, que contiene la página de Feedback misma, la página de información de la aplicación y una lista de todos los Loaded Assemblies actualmente (aparece el mismo cuadro de diálogo, pero sin la página de Feedback, cuando se selecciona el comando Application Information desde el menú Help):



Servidor Web ASP.NET de Web Matrix de Microsoft

Antes de pasar a la *Parte 2*, en donde veremos Web Matrix en acción, veremos rápidamente el servidor Web incluido en Web Matrix. Éste es un servidor Web delgado y ligero que se puede utilizar para ejecutar páginas ASP.NET y otros recursos (tales como Servicios Web) sobre máquinas que aún no tienen instalado un servidor Web local.

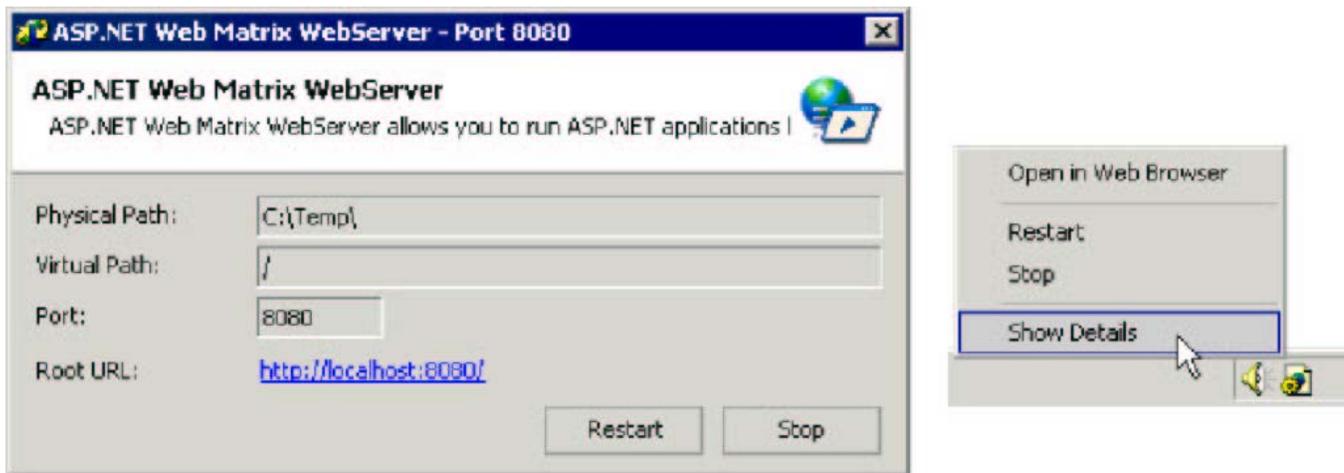
Cuando ejecuta por primera vez una página ASP.NET o un Servicio Web desde dentro del IDE de Web Matrix, se abre un cuadro de diálogo que le pregunta cuál servidor Web desea utilizar. Como se muestra en la siguiente pantalla, puede permitir al Servidor Web ASP.NET de Web Matrix de Microsoft ejecutar su página o Servicio Web:



Como puede ver, el valor predeterminado para el Servidor Web es ejecutar sobre el puerto 8080. Esto es ideal si la máquina que está utilizando ya cuenta con un Servidor Web (como IIS) instalado y funcionando. El Servidor Web existente tiende a utilizar el puerto 80, y así, al utilizar un puerto diferente el Servidor Web de Web Matrix evita cualquier posibilidad de error. Puede cambiar el Application Port a un puerto diferente si así lo desea (tal como un puerto 80, si no tiene IIS instalado).

De manera alterna, puede seleccionar una instancia existente de Internet Information Server (IIS) para ejecutar su página, en cuyo caso Web Matrix creará una nueva raíz virtual (con el nombre que usted especifique) que señala hacia la carpeta que contiene el archivo que está editando. Si lo desea, también puede activar la exploración del directorio para esta raíz virtual, lo cual facilita encontrar y ejecutar páginas individuales conforme desarrolla su aplicación.

Una vez que se ha iniciado el Servidor Web de Web Matrix, aparece un icono en la barra de tareas de Windows. Al hacer clic con el botón alterno en este icono aparece un menú en el cual puede abrir el sitio Web que le proporciona el Servidor Web en su explorador predeterminado, Restart o Stop el Servidor Web, o mostrar detalles sobre éste:



Parte 2 – Poner Web Matrix a trabajar

Ahora que conoce las tareas que puede realizar el ASP.NET de Web Matrix de Microsoft, es hora de ponerlas en práctica. Web Matrix es fácil de usar; por ello, no mostraremos todos sus aspectos en acción. En cambio, crearemos un sitio Web sencillo para una compañía de pizzas, concentrándonos en las páginas que se utilizan con mayor frecuencia. Esto le mostrará qué poco necesita hacer con Web Matrix para tener un sitio listo y funcionando.

Pizza Pretty Quick

Nuestro sitio Web de ejemplo se diseñó para permitir a los clientes elegir pizzas y bebidas, agregarlas a una canasta de compras sencilla y después continuar hacia la salida, en donde pagan, ya sea en efectivo al momento de la entrega, o facturándoles a una cuenta. Es realmente un sitio de comercio electrónico sencillo, y deja fuera muchas funciones (¡cómo tener una apariencia más bonita!) que no se requieren. Nuestro resultado es un sitio sencillo como éste:

Due to our Brownian Motion based Quantum improbability drive, we can get our pizzas to you almost before you've order them. Causality doesn't stand in the way of hot pizza.

Pizzas

(click a slice to select)

- Margherita**
The basic pizza. Nice, but dull.
Toppings: Cheese and tomato
- Hawiian**
A bit fruity. Server by someone wearing a loud shirt.
Toppings: Ham and pineapple
- Carnivore Special**
For those who need their meat. A thin crust pizza base, topped with a 16oz steak.
Toppings: A cow
- Three Cheese Special**
It's a bit runny
Toppings: Cheese, Cheese and more Cheese.

Sizes

- small (\$4.95)
- medium (\$6.95)
- large (\$8.95)
- huge (\$14.95)

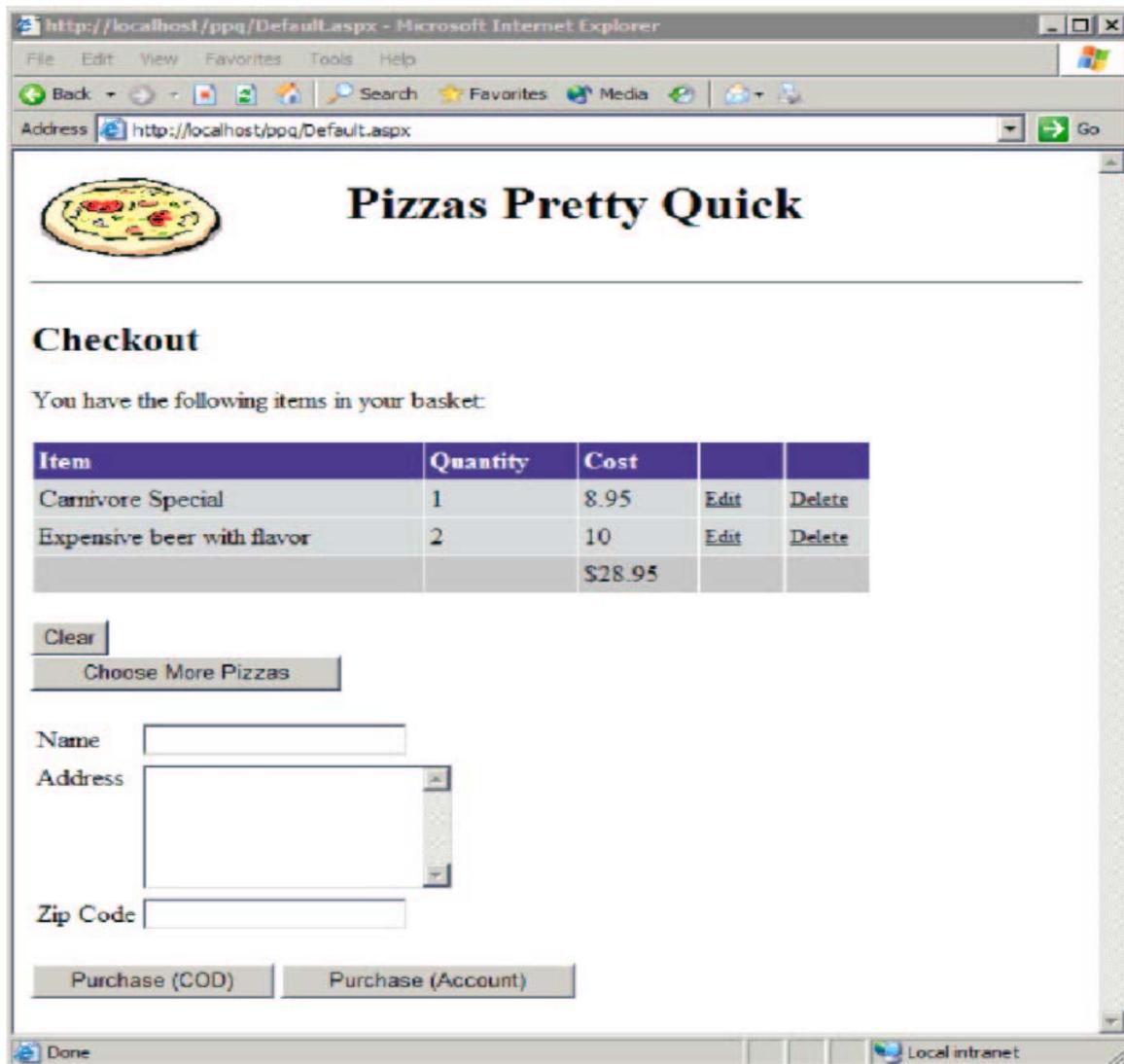
Drinks

Drink	Quantity
Cola (\$ 2)	<input type="text" value="1"/>
Cheap tasteless beer (\$ 5)	<input type="text"/>
Expensive beer with flavor (\$ 10)	<input type="text"/>

Basket

Item	Quantity	Cost		
Carnivore Special	1	8.95	Edit	Delete
Expensive beer with flavor	2	10	Edit	Delete
		\$28.95		

Desde esta página, un cliente puede seleccionar entre una amplia gama de tipos y tamaños de pizza y una gran variedad de bebidas. Su selección se agrega a una canasta de compras y cuando el cliente ha elegido todas sus opciones, puede continuar hacia la salida del sitio:



La página de salida muestra de nuevo la selección del cliente y permite colocar el pedido. El cliente puede elegir pagar cuando se le entrega la pizza, o que se le facture a una cuenta. Si el cliente elige que se le facture a una cuenta, será llevado a una página de registro segura, donde puede acceder sus detalles de la cuenta.

Todo el código para este ejemplo está disponible desde <http://www.AIAndDave.com/books/Web Matrix/>.

Crear páginas ASP.NET

Debido a que no estamos creando un sitio completamente funcional, hemos eliminado ciertas partes que normalmente utilizaría. Por ejemplo, sólo contamos con un nivel de acceso de datos mínimo, seguridad limitada y algunas funciones avanzadas. Esto es porque lo importante es mostrar los tipos de páginas que Web Matrix puede crear, y lo que usted debe hacer para personalizarlas según sus requerimientos.

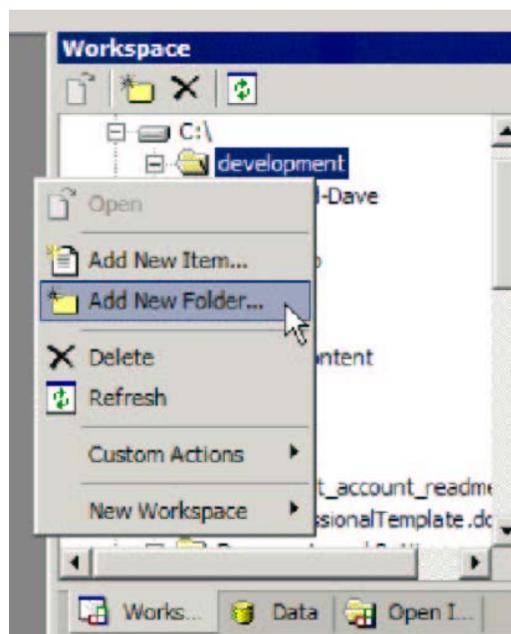
Debido a que Web Matrix se basa en archivos, necesita configurar el Directorio virtual de IIS usted mismo. Yo lo llamo PPQ.

En las siguientes secciones vamos a:

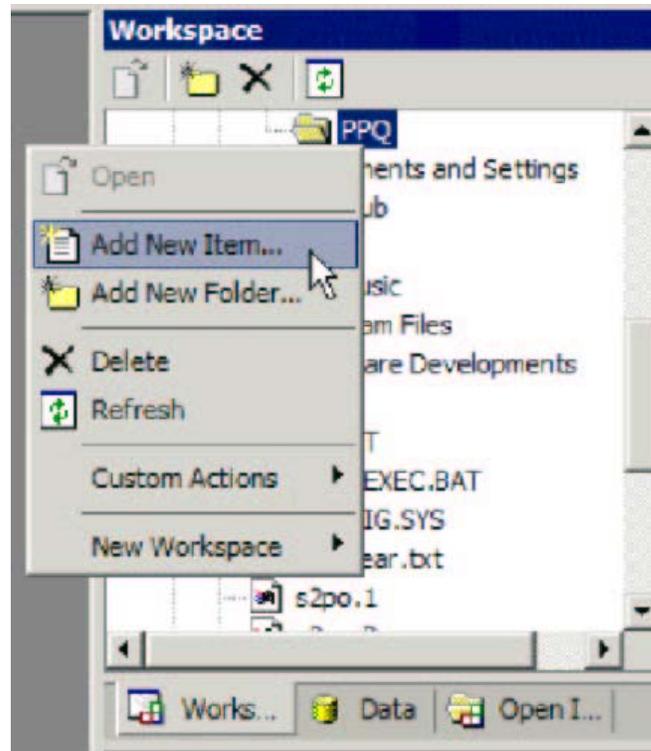
- ❑ Crear un Nivel de datos que consiste en un archivo XML, un componente VB.NET y algunos procedimientos y tablas almacenados en SQL
- ❑ Crear Controles de usuario para el encabezado de la página y para la canasta de compras
- ❑ Crear la página de selección de pizzas, en donde usamos diversos controles ASP.NET, así como los Controles de usuario recientemente creados
- ❑ Crear la página de salida, en donde tomamos los detalles del usuario y cómo desean pagar
- ❑ Crear páginas seguras para los clientes con cuentas
- ❑ Crear una variedad de páginas diferentes, tal como aquellas que utilizan una rejilla maestra y de detalles, o aquellas que requieren memoria caché
- ❑ Aprender cómo crear Servicios Web
- ❑ Aprender cómo utilizar otros controles, tal como los Controles de Internet Explorer y los controles personalizados

La capa de datos

La capa de datos para esta aplicación consiste en dos archivos: un archivo XML que contiene los datos y una clase que carga los datos y realiza ciertas conexiones de la base de datos. Cuando inicia por primera vez Web Matrix aparecerá un cuadro de diálogo **New File** (recuerde que Web Matrix se basa en archivos, y no en proyectos como Visual Studio .NET). Para mantener juntos los archivos para este sitio Web, debemos crear un directorio – esto se puede hacer ya sea externamente en el explorador, o desde Web Matrix utilizando el **Workspace**, en donde seleccionamos **New Directory** desde el menú de contexto.



Una vez que creamos el directorio, debemos empezar a crear los archivos para la aplicación. Antes que nada, debemos crear el archivo XML. Esto se puede hacer desde el elemento New... en el menú File, o mediante el menú de contexto en el directorio:

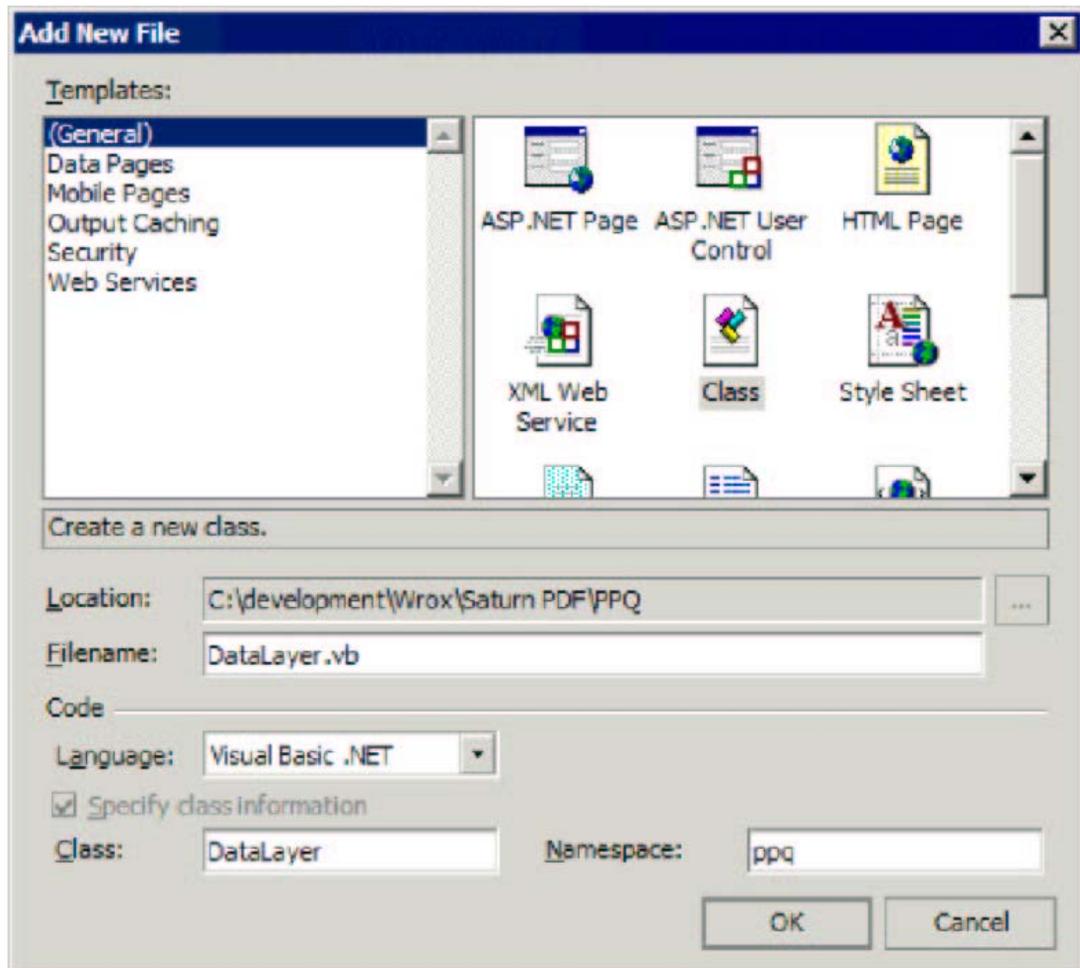


Esto le presenta el cuadro de diálogo New File, donde puede seleccionar XML File de las plantillas generales (General). Después sólo necesita agregar sus datos:

```
C:\development\Wrox\Saturn PDF\PPQ\pizzas.xml
1 <?xml version="1.0" encoding="utf-8" ?>
2 <ppq>
3   <pizza name="Margherita" ingredients="Cheese and tomato"
4     description="The basic pizza. Nice, but dull."/>
5   <pizza name="Hawian" ingredients="Ham and pineapple"
6     description="A bit fruity. Server by someone wearing a loud shirt."/>
7   <pizza name="Carnivore Special" ingredients="A cow"
8     description="For those who need their meat. A thin crust pizza base, topped with a 16oz steak."/>
9   <pizza name="Three Cheese Special" ingredients="Cheese, Cheese and more Cheese."
10    description="It's a bit runny"/>
11   <drink name="Cola" price="2"/>
12   <drink name="Cheap tasteless beer" price="$"/>
13   <drink name="Expensive beer with flavor" price="10"/>
14   <size name="small ($4.95)" price="4.95"/>
15   <size name="medium ($6.95)" price="6.95"/>
16   <size name="large ($8.95)" price="8.95"/>
17   <size name="huge ($14.95)" price="14.95"/>
18 </ppq>
19
20
```

En realidad, es como si usara una base de datos para todos estos detalles, pero esta es una solución rápida que muestra la sencillez de Web Matrix; observe que no existen funciones de edición XML especiales, tal como validación de XML, que agreguen complejidad a la herramienta.

Para utilizar estos datos, creamos una clase llamada DataLayer:



Aquí tenemos la opción de seleccionar el lenguaje predeterminado, el nombre de clase y el espacio de nombre para la clase. La plantilla creada es un talonario en el cual puede agregar el código requerido. Pudimos utilizar el desarrollador de código Insert Data Method para agregar el código, pero el código generado por el desarrollador de código crea una declaración SQL que se debe ejecutar, y nosotros deseamos utilizar un par de procedimientos almacenados. Agregaremos el código manualmente (aunque aún se podría utilizar el desarrollador de código y después modificar el código generado):

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Imports System.Web
Imports System.Xml

Namespace ppq

    Public Class DataLayer

        Public Sub New()
            End Sub

        Public Shared Function GetData() As DataSet

            Dim ctx As HttpContext = HttpContext.Current
            Dim ds As New DataSet()
```

```

    dr.ReadXml(ctx.Server.MapPath("~/pizza.xml"))
    Return dr
End Function

Public Shared Sub LogOrder (Name As String, Address As String, _
    ZipCode As String)

    Dim ctx As HttpContext = HttpContext.Current
    Dim Basket As DataTable = CType(ctx.Session("Basket"), DataTable)

    Dim conn As New SqlConnection("server=... | * & _
        *Database=AlandEve; Trusted_Connection=true")
    conn.Open ()

    ' add the order
    Dim cmd As New SqlCommand()
    cmd.Connection = conn
    cmd.CommandText = "sp_PPInsertOrder"
    cmd.CommandType = CommandType.StoredProcedure

    cmd.Parameters.Add("@Name", SqlDbType.VarChar, 25).Value = Name
    cmd.Parameters.Add("@Address", SqlDbType.VarChar, 155).Value = Address
    cmd.Parameters.Add("@ZipCode", SqlDbType.VarChar, 15).Value = ZipCode

    Dim OrderID As Integer = cmd.ExecuteScalar()

    ' add the order details
    cmd.Parameters.Clear()
    cmd.CommandText = "sp_PPInsertOrderItem"
    cmd.Parameters.Add("@fkOrderID", SqlDbType.Int)
    cmd.Parameters.Add("@Item", SqlDbType.VarChar, 25)
    cmd.Parameters.Add("@quantity", SqlDbType.Int)
    cmd.Parameters.Add("@Cost", SqlDbType.Decimal)

    cmd.Parameters("@fkOrderID").Value = OrderID
    Dim row As DataRow
    For Each row In Basket.Rows
        cmd.Parameters("@Item").Value = row("Description")
        cmd.Parameters("@quantity").Value = row("quantity")
        cmd.Parameters("@Cost").Value = row("Cost")
        cmd.ExecuteNonQuery()
    Next

    conn.Close()
End Sub

End Class

End Namespace

```

Esta clase tiene dos métodos sencillos. El primer método simplemente carga y devuelve el archivo XML como un DataSet. El segundo método accede a una base de datos SQL para poder agregar detalles del pedido y líneas del pedido. Todos estos son códigos estándar que utilizan un par de procedimientos y parámetros almacenados. Lo más importante que debe comprender acerca de estos códigos, es que la canasta de compras se sostiene en la sesión actual; más adelante veremos cómo se crea esto.

Compilar clases

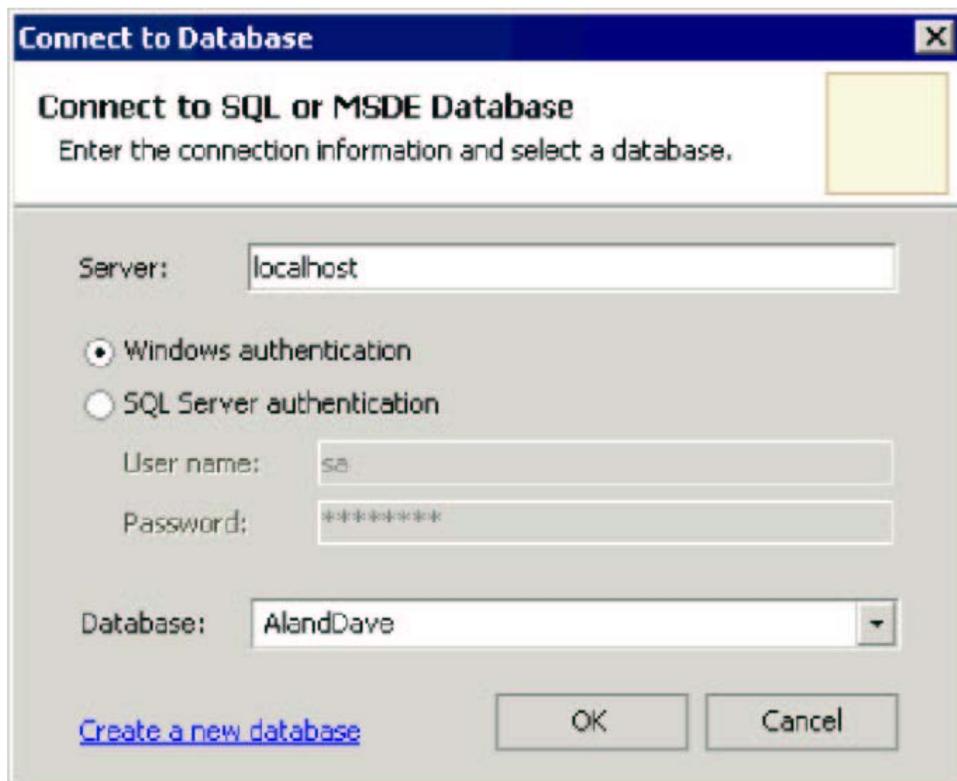
Una cosa que no hace Web Matrix es la compilación, por lo que debemos realizarla manualmente. Sólo creamos un archivo de lote conteniendo lo siguiente (observe que este comando está todo en una línea):

```
vbc /debug /nologo /t:library /out:bin/DataLayer.dll /r:System.dll  
/r:System.Xml.dll /r:System.Web.dll /r:System.Data.dll Datalayer.vb
```

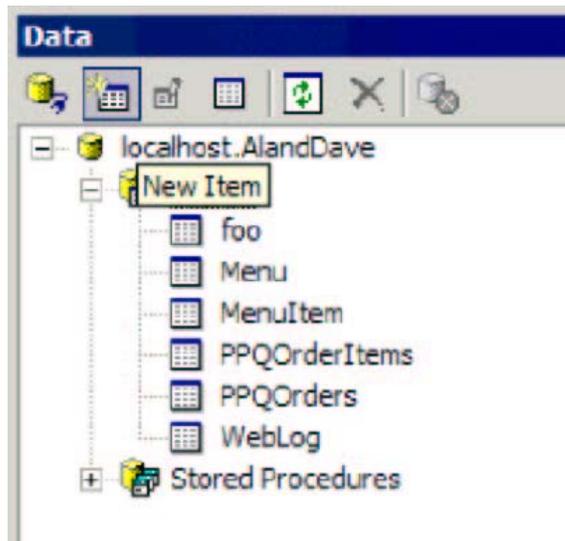
Después podemos ejecutar este archivo de lote desde la línea de comando. Algo así podría ser realmente un complemento bastante bueno.

Manejar los datos

Analizamos las funciones de administración de datos de Web Matrix en la *Sección 1*, pero ahora pondremos en práctica estas funciones. Primero, usaremos la pestaña Data para crear una conexión nueva (esto supone que tenemos una base de datos ya creada):



Una vez conectados a la base de datos, podemos crear una tabla nueva al seleccionar Tables y luego hacer clic en el botón New Item:



Si no desea pasar por este proceso, puede utilizar los scripts SQL que se proporcionan con la descarga de códigos para crear las tablas y los procedimientos almacenados.

Necesitará crear las siguientes tablas:

Table Name: PPQOrders

Columns: OrderID, Name, Address, ZipCode, OrderDate

Column Properties for OrderID:
Name: OrderID
Data Type: Int
 Required
 Primary Key
 Auto-Increment
First Value: 1
Increment: 1
Default:
Field Size: 4
Total Digits: 10
Decimal Places: 0

Buttons: New, Delete, OK, Cancel

Table Name: PPQOrderItems

Columns: fkOrderID, Item, Quantity, Cost

Column Properties for fkOrderID:
Name: fkOrderID
Data Type: Int
 Required
 Primary Key
 Auto-Increment
First Value: 0
Increment: 0
Default:
Field Size: 4
Total Digits: 10
Decimal Places: 0

Buttons: New, Delete, OK, Cancel

También necesitará crear los siguientes procedimientos almacenados:

```
CREATE PROCEDURE sp_PPQInsertOrder
@Name varchar(25),
@Address varchar(255),
@ZipCode varchar(15)
AS
INSERT INTO PPQOrders(Name, Address, ZipCode, OrderDate)
VALUES (@Name, @Address, @ZipCode, GetDate())

SELECT SCOPE_IDENTITY()
GO
```

```
CREATE PROCEDURE sp_PPQInsertOrderItem
@fkOrderID int,
@Item varchar(25),
@Quantity int,
@Cost decimal
AS
INSERT INTO PPQOrderItems
VALUES (@fkOrderID, @Item, @Quantity, @Cost)
GO
```

Aunque la ventana de procedimientos almacenados parece ser sólo un editor de textos sencillo, en realidad valida el procedimiento almacenado conforme lo guarda en SQL Server. Sin embargo, no añade los permisos requeridos (como solicitudes `GRANT` para usuarios particulares) por lo que usted deberá agregarlos. La forma exacta de estos permisos dependerá de sus detalles de conexión y de los permisos que requiera su usuario.

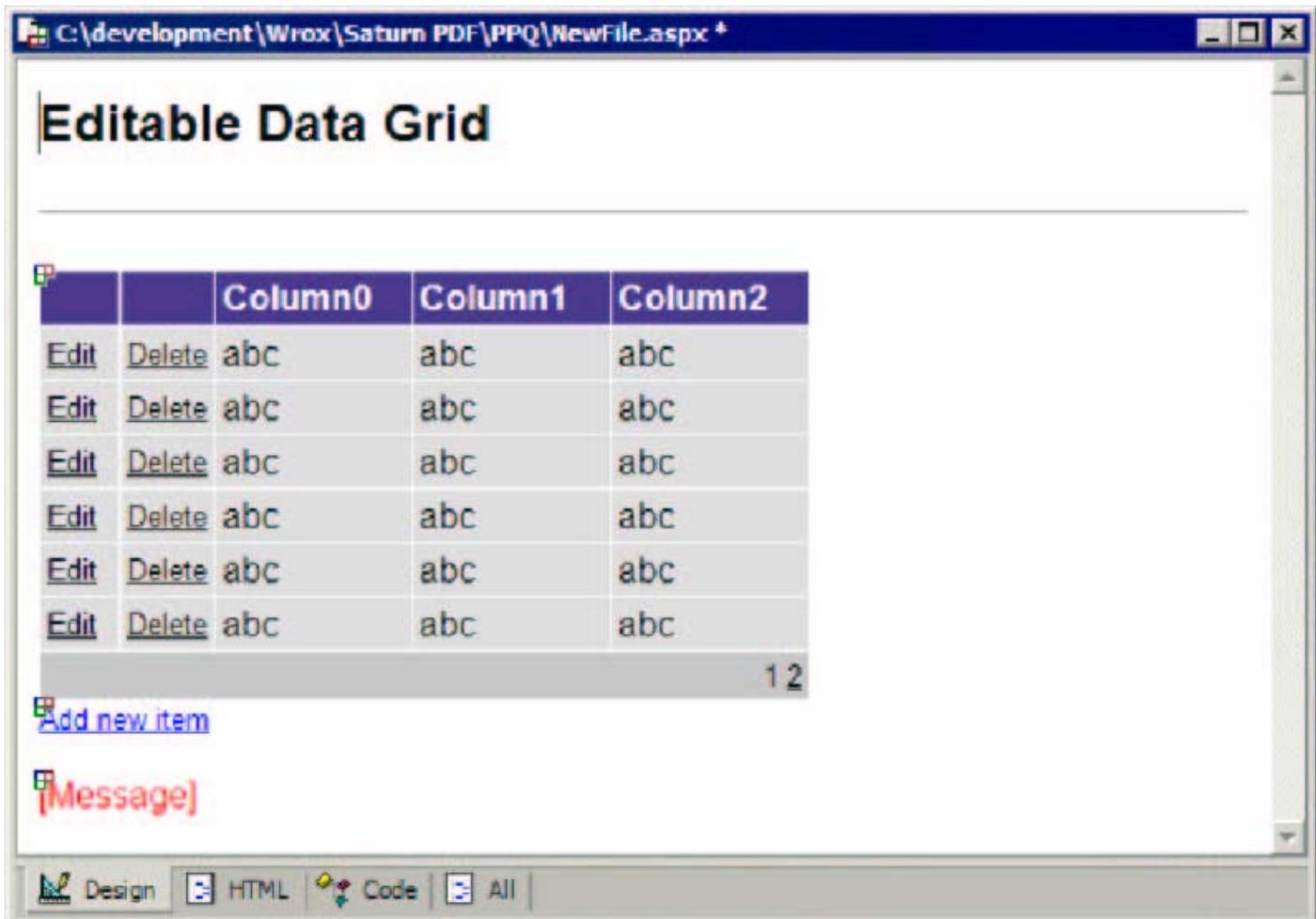
Crear Controles de usuario

Utilizaremos dos Controles de usuario en nuestra aplicación. El primer control es un banner sencillo en la parte superior de la página, que contendrá una imagen y un encabezado. El segundo control es para la canasta de compras. Usaremos un Control de usuario para la canasta de compras, pues se utilizará en varias páginas y contiene mucha funcionalidad.

Puede crear un Control de usuario al seleccionar ASP.NET User Control desde el cuadro de diálogo New Item. Nuestro primer control es muy sencillo y simplemente podemos arrastrar algunos controles HTML a la superficie de diseño. El control incluye una tabla HTML que contiene una sola fila y dos columnas, dentro de las cuales se encuentra una imagen y un texto. No es necesario agregar ningún código:



El segundo Control de usuario es nuestra canasta de compras, y ya que este control requiere cierta funcionalidad en la base de datos, no usamos la plantilla ASP.NET User Control. En cambio, utilizamos la Editable Data Grid desde la sección Data Pages del cuadro de diálogo New Item. Renombre la extensión del archivo como `.ascx`; esto es necesario, pues queremos que la canasta de compras sea una rejilla editable. Al usar la plantilla y extensión correctas, Web Matrix generará gran cantidad de código útil para nosotros. Se crea la siguiente página predeterminada:



Web Matrix creó una rejilla editable, junto con el código que nos permite editar y guardar los datos. Por ejemplo, se proporcionan los siguientes códigos para permitir actualizaciones:

```
<Columns>
  <asp:TemplateColumn>
    <ItemTemplate>
      <asp:ImageButton CommandName="Select"
        ImageURL="images/slice.gif" height="40" width="40"
        CausesValidation="false" runat="server"/>
    </ItemTemplate>
  </asp:TemplateColumn>
  <asp:TemplateColumn>
    <ItemTemplate>
      <b><asp:Label id="pizza"
        Text='<%=# DataBinder.Eval(Container.DataItem, "name") %>'
        runat="server"/>
      </b>
      <br/>
      <%=# DataBinder.Eval(Container.DataItem, "description") %><br/>
      Toppings:
      <%=# DataBinder.Eval(Container.DataItem, "ingredients") %>
    </ItemTemplate>
  </asp:TemplateColumn>
</Columns>
```

```

End If

UpdateCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11).Value = id
UpdateCommand.Parameters.Add("@au_lname", SqlDbType.VarChar, 40).Value =
    = lname
UpdateCommand.Parameters.Add("@au_fname", SqlDbType.VarChar, 20).Value =
    = fname

' execute the command
Try
    myConnection.Open()
    UpdateCommand.ExecuteNonQuery()

Catch ex as Exception
    Message.Text = ex.ToString()

Finally
    myConnection.Close()

End Try

' Resort the grid for new records
If AddingNew = True Then
    DataGrid1.CurrentPageIndex = 0
    AddingNew = false
End If

' rebind the grid
DataGrid1.EditItemIndex = -1
BindGrid()

End Sub

```

Este código utiliza la base de datos pubs como su plantilla, e incluye una cadena de conexión en la parte superior de la página. Todo lo que necesitamos hacer para utilizar este código en nuestro propio ejemplo es cambiar algunos detalles para que se ajusten a nuestra base de datos.

Tal vez prefiera sacar gran parte de este código de acceso a datos y reemplazarlo con invocaciones para una capa de datos que realice esta funcionalidad por usted. Si esto es algo que hará con mucha frecuencia, entonces vale la pena crear sus propias plantillas ajustadas a su estilo de códigos. Veremos cómo crear nuestras propias plantillas en la *Sección 3*.

Modificar el código

Es sencillo modificar el código estándar de la plantilla para que funcione con nuestros propios datos. Debemos cambiar la cadena de conexión y la declaración `SELECT` usadas para analizar los datos. Ambas se encuentran en la parte superior de la página:

```
Dim connectionString As String = "server=(local);database=pubs;Integrated Security=SSPI"  
Dim selectCommand As String = "SELECT au_id, au_lname, au_fname from Authors"
```

También debemos modificar tres manejadores de eventos:

- ❑ DataGrid_Update
- ❑ DataGrid_Delete
- ❑ AddNew_Click

Veamos los tipos de cambios que deberá realizar a estos métodos frecuentemente. Después de esto, veremos los cambios específicos que debemos efectuar para nuestra aplicación de ejemplo.

DataGrid_Update

Se deben realizar las siguientes modificaciones al método DataGrid_Update:

```
' get the edit text boxes  
Dim id As String = CType(e.Item.Cells(2).Controls(0), TextBox).Text  
Dim lname As String = CType(e.Item.Cells(3).Controls(0), TextBox).Text  
Dim fname As String = CType(e.Item.Cells(4).Controls(0), TextBox).Text
```

Este manejador de eventos se invoca cuando se actualiza una fila, y el objeto `DataGridCommandEventArgs` se pasa a los puntos de método de la fila que se está actualizando. Los datos se cambian desde las celdas a variables – necesitará modificar estas líneas para sacar sus datos de la rejilla. Las dos primeras columnas son las columnas editar y eliminar, razón por la cual los datos inician en la tercera columna (el conjunto de `Cells` se basa en ceros).

Después necesitamos configurar el texto del comando:

```
' TODO: update the Command value for your application  
Dim myConnection As New SqlConnection(connectionString)  
Dim updateCommand As SqlCommand = new SqlCommand()  
updateCommand.Connection = myConnection  
  
If AddingNew = True Then  
    updateCommand.CommandText = "INSERT INTO authors(au_id, " & _  
        "au_lname, au_fname, contract) " & _  
        "VALUES (@au_id, @au_lname, @au_fname, 0)"  
Else  
    updateCommand.CommandText = "UPDATE authors " & _  
        "SET au_lname = @au_lname, au_fname = @au_fname " & _  
        "WHERE au_id = @au_id"  
End If
```

Nuevamente, esto se debe modificar para que se ajuste a sus datos. De manera alterna, puede configurar la propiedad `CommandText` del comando, al nombre de un procedimiento almacenado, y el `CommandType` a `CommandType.StoredProcedure`.

A continuación aparecen los parámetros para el comando. Estos también se deben modificar:

```
UpdateCommand.Parameters.Add("@au_id", SqlDbType.VarChar, 11).Value = id
UpdateCommand.Parameters.Add("@au_lname", SqlDbType.VarChar, 40).Value _
    = lname
UpdateCommand.Parameters.Add("@au_fname", SqlDbType.VarChar, 20).Value _
    = fname
```

El resto del procedimiento permanece igual.

DataGrid_Delete

Para `DataGrid_Delete` todo lo que necesitamos es cambiar el comando que realiza la eliminación:

```
Dim DeleteCommand As New SqlCommand("DELETE from authors where au_id='" & _
    keyValue & "'", myConnection)
```

Igual que la actualización, esto también se puede cambiar para invocar un procedimiento almacenado, siempre y cuando `CommandType` se configure correctamente a `CommandType.StoredProcedure`.

AddNew_Click

Para las adiciones, el código crea un nuevo arreglo de datos que se inserta en la tabla:

```
' add a new blank row to the end of the data
Dim rowValues As Object() = {"", "", ""}
ds.Tables(0).Rows.Add(rowValues)
```

En este código los valores nuevos (tres de ellos) son cadenas, por lo que se utilizan cadenas vacías. Tendrá que agregar objetos del tipo correcto que se ajusten al tipo definido en las columnas de la tabla.

Modificar nuestro Control de usuario

Para nuestro control, simplemente mantenemos los detalles de la canasta de compras en una `DataTable` en `Session`, de manera que se pueda eliminar todo el código de acceso a la base de datos. Por ejemplo, la rutina de actualización de la rejilla ahora es:

```
Sub DataGrid_Update(Sender As Object, E As DataGridCommandEventArgs)

    ' get the edit text boxes
    Dim id As Integer =
```

```

CInt (ShoppingBasket.DataKeys(e.Item.ItemIndex))
    Dim qty As String = CType(e.Item.Cells(1).Controls(0), TextBox).Text

    ChangeQuantity(id, qty)

    ' rebind the grid
    ShoppingBasket.EditItemIndex = -1
    BindGrid()

End Sub

```

Las otras funciones de datos también se modifican para utilizar métodos que manipulen los elementos de la canasta de compras. Estos métodos se han hecho públicos para poderlos invocar desde la página que aloja al Control de usuario. No entraremos en detalles para todos estos métodos, ya que son bastante fáciles de comprender y no son específicos de Web Matrix.

Un aspecto que vale la pena mencionar es una propiedad que creamos para la canasta:

```

Public WriteOnly Property ViewMode As Boolean
    Set
        If Value = True Then
            btnClear.Visible = False
            Dim cols As Integer = ShoppingBasket.Columns.Count
            ShoppingBasket.Columns(cols-1).Visible = False
            ShoppingBasket.Columns(cols-2).Visible = False
        End If
    End Set
End Property

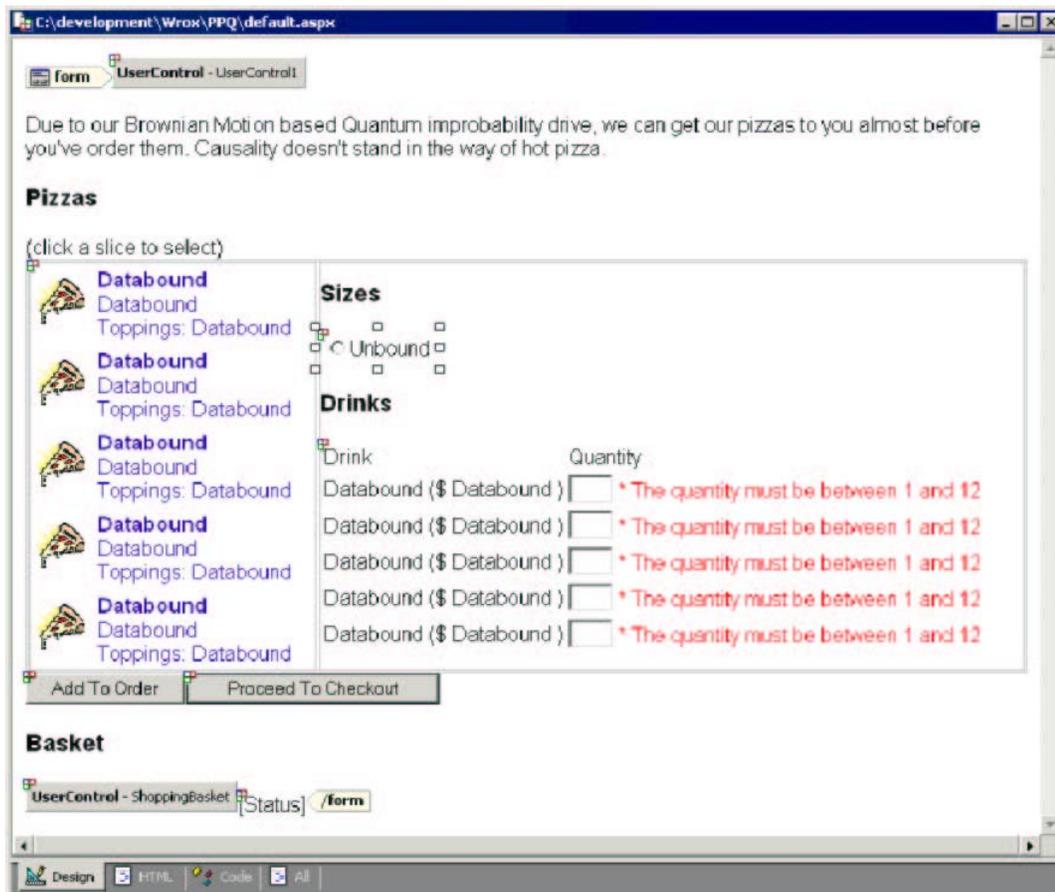
```

El configurador para esta propiedad cambia la rejilla editable a una rejilla de sólo lectura. Lo logra al hacer invisibles la columna de edición, la columna de eliminar y el botón de borrar. Nuestro Control de usuario ahora se puede utilizar en varias páginas, tanto en los modos editables como no editables.

Así, ya tenemos una canasta de compras completamente funcional, que se puede soltar en otras páginas.

La página principal

Aunque nuestra página principal predeterminada mostrará datos, las plantillas de datos proporcionadas por Web Matrix producen rejillas y nosotros deseamos una combinación de diferentes controles de datos. Por ello, empezaremos con una página ASP.NET en blanco. Una vez que agreguemos algunos controles a la página, se verá de la siguiente manera:



Aquí podemos observar una combinación de varios controles. En la parte superior de la página, tenemos el Control de usuario que creamos antes y que representa el banner. Web Matrix no proporciona soporte al tiempo de diseño para estos tipos de controles y no hay forma de arrastrarlos a la superficie de diseño. Para agregarlos a una página, tenemos que utilizar la vista All, para poder agregar tanto la directiva como el control @Register:

```
<%@ Register TagPrefix="ppq" TagName="Banner" Src="Banner.ascx" %>
```

```
<ppq:banner id="UserControl1" runat="server" ></ppq:banner>
```

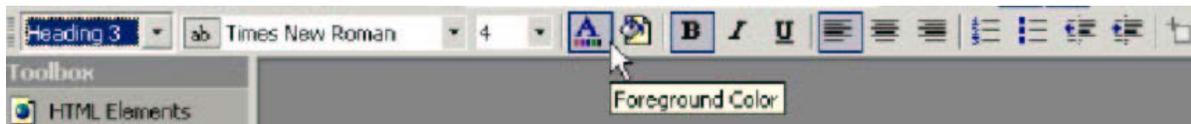
Debemos repetir el proceso para la canasta de compras:

```
<%@ Register TagPrefix="ppq" TagName="ShoppingBasket"
    Src="ShoppingBasket.ascx" %>
```

```
<ppq:ShoppingBasket id="ShoppingBasket" runat="server" >
</ppq:ShoppingBasket >
```

Si regresa a la vista Design observará que los controles de usuario aparecen como paneles grises.

Puede agregar el texto y las etiquetas a la página simplemente al arrastrarlos y soltarlos desde la Caja de herramientas. Después puede establecer sus propiedades según se requieran. Web Matrix contiene una barra de herramientas de formateo, por lo que ni siquiera tiene que recordar cuáles son los atributos de formateo:



Para los datos reales, queremos mostrarles tres aspectos:

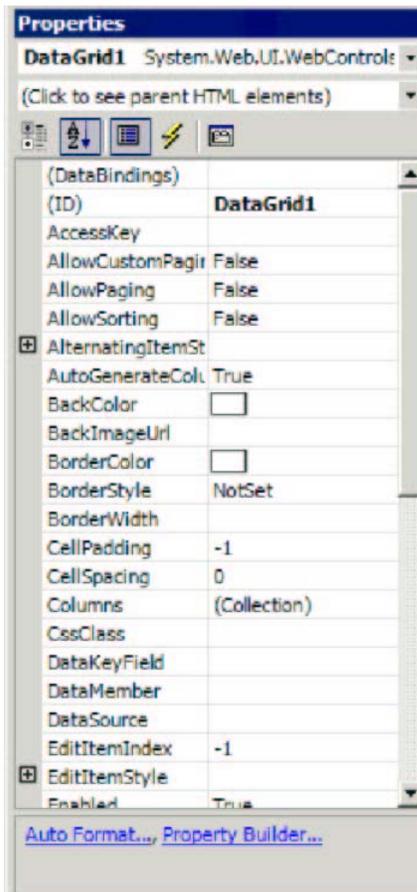
- La selección de pizzas
- Los tamaños en que se presentan
- Bebidas disponibles

Cada uno de ellos requiere un control de datos diferente: las pizzas se desplegarán utilizando una `DataGrid`; los tamaños mediante un `RadioButtonList`, y las bebidas usando un `Repeater` con contenido personalizado. Para poder formatearlos de forma agradable, los controles se incluyen dentro de una tabla HTML. Al arrastrar una tabla hacia la superficie de diseño aparece, por predeterminación, una tabla que tiene tres filas y tres columnas, por lo que debemos editar el código en la vista HTML para poder eliminar dos de las filas.

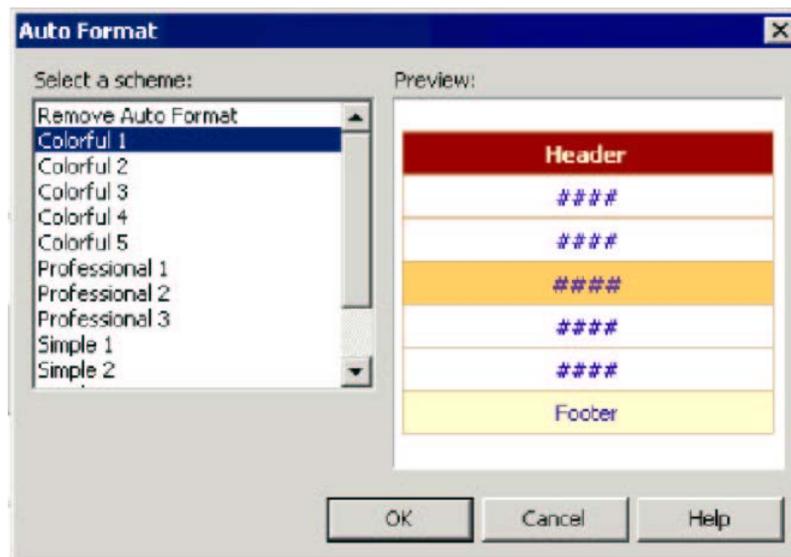
Así, para mostrar la selección de pizzas, arrastramos una `DataGrid` a la primera celda de la tabla:

A screenshot of a table with 5 rows and 3 columns. The columns are labeled 'Column0', 'Column1', and 'Column2'. The first row contains these labels. The following four rows each contain the text 'abc' in each of the three columns. The table has a border and small square handles at the corners for resizing.

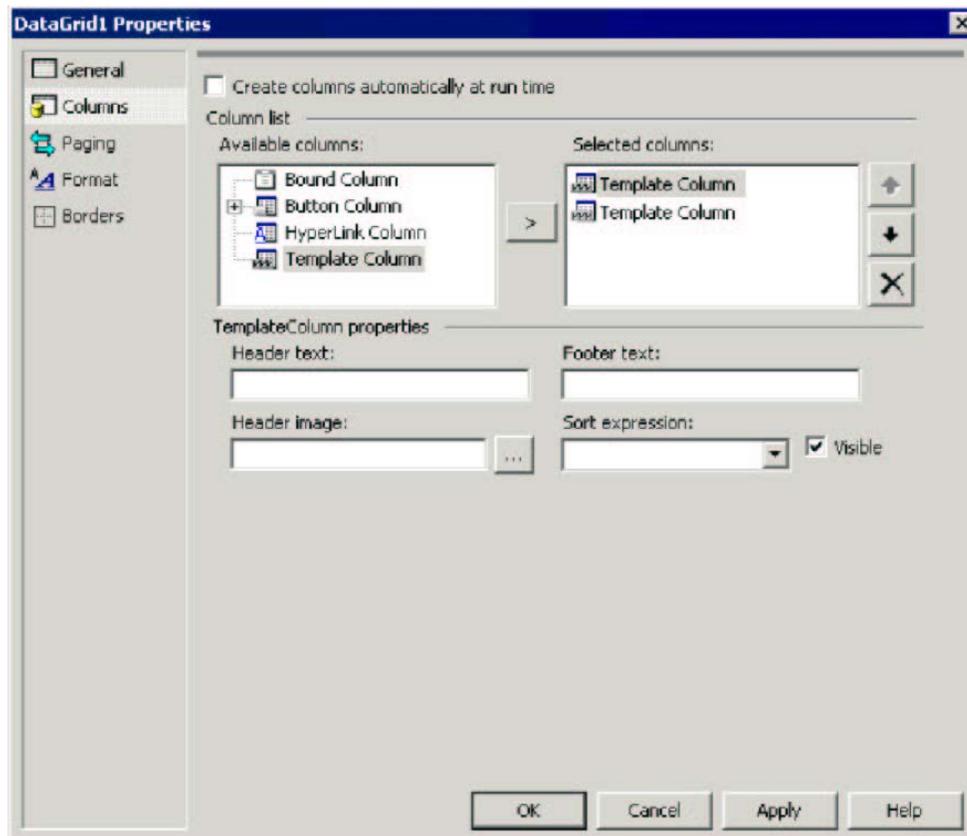
Los vínculos `Auto-Format` y `Property Builder`, localizados en la parte inferior del panel `Properties`, se pueden utilizar para formatear y personalizar la `DataGrid`:



Estos vínculos brindan acceso a los mismos diseñadores que utiliza Visual Studio .NET. El desarrollador de Auto Format nos permite elegir un estilo visual; el diseñador configurará las propiedades de estilo de la rejilla por nosotros:



El Property Builder nos permite, entre otras cosas, configurar los detalles de la columna:



Aquí, establecimos que no deseamos que las columnas se generen automáticamente y agregamos dos TemplateColumns. Al cerrar el cuadro de diálogo y cambiar a la vista HTML nos permite observar lo que esto realizó:

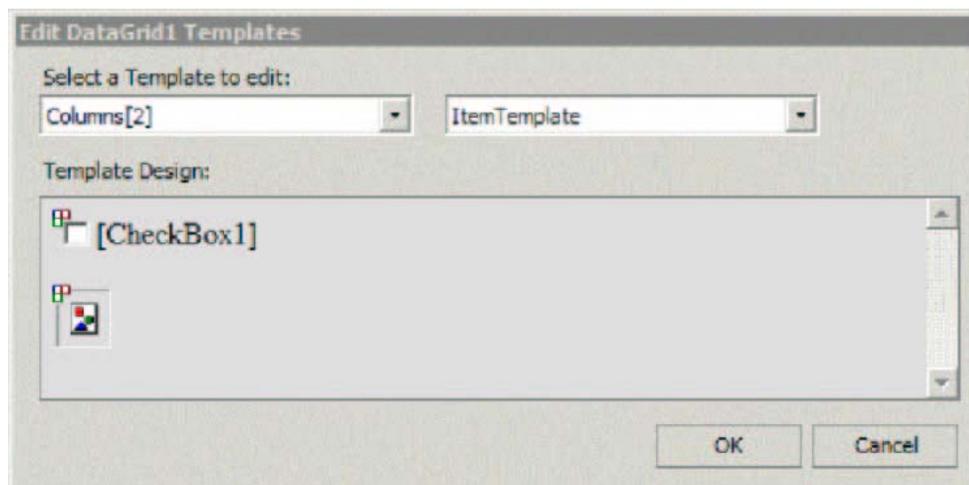
```
<asp:DataGrid id="DataGrid1" runat="server" BorderStyle="None"
    BorderWidth="1px" BorderColor="#CC9966" BackColor="White"
    CellPadding="4" AutoGenerateColumns="False">
  <FooterStyle forecolor="#330099" backcolor="#FFFFCC"></FooterStyle>
  <HeaderStyle font-bold="True" forecolor="#FFFFCC" backcolor="#990000">
</HeaderStyle>
  <PagerStyle horizontalalign="Center" forecolor="#330099"
    backcolor="#FFFFCC"></PagerStyle>
  <SelectedItemStyle font-bold="True" forecolor="#663399"
    backcolor="#FFCC66"></SelectedItemStyle>
  <ItemStyle forecolor="#330099" backcolor="White"></ItemStyle>
  <Columns>
    <asp:TemplateColumn></asp:TemplateColumn>
    <asp:TemplateColumn></asp:TemplateColumn>
  </Columns>
</asp:DataGrid>
```

Ahora podemos agregar nuestros detalles dentro de las columnas de la plantilla. Hay dos formas de hacerlo. La primera es simplemente cambiar a la vista HTML y escribir los detalles:

```
<Columns>
  <asp:TemplateColumn>
    <ItemTemplate>
      <asp:ImageButton CommandName="Select"
        ImageURL="images/slice.gif" height="40" width="40"
        CausesValidation="false" runat="server"/>
    </ItemTemplate>
  </asp:TemplateColumn>
  <asp:TemplateColumn>
    <ItemTemplate>
      <b><asp:Label id="pizza"
        Text='<# DataBinder.Eval(Container.DataItem, "name") %>'
        runat="server"/>
      </b>
      <br/>
      <# DataBinder.Eval(Container.DataItem, "description") %><br/>
      Toppings:
      <# DataBinder.Eval(Container.DataItem, "ingredients") %>
    </ItemTemplate>
  </asp:TemplateColumn>
</Columns>
```

La primera columna es un botón de imagen, que actúa como nuestro método seleccionado. La segunda columna muestra los detalles de la pizza.

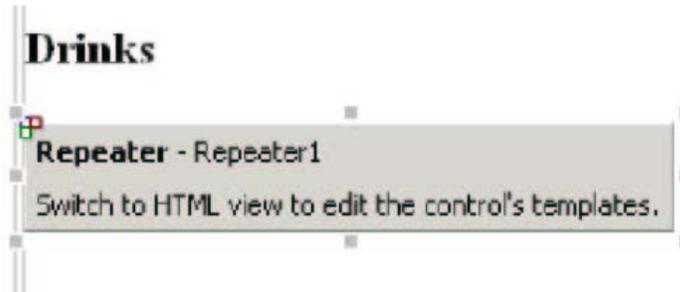
De manera alterna, puede utilizar el Template Editor, al cual se accede desde el elemento del menú Edit Template... desde el menú Edit, cuando usted ha seleccionado DataGrid en la vista Design:



El cuadro de diálogo le ofrece la opción de seleccionar la columna y la plantilla; después puede arrastrar los controles desde la Caja de herramientas hacia la superficie Template Design.

Para desplegar los tamaños de las pizzas agregaremos una `RadioButtonList` desde `Toolbox` hasta la segunda celda de la tabla. Agregaremos los datos unidos en el código, por lo que no resta nada más que hacer para este control por el momento.

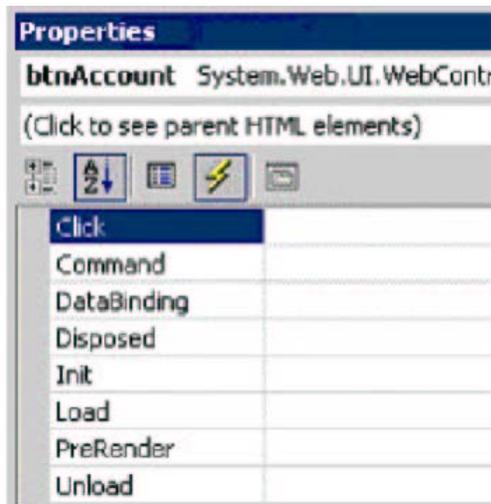
Para mostrar la selección de bebidas utilizaremos un control `Repeater`:



`Repeater` es un control sin apariencia, lo cual significa que no tiene funciones de edición de interfaz. En cambio, la apariencia se crea mediante plantillas en la vista `HTML`:

```
<asp:Repeater id="Drinks" runat="server">
  <HeaderTemplate>
    <table>
      <tr>
        <td>Drink</td>
        <td>Quantity</td>
      </tr>
    </HeaderTemplate>
  <ItemTemplate>
    <tr>
      <td>
        <asp:Label id="Drink"
          Text='<%=# DataBinder.Eval(Container.DataItem, "name") %>'
          runat="server"/>
        ($
        <asp:Label id="DrinkCost"
          Text='<%=# DataBinder.Eval(Container.DataItem, "price") %>'
          runat="server"/>
        )
      </td>
      <td>
        <asp:TextBox id="DrinkQuantity" columns="2" runat="server"/>
        <asp:RangeValidator ControlToValidate="DrinkQuantity"
          MinimumValue="1" MaximumValue="12" Type="Integer"
          ErrorMessage="* The quantity must be between 1 and 12"
          Display="Dynamic" runat="server"/>
      </td>
    </tr>
  </ItemTemplate>
  <FooterTemplate>
    </table>
  </FooterTemplate>
</asp:Repeater>
</td>
```

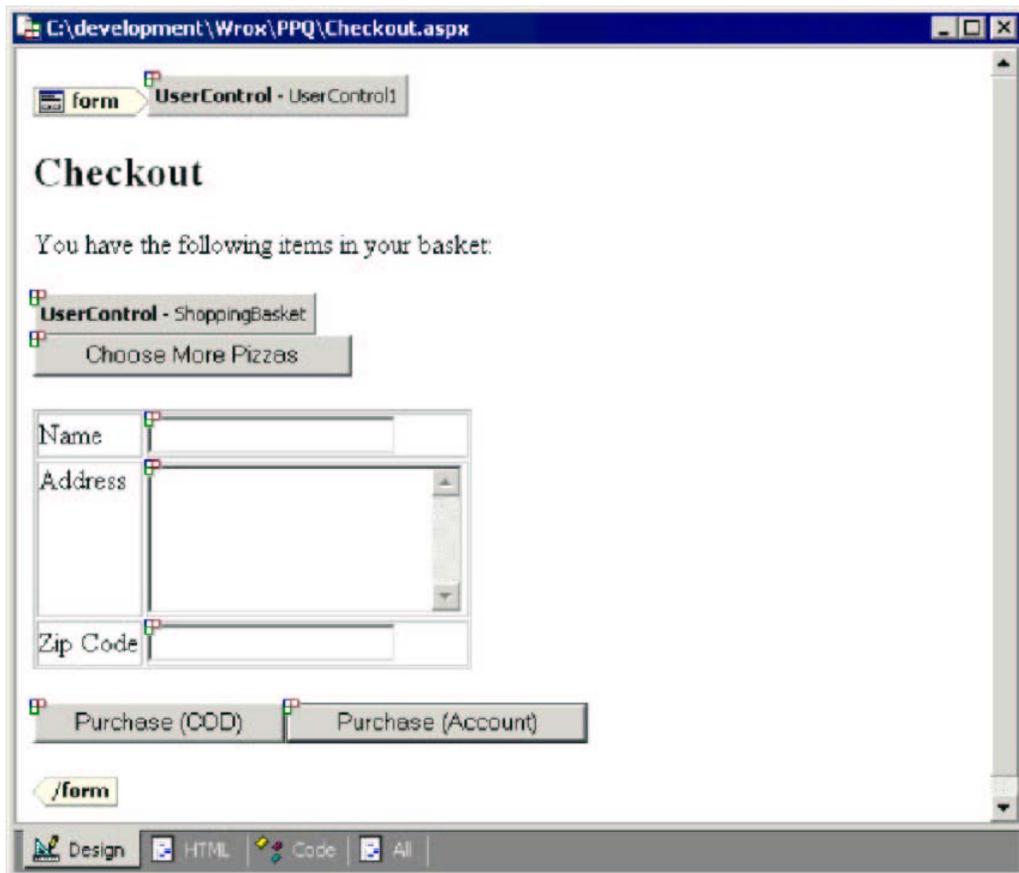
Por último, debemos agregar dos botones al formulario. El primero añade los elementos seleccionados por un cliente a la canasta de compras, mientras el segundo lleva al cliente a la página de salida. Es muy sencillo crear los manejadores de eventos para esto – ya sea hacer doble clic en el botón, lo cual nos lleva al editor de códigos, o utilizar los eventos enumerados en el panel Properties:



A partir de aquí podemos elegir los procedimientos de eventos existentes, o hacer doble clic dentro del nombre de un evento para crear el manejador de eventos y colocarlo en la ventana de eventos.

La página Checkout

La página de salida es bastante sencilla y, como hicimos con la página principal predeterminada, empezaremos con una página ASP.NET en blanco, a la cual podemos agregar controles:



Aquí usamos los mismos controles de usuario que utilizamos en nuestra página principal, además de los campos de entrada de texto para los detalles de la entrega. El cliente tiene la opción de hacer el pedido y pagar en efectivo cuando se le entregue, o conectarse a un área segura del sitio, en donde se agregará la cantidad a su cuenta.

Aquí es de particular interés observar que esta página expone tres propiedades personalizadas:

```
Public ReadOnly Property DeliveryName As String
    Get
        Return txtDeliveryName.Text
    End Get
End Property

Public ReadOnly Property DeliveryAddress As String
    Get
        Return txtDeliveryAddress.Text
    End Get
End Property

Public ReadOnly Property DeliveryZipCode As String
    Get
        Return txtDeliveryZipCode.Text
    End Get
End Property
```

Estas propiedades aseguran que la información de entrega esté disponible para la página de entrega. Debemos hacerlo de esta manera, pues las funciones de seguridad del estado de vista, evitan que se pueda colocar esta información en otra página y se puedan leer los detalles desde la forma. Podríamos transferirnos a otra página y pasar estos detalles como parte de la cadena de consulta, pero consideramos eso como una solución bastante mala. Es mucho mejor utilizar el marco proporcionado y exponer el contenido como propiedades que se pueden acceder en la siguiente página. Para que esto funcione, también debemos añadir un nombre de clase a la página actual:

```
<%@ Page Language="VB" ClassName="Checkout" %>
```

Si sabe anticipadamente que su página va a necesitar un nombre de clase, entonces lo puede configurar en el cuadro de diálogo New File.

La página Delivery

La página de entrega sirve sólo para confirmar los detalles de entrega de un pedido. En esta página se muestra la canasta de compras, y ahora configuramos la propiedad `ViewMode` que creamos antes en el Control de usuario en `True`, para que no se pueda editar la canasta:

```
Dim chk As Checkout

Sub Page_Load()

    If Not IsPostBack Then
        chk = CType(Context.Handler, Checkout)
        ShoppingBasket.ViewMode = True
    End If

End Sub
```

También hacemos referencia a la página anterior, razón por la cual le dimos un nombre de clase, para que podamos obtener los detalles de entrega simplemente al hacer referencia a las propiedades:

```
Delivery to:
<br />
<% = chk.DeliveryName %>
<br />
<% = chk.DeliveryAddress %>
<br />
<% = chk.DeliveryZipCode %>
<br />
```

Por último, vaciamos la canasta de compras:

```
Sub Page_Unload()

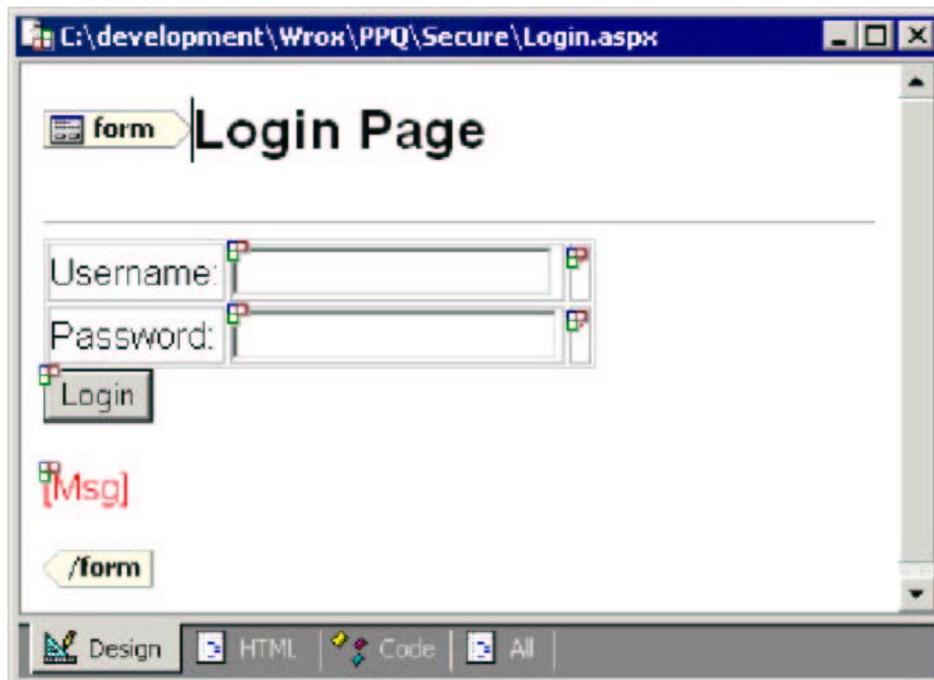
    Session("Basket") = Nothing

End Sub
```

Agregar una página de registro

La seguridad bajo ASP.NET es un tema bastante largo, y ciertamente no pretendemos cubrirlo aquí. Sin embargo, para poder agregar páginas de seguridad a la aplicación, debemos comprender que la seguridad funciona al nivel de la aplicación, no sobre una base de página por página. Esto significa que debemos crear un área segura en un directorio por separado (yo lo llamaré *secure*), y marcar este directorio como una aplicación mediante la herramienta de administración IIS.

En este nuevo directorio añadimos una página de registro, elegida entre las plantillas de Security, que están disponibles desde el cuadro de diálogo New Item:



El código que genera Web Matrix para esta página es extremadamente sencillo:

```
Sub LoginBtn_Click(Sender As Object, E As EventArgs)

    If Page.IsValid Then
        If (UserName.Text = "jdoe@somewhere.com") And _
            (UserPass.Text = "password") Then
            FormsAuthentication.RedirectFromLoginPage(UserName.Text, true)
        Else
            Msg.Text = "Invalid Credentials: Please try again"
        End If
    End If

End Sub
```

Este manejador de eventos utiliza un nombre y contraseña fijos, por lo que deberemos personalizarlos:

```
Sub LoginBtn_Click(Sender As Object, E As EventArgs)

    If Page.IsValid Then
        If FormsAuthentication.Authenticate(UserName.Text, UserPass.Text) Then
            FormsAuthentication.RedirectFromLoginPage(UserName.Text, true)
        Else
            Msg.Text = "Invalid Credentials: Please try again"
        End If
    End If

End Sub
```

Aquí sólo tomamos los detalles de la pantalla y los pasamos al método `Authenticate`. La forma en que se realiza la autenticación depende de cómo se configuró la seguridad. Esta configuración se realiza mediante el archivo de configuración de la aplicación (llamado `Web.config`), por lo cual agregamos un archivo `Web.config` a nuestra área segura, usando el que está incluido en las plantillas de `Security`.

La plantilla `Web.config` predeterminada contiene todas las secciones posibles que comentamos, por lo cual no tiene que dirigirse a la documentación para buscar los detalles. Existen varias formas para realizar la configuración, pero usaremos la más sencilla; guardaremos los nombres y las contraseñas de usuario en la sección `credentials`:

```
<authentication mode="Forms">
  <forms name="ppq" path="/" loginUrl="login.aspx"
    protection="All" timeout="30">
    <credentials passwordFormat="Clear">
      <user name="billjones" password="test" />
      <user name="marthasmith" password="test" />
      <user name="joesoap" password="test" />
    </credentials>
  </forms>
</authentication>

<authorization>
  <allow users="billjones,marthasmith,joesoap" />
  <deny users="?" />
</authorization>
```

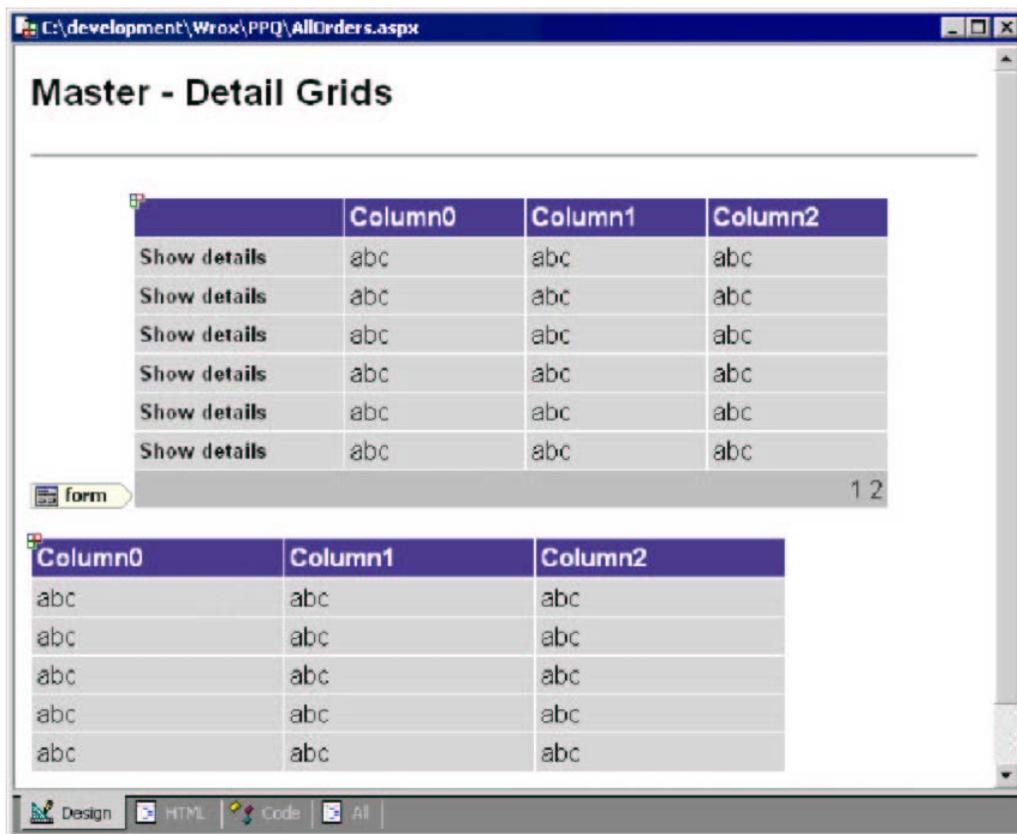
Obviamente, en un sitio real no desearía utilizar algo tan inseguro como esto (en especial no desearía almacenar las contraseñas en texto plano); sin embargo, este arreglo es suficiente para nuestro ejemplo.

Hemos especificado la autenticación de las Formas, por lo cual, si un usuario intenta acceder a cualquier página en este directorio por separado, será redireccionado a la página de registro, si no está registrado.

Cuando ejecutamos el código de registro, la clase `FormsAuthentication` usará las credenciales en el archivo de configuración para determinar si un usuario está autorizado para ver la página. Si está autorizado, entonces podemos redireccionarlo de nuevo a la página que solicitó. Así pues, realmente no tenemos que verificar para ver si el usuario está registrado - ASP.NET maneja eso por nosotros.

Master – Detail Grids

Web Matrix tiene muchas otras páginas de plantillas, que ayudan a reducir la cantidad de código que debemos escribir, aunque con frecuencia requieren un poco de personalización para que se ajusten exactamente a nuestras necesidades. Por ejemplo, considere una página que muestre todos los pedidos y los detalles de los mismos; podríamos usar la plantilla Master – Detail Grids para dicha página:



Sólo tenemos que cambiar unas cuantas líneas de código en esta página para que realice lo que deseamos. En el procedimiento `BindMasterGrid`, hay dos líneas que debemos modificar (las hemos dividido en múltiples líneas para facilitar su lectura):

```
Dim ConnectionString As String = "server=(local);database=pubs;" & _
    "Integrated Security=SSPI"
Dim CommandText As String = "select au_lname as [Last Name], " & _
    "au_fname as [First Name], Address, City, " & _
    "State from Authors order by [Last Name]"
```

Cambiamos estas líneas a:

```
Dim ConnectionString As String = "server=(local);database=AlandDave;" & _  
    " Trusted_Connection=true "  
Dim CommandText As String = "select * from PPQOrders order by OrderDate"
```

También debemos hacer una pequeña modificación a la definición DataGrid; debemos cambiar la propiedad DataKeyField de Last Name a OrderID:

```
<asp:datagrid id="MasterGrid" DataKeyField="OrderID" . . .
```

Este cambio se puede hacer ya sea en la vista HTML, o mediante la caja Properties en la vista Design.

También debemos modificar unas cuantas líneas en el procedimiento BindDetailGrid. Debemos cambiar esto:

```
' TODO: update the ConnectionString value for your application  
Dim ConnectionString As String = "server=(local);database=pubs;" & _  
    "Integrated Security=SSPI"  
  
' TODO: update the CommandText value for your application  
Dim filterValue As String = _  
    CStr(MasterGrid.DataKeys(MasterGrid.SelectedIndex)).Replace("'", "'')  
Dim CommandText As String = "select title as Title, price as Price, " & _  
    "ytd_sales as [YTD Sales] from titleview " & _  
    "where au_lname = '" & filterValue & "'"
```

A esto:

```
Dim ConnectionString As String = "server=(local);database=AlandDave;" & _  
    " Trusted_Connection=true "  
  
Dim filterValue As String = CStr(MasterGrid.DataKeys(MasterGrid.SelectedIndex))  
Dim CommandText As String = "select Item, Quantity, Cost " & _  
    "from PPQOrderItems where fkOrderID=" & filterValue
```

Eso es todo lo que debemos cambiar; ahora tenemos una plantilla Master – Detail Grid completamente funcional:

Address <http://localhost/ppq/allorders.aspx>

Master - Detail Grids

	OrderID	Name	Address	ZipCode	OrderDate
Show details	4	Dave	here	there	4/10/2002 13:03:32
Show details	5	A.N. Other	elsewhere		4/11/2002 16:18:37
Show details	6	Dave	Hungry House		4/11/2002 16:18:59
Show details	7	him	there		4/11/2002 16:19:23
Show details	8	me again			4/11/2002 16:20:09

1

Item	Quantity	Cost
Carnivore Special	1	15
Expensive beer with flavo	2	10
Three Cheese Special	1	5
Cola	1	2
Hawaiian	1	5
Cheap tasteless beer	3	5
Margherita	1	9

Aunque esta página es bastante básica, también se puede personalizar un poco más. Por ejemplo, se puede eliminar la columna `OrderID` y mostrar los totales de línea y de pedido con muy poco esfuerzo adicional.

Páginas en memoria caché

Las plantillas de Web Matrix para la Output Caching mantienen un tema familiar. La primera es la plantilla `Vary By None`, que produce el siguiente código:

```
<%@ Page Language="VB" %>
<%@ outputcache duration="10" varybyparam="none" %>

<script runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)

        TimestampCreated.Text = DateTime.Now.ToString("r")
        TimestampExpires.Text = DateTime.Now.AddSeconds(10).ToString("r")

    End Sub
</script>

<html>
<body style="font-family:arial">
    <h2>
        Output caching for 10 seconds...
    </h2>
    <hr size="1" />
    <p>
        Output Cache created:
        <asp:Label id="TimestampCreated" ForeColor="red" Font-
```

```

    Bold="true"
        runat="server"></asp:Label>
    <br />
    Output Cache expires:
    <asp:Label id="TimestampExpires" ForeColor="red" Font-Bold="true"
        runat="server"></asp:Label>
</p>
</body>
</html>

```

Esta plantilla contiene dos etiquetas que muestran la hora actual y la hora en que expira la memoria caché. La hora de la memoria caché se configura mediante la directiva `outputcache` en la parte superior de la página:

```
<%@ outputcache duration="10" varybyparam="none" %>
```

El valor del atributo `duration` especifica el tiempo en segundos para el cual esta página se debe incluir en la memoria caché, y el valor del atributo `varybynone` indica que no hay parámetros que se puedan usar para definir cuándo se debe incluir la página en la memoria caché.

La plantilla `Vary By Browser` es similar en contenido:

```

<%@ Page Language="VB" %>
<%@ outputcache duration="10" varybyparam="none" varybycustom="browser" %>

<script runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)

        BrowserDetails.Text = Request.Browser.Browser & " " & _
            Request.Browser.MajorVersion.ToString()
        TimestampCreated.Text = DateTime.Now.ToString("r")
        TimestampExpires.Text = DateTime.Now.AddSeconds(10).ToString("r")

    End Sub
</script>

<html>
<body style="font-family:arial">
    <h2>
        Vary Cache by Browser
    </h2>
    <hr size="1" />
    <p>
        Varying output by browser:
        <asp:Label id="BrowserDetails" ForeColor="red" Font-Bold="true"
            runat="server"></asp:Label>
        <br />
        Output Cache created:
        <asp:Label id="TimestampCreated" ForeColor="red" Font-

```

```

    Bold="true"
        runat="server"></asp:Label>
    <br />
    Output Cache expires:
    <asp:Label id="TimestampExpires" ForeColor="red" Font-Bold="true"
        runat="server"></asp:Label>
</p>
</body>
</html>

```

Están presentes las mismas etiquetas que en la plantilla **Output Caching**, junto con otra que muestra los detalles del explorador. La directiva `outputcache` también tiene el atributo `varybycustom` configurado para el explorador, por lo cual los diferentes exploradores recibirán diferentes versiones de la memoria caché de la página.

La plantilla **Vary Cache By Headers** permite que la salida se coloque en la memoria caché dependiendo de los encabezados HTTP específicos. A continuación se muestra el código generado por Web Matrix:

```

<%@ Page Language="VB" %>
<%@ outputcache duration="10" varybyparam="none"
    varybyheader="Accept-Language" %>

<script runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)

        HeaderDetails.Text = Request.Headers("Accept-Language")
        TimestampCreated.Text = DateTime.Now.ToString("r")
        TimestampExpires.Text = DateTime.Now.AddSeconds(10).ToString("r")

    End Sub
</script>

<html>
<body style="font-family:arial">
    <h2>
        Vary Cache By Headers
    </h2>
    <hr size="1" />
    <p>
        Varying output on header:
        <asp:Label id="HeaderDetails" ForeColor="red" Font-Bold="true"
            runat="server"></asp:Label>
        <br />
        Output Cache created:
        <asp:Label id="TimestampCreated" ForeColor="red" Font-Bold="true"
            runat="server"></asp:Label>
        <br />
        Output Cache expires:
        <asp:Label id="TimestampExpires" ForeColor="red" Font-

```

```

Bold="true"
runat="server"></asp:Label>
</p>
</body>
</html>

```

En lugar de mostrar los detalles de la versión del explorador, esta página muestra el encabezado Accept-Language, que es el lenguaje soportado por el explorador. La directiva `outputcache` tiene el atributo `varybyparam` configurado como ninguno, pero también el atributo `varybyheader` configurado como `Accept-Language`. Esto significa que habrá páginas de la memoria caché por separado para cada diferente lenguaje utilizado por los exploradores.

La plantilla final de la memoria caché es `Vary Cache By Parameters`, que incluye en la memoria caché conforme a los valores de la página. A continuación aparece el código generado por Web Matrix:

```

<%@ Page Language="VB" %>
<%@ outputcache duration="120" varybyparam="Category" %>

<script runat="server">
    Sub Page_Load(Sender As Object, E As EventArgs)

        TimestampCreated.Text = DateTime.Now.ToString("r")
        TimestampExpires.Text = DateTime.Now.AddSeconds(10).ToString("r")

    End Sub

    Sub Button1_Click(Sender As Object, E As EventArgs)

        CategoryItem.Text = "You selected: " & Category.SelectedItem.Text

    End Sub
</script>

<html>
<body style="font-family:arial">
    <form runat="server">
        <h2>
            Vary Cache By Parameters
        </h2>
        <asp:Label id="CategoryItem" runat="server"></asp:Label>
        <hr size="1" />
        Category:
        <asp:DropDownList id="Category" runat="server">
            <asp:ListItem value="default">-- Select Category --</asp:ListItem>
            <asp:ListItem>psychology</asp:ListItem>
            <asp:ListItem>business</asp:ListItem>
            <asp:ListItem value="Popular

```

```

Computer">popular_comp</asp:ListItem>
</asp:DropDownList>
<asp:Button ID="Button1" Text="Lookup" OnClick="Button1_Click"
runat="server"></asp:Button>
<p>
Output Cache created:
<asp:Label id="TimestampCreated" ForeColor="red" Font-Bold="true"
runat="server"></asp:Label>
<br />
Output Cache expires:
<asp:Label id="TimestampExpires" ForeColor="red" Font-Bold="true"
runat="server"></asp:Label>
</p>
</form>
</body>
</html>

```

En esta página la propiedad `varybyparam` se configura como `Category`, que es el ID de una `DropDownList`. Siempre que la página se coloque de nuevo en el servidor, el valor almacenado en la `Category` se utiliza para determinar si la página debe recibir el servicio de la memoria caché o no. Puede utilizar cualquier ID de control, o múltiples IDs de control.

Crear Servicios Web ASP.NET

Los Servicios Web proporcionan una forma de exponer la funcionalidad de programación a través del Web, y su uso en Web Matrix es tan fácil como en otros tipos de páginas ASP.NET. En Web Matrix no hay funciones avanzadas para usar los servicios Web, pero se ofrecen plantillas para facilitar su creación.

Servicios Web: Sencillo y Memoria caché

Las dos plantillas más simples son Simple y Output Caching. La plantilla Simple produce la siguiente clase:

```

<%@ WebService language="VB" class="menu" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class menu

    <WebMethod> Public Function Add(a As Integer, b As Integer) As Integer
        Return a + b
    End Function

End Class

```

Esto proporciona un método predeterminado que contiene el atributo `WebMethod`. La página generada por Web Matrix utilizando la plantilla Output Caching es similar:

```

<%@ WebService language="VB" class="menuCache" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class menu

    <WebMethod(CacheDuration:=30)> Public Function TimeStampForOutputCache()
        Return "Output Cached at: " & DateTime.Now.ToString("r")
    End Function

End Class

```

Aquí el atributo `WebMethod` también especifica la duración en la cual se debe colocar la página en la memoria caché.

Encabezados SOAP personalizados en un Servicio Web

La plantilla SOAP Headers contiene el siguiente código (utiliza un nombre de clase y un espacio de nombre especificado cuando usted crea el archivo):

```

<%@ WebService language="VB" class="menuSOAP" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization
Imports System.Web.Services.Protocols

Public Class SimpleHeader
    Inherits SoapHeader

    Public Value As String
End Class

Public Class menuSOAP
    Public soapHeader As SimpleHeader

    <WebMethod, SoapHeader("soapHeader")> _
    Public Function GetValueOfSoapHeader() _
        If (soapHeader Is Nothing)
            Return "SOAP Header is empty"
        Else
            Return soapHeader.Value
        End If
    End Function

End Class

```

El código utiliza una clase (heredada de `SoapHeader`) que ofrece los detalles personalizados para el encabezado SOAP. Un método público, `GetValueOfSoapHeader`, accede a la clase personalizada y devuelve los elementos requeridos para formar parte del encabezado.

Servicio Web basado en una clase personalizada

Los Servicios Web también se pueden basar en clases personalizadas. El código contenido en la plantilla Custom Class de Web Matrix utiliza las clases `OrderDetails` y `OrderItem` para detallar los pedidos:

```
<%@ WebService language="VB" class="menuCustom" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization
Imports System.Web.Services.Protocols

Public Class menuCustom

    <WebMethod> Public Function GetOrderDetails()
        Dim orderDetails As New OrderDetails()

        ' Set values for OrderDetails class
        orderDetails.OrderNumber = 35
        orderDetails.CompanyName = "ACME Paint"
        orderDetails.CustomerFirstName = "John"
        orderDetails.CustomerLastName = "Smith"
        orderDetails.CustomerEmail = "John.Smith@IBuySpy.com"

        ' Return an array of 2 OrderItems
        orderDetails.OrderItems As New OrderItem[1];

        orderDetails.OrderItems[0].ItemName = "Sunset Yellow"
        orderDetails.OrderItems[0].ItemId = 12
        orderDetails.OrderItems[0].ItemName = "at the beach"

        orderDetails.OrderItems[1].ItemName = "Seattle Sky Blue"
        orderDetails.OrderItems[2].ItemId = 93
        orderDetails.OrderItems[3].ItemName = "A rare shade of blue"

        Return orderDetails
    End Function

End Class

Public Class OrderDetails

    ' Serialize as an attribute named OrderId
    <XmlAttribute("OrderId")> Public OrderNumber As Integer

    ' Serialize as an attribute
    <XmlAttribute()> Public CompanyName As String

    Public CustomerFirstName As String

    Public CustomerLastName As String

    ' Serialize as an element, but change the name
    <XmlElement("email")> Public CustomerEmail As String

```

```

' Rename the array OrderItemDetail
<XmlArray("OrderItemDetail")> Public OrderItem[] OrderItems
End Class

Public Class OrderItem

' Serialize all member variables as attributes

<XmlAttribute()> Public ItemName As String

<XmlAttribute()> Public ItemId As Integer

<XmlAttribute()> Public ItemDescription As String
End Class

```

Ambas clases (`OrderDetails` y `OrderItem`) tienen variables públicas que contienen los atributos que especifican cómo se debe serializar la propiedad. El método Web de la clase personalizada `GetOrderDetails`, devuelve una instancia de la clase `OrderDetails`, que se serializa conforme a los atributos especificados.

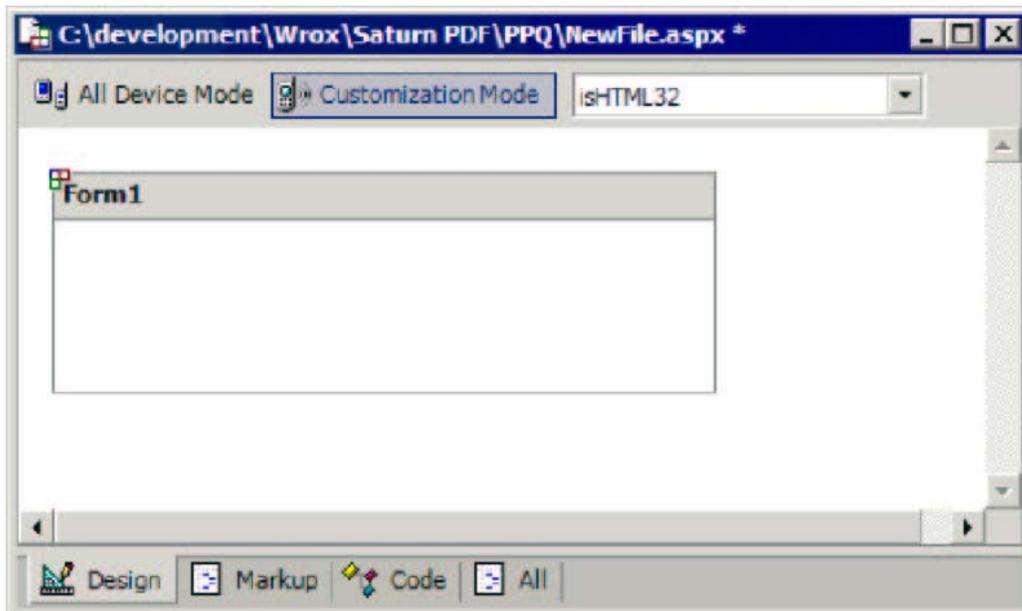
Utilizar otros componentes

Además de los controles HTML y del servidor ASP.NET estándar, existen muchos otros controles que se pueden utilizar con Web Matrix. El mercado de controles de terceros crece rápidamente y estos se pueden agregar a Web Matrix al personalizar la Caja de herramientas. Incluso puede crear sus propios controles personalizados y agregarlos a la Caja de herramientas. También existen dos conjuntos de controles adicionales proporcionados por Microsoft: los Controles del kit de herramientas de Internet móvil para crear aplicaciones para los dispositivos móviles y los Controles Web de Internet Explorer para crear contenidos amplios para IE.

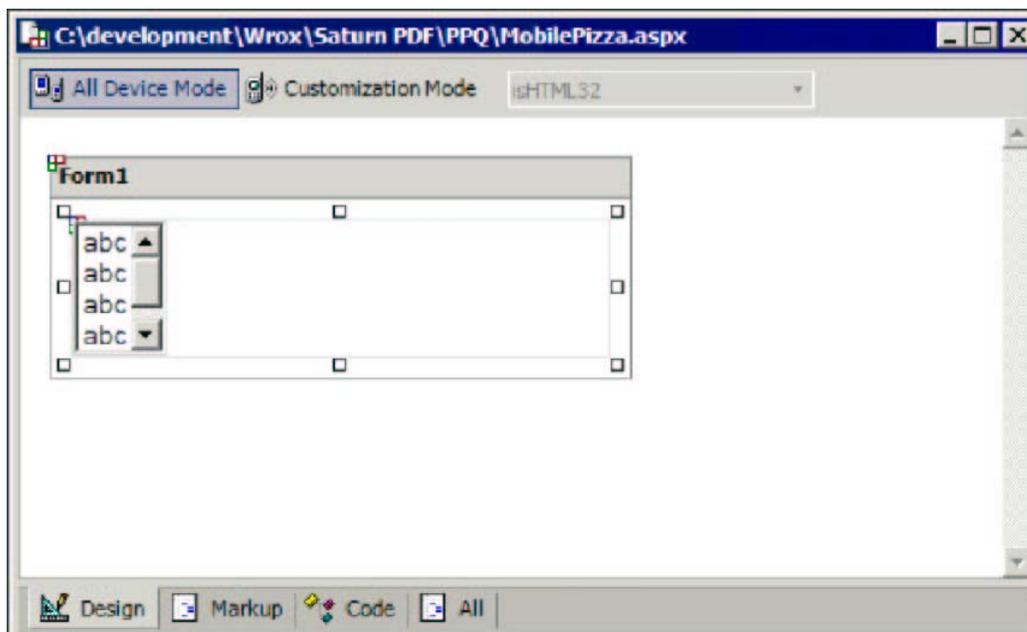
Web Matrix utiliza el mismo modelo de diseñador de controles que Visual Studio .NET, por lo cual los controles funcionarán en ambas herramientas.

Controles del kit de herramientas de Internet móvil

El soporte para los dispositivos móviles es tan bueno como para las páginas de un explorador tradicional, pero deberá instalar primero el kit de herramientas de Internet móvil. Puede elegir una Simple Mobile Page o un Simple Mobile User Control, ambos con la siguiente apariencia:



Por determinación, esta página soportará todos los dispositivos móviles, pero puede seleccionar el Customization Mode, que le permite enfocar implementaciones específicas de los dispositivos. Aunque el diseñador difiere en apariencia, su uso es exactamente el mismo que para otras páginas. Usted arrastra y suelta los controles móviles sobre la superficie de diseño, y luego añade su propio código. Por ejemplo, podemos agregar una lista de selección para las pizzas:



Aquí agregamos un control `SelectionList` a la forma y configuramos su propiedad `SelectType` como `MultiSelectListBox`, y la propiedad `DataTextField` como nombre (para el nombre de la pizza). El código a ejecutar cuando se carga la forma es:

```
Sub Form1_Load(sender As Object, e As EventArgs)

    Dim ds As DataSet = DataLayer.GetData()

    Pizzas.DataSource = ds.Tables("pizza").DefaultView
    Pizzas.DataBind()

End Sub
```

La siguiente pantalla muestra cómo se ve la página resultante en un simulador Nokia:



Este ejemplo demuestra cómo desarrollar aplicaciones móviles con Web Matrix es justo tan fácil como desarrollar aplicaciones Web estándar.

Controles Web de Internet Explorer

Los Controles Web de Internet Explorer consisten en un conjunto de controles de servidor que proporcionan rico soporte DHTML para su uso en Internet Explorer (versión 5.5 y posterior). Se accede a esto al personalizar la Caja de herramientas de Web Matrix. Después los puede utilizar como usaría cualquier otro control, al arrastrarlos hacia la superficie de diseño. De los cuatro controles incluidos en el paquete de IE, sólo TabStrip tiene soporte para el diseñador – el resto se debe personalizar, ya sea en la vista HTML o en la vista Code.

Crear y Usar un control personalizado

Se pueden crear fácilmente controles de servidor personalizados utilizando Web Matrix, ya que estos controles son sólo clases. Sin embargo, Web Matrix no soporta la creación automática de archivos de clase, o ciertas funciones avanzadas que quizá desee utilizar al crear controles personalizados, tal como la capacidad para incrustar recursos, como un mapa de bits.

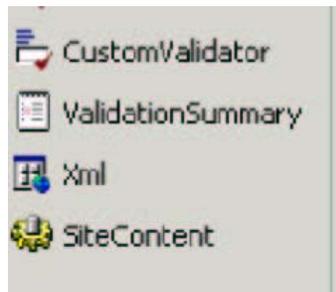
A pesar de estas limitaciones, y del hecho de que Web Matrix se diseñó principalmente para editar páginas ASP.NET, sigue siendo una buena opción para crear controles personalizados. Aquí pueden observar el control de contenidos de un sitio sencillo que acabo de crear (que soporta plantillas) al agregar una clase y utilizar el editor de códigos:

```
using System;
using System.Web;
using System.Web.UI;
using System.Collections;
using System.Web.UI.WebControls;
using System.ComponentModel;

[assembly: TagPrefix("ipona.Controls", "ipona")]
[assembly: AssemblyKeyFile("../..\\ipona.snk")]

namespace ipona.Controls
{
    [ToolboxData("<{0}:SiteContent runat=\\\"server\\\"></{0}:SiteContent>")]
    public class SiteContent : WebControl, INamingContainer
    {
        . . .
    }
}
```

Ya que este control se agregará a la barra de herramientas, el control necesita un nombre sólido, que se debe crear manualmente mediante la utilidad `sn.exe` (encontrará más información acerca de `sn.exe` en la documentación del SDK de .NET Framework). Después, podemos compilar manualmente la clase, y usar la opción `Add Local Toolbox Components`, en el menú `Tools`, para agregar este control a la Caja de herramientas:



No añadimos ningún soporte de diseñador, pero podemos arrastrar este control hacia una página y después agregar el contenido a través de la vista HTML.

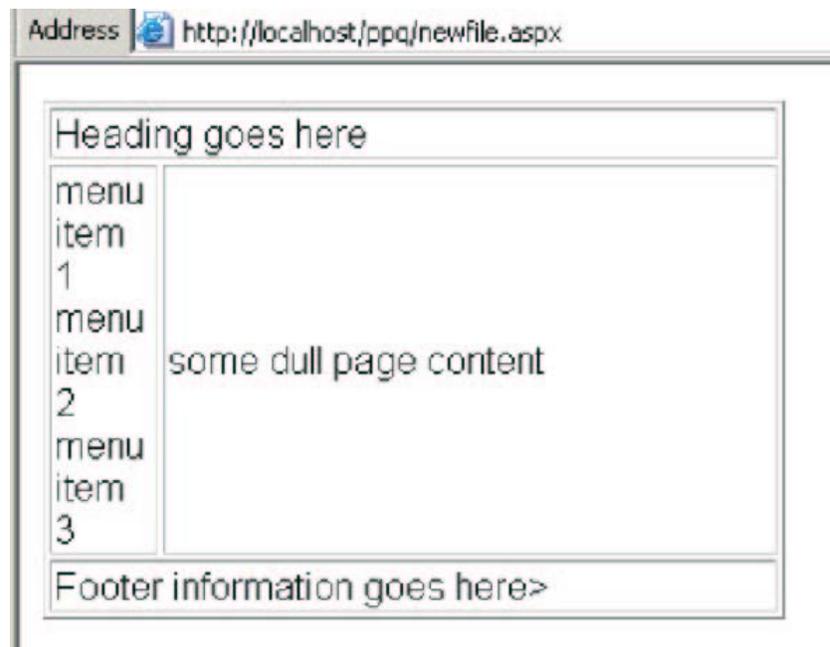
```
<ipona:SiteContent id="SiteContent1" style="WIDTH: 315px; HEIGHT: 151px"
    runat="server">
  <HeadingTemplate>
    Heading goes here
  </HeadingTemplate>

  <MenuTemplate>
    menu item 1<br/>
    menu item 2<br/>
    menu item 3<br/>
  </MenuTemplate>
</ipona:SiteContent>
```

```
<ContentTemplate>
  some dull page content
</ContentTemplate>

<FootingTemplate>
  Footer information goes here>
</FootingTemplate>
</ipona:SiteContent>
```

Si observamos la página, veremos que el control actúa justo como cualquier otro:



Parte 3 – Configurar y ampliar Web Matrix

Web Matrix se diseñó para ser ampliable y altamente configurable, lo cual le permite ser modificado para ajustarse a los requerimientos individuales. En esta sección final, veremos las diferentes formas en que podemos aprovechar esto. Cubriremos:

- ❑ Los archivos de configuración de Web Matrix
- ❑ El cuadro de diálogo Preferencias de Web Matrix
- ❑ Crear y modificar las plantillas de documento contenidas en el cuadro de diálogo New File
- ❑ Instalar y usar complementos y desarrolladores de código, que se pueden utilizar para lograr tareas específicas
- ❑ Personalizar la interfaz en general

Los archivos de configuración de Web Matrix

Así como ASP.NET, Web Matrix obtiene la mayoría de sus configuraciones de tiempo de ejecución de un archivo de configuración. Este archivo, nombrado `Web Matrix.exe.config`, se localiza en la carpeta del programa de la matriz: `\Program Files\Microsoft ASP.NET Web Matrix\version\`

Esta carpeta también incluye un archivo de nombre `ClassBrowser.exe.config`, que contiene las configuraciones de tiempo de ejecución para la herramienta Class Browser. El archivo `ClassBrowser.exe.config` es similar en formato a la sección correspondiente del archivo `Web Matrix.exe.config`.

El formato y el contenido de los archivos de configuración de Web Matrix tienden a cambiar conforme se desarrolla el producto, pero la versión actual de `Web Matrix.exe.config` contiene las siguientes secciones. Observe que “saturn” fue el primer nombre del proyecto/desarrollo para Web Matrix, y se sigue utilizando internamente:

```
<configuration>

  <configSections />           - config file sections and corresponding handlers
  <runtime />                 - the assemblies used by Web Matrix
  <appSettings />            - application-specific values for Web Matrix

  <microsoft.saturn>

    <packages />              - the packages that actually implement Web Matrix
    <projects />              - the assemblies to create each project type
    <documentTypes />        - the types of document and template available
    <toolbox />              - the sections available in the Toolbox
    <addIns />               - the Add-ins that are available
```

```

<webLinks />           - the list of Help and Community links
  <classView />         - the assemblies and folders in the Classes window
  <dataObjectMappings /> - data type conversion formats for custom objects

</microsoft.saturn>

</configuration>

```

Al editar los contenidos de estas secciones del archivo de configuración, usted puede cambiar la apariencia de Web Matrix, así como controlar los elementos que aparecerán en el IDE. Veremos algunos ejemplos más adelante en esta sección.

Observe que Web Matrix guarda en la memoria caché las configuraciones reales de tiempo de ejecución generadas del archivo de configuración dentro de un archivo por separado llamado `Web Matrix.settings`, que se localiza en su propia carpeta "perfil de usuario". Por predeterminación éste es:

```

\Documents and Settings
  \[username]
    \Application Data
      \Microsoft Corporation
        \ASP.NET Matrix Project
          \[version]
            \WebMatrix.settings

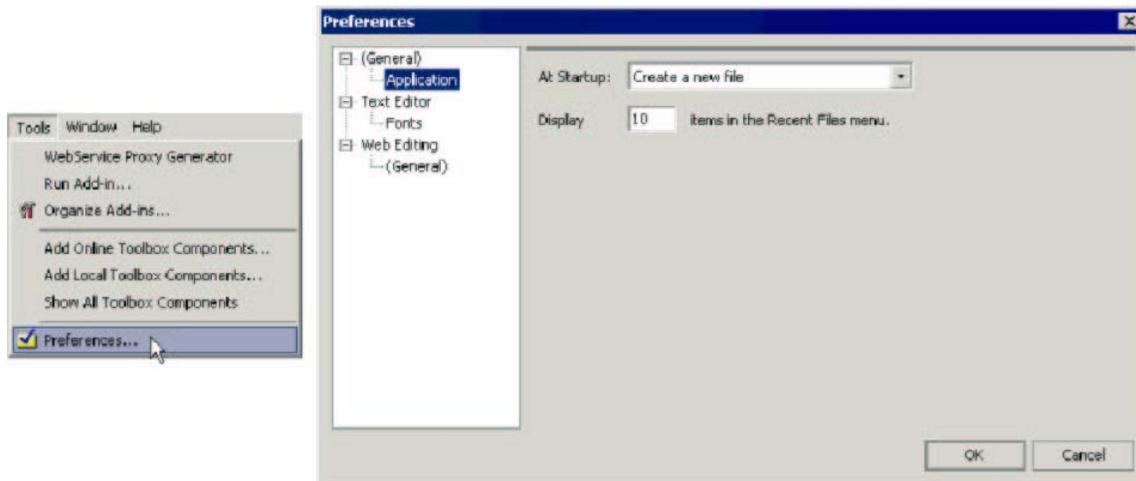
```

Este archivo se genera automáticamente cada vez que cierra Web Matrix, y después se lee cuando reinicia Web Matrix.

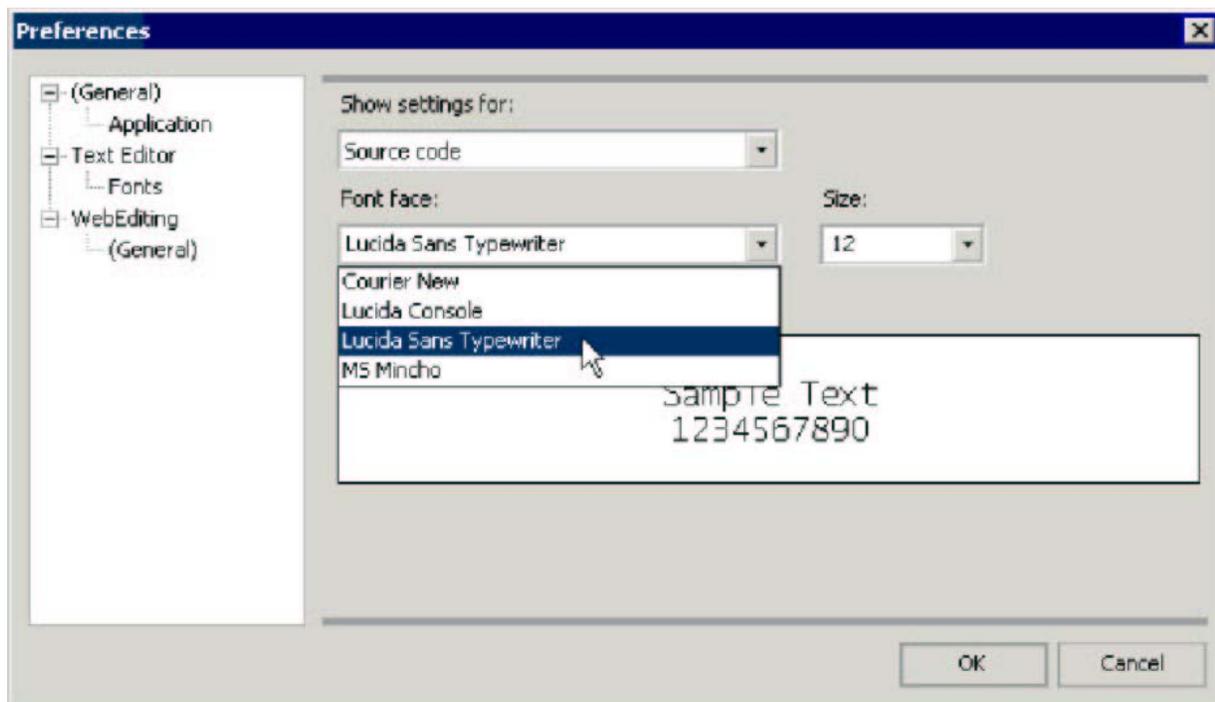
Si al iniciar Web Matrix la siguiente vez descubre que los cambios realizados al archivo `Web Matrix.exe.config` no se registraron, debe eliminar manualmente el archivo `Matrix.settings` para obligar a Web Matrix a volver a leer el archivo `Web Matrix.exe.config` y generar un nuevo archivo `Web Matrix.settings` cuando usted lo cierre nuevamente. Observe que esto eliminará cualesquiera ensamblados que haya agregado a la ventana `Classes` utilizando el cuadro de diálogo `Customize`. Para agregar permanentemente los ensamblados a la ventana `Classes`, debe editar el archivo `Web Matrix.exe.config`, como se muestra en *Personalizar la ventana Classes*.

El cuadro de diálogo Preferences

Puede acceder al cuadro de diálogo `Preferences` de Web Matrix desde el menú `Tools`. Le permite modificar varias funciones en el IDE. En la sección (`General`), bajo la única entrada actualmente disponible `Applications`, puede especificar la acción que se debe realizar cuando se inicia Web Matrix (mostrar el cuadro de diálogo `New File`, el cuadro de diálogo `Open File` o ninguno de los dos) y controlar cuántas entradas deben aparecer en la lista de archivos recientes en el menú `File`:

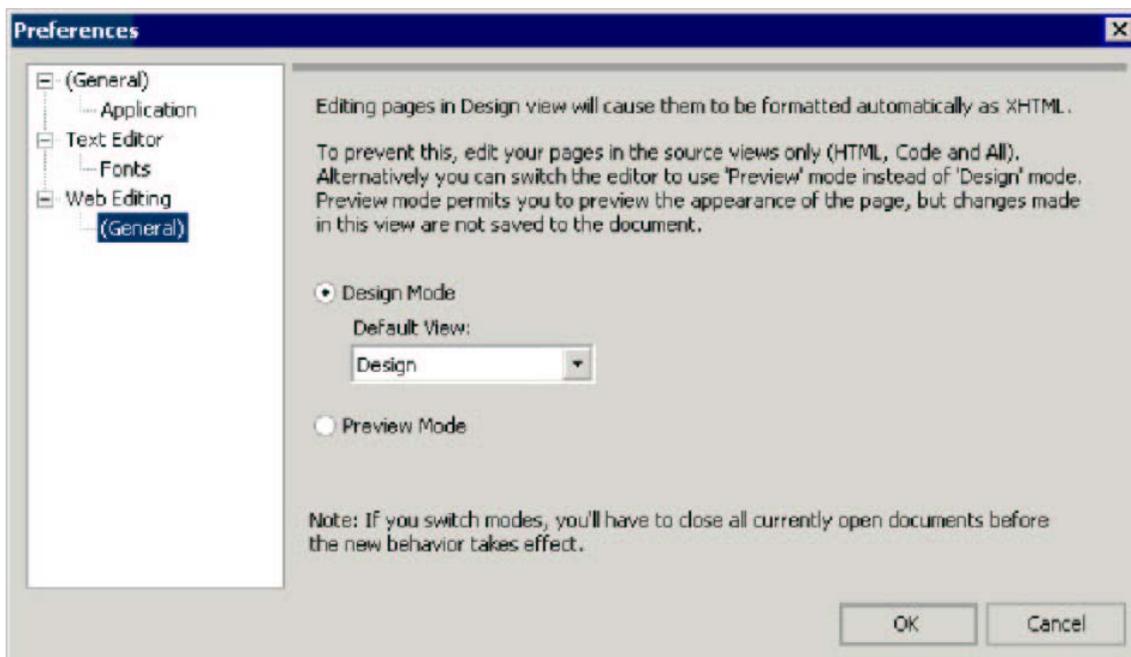


En este momento, la sección Text Editor sólo contiene la página Fonts. Aquí puede especificar la fuente que se va a utilizar en la ventana Edit, tanto en la vista Source como en la de Code, y una fuente diferente a utilizar para imprimir la fuente o el código. Sólo están disponibles algunas fuentes con ancho fijo (mono espacio), pero puede especificar el tamaño y ver previamente el resultado:



La sección Web Editing tiene una página (General) en donde puede especificar la manera en que Web Matrix debe tratar a su código cuando éste aparece en la ventana Edit. Analicemos las dos opciones – Design Mode y Preview Mode – en *La ventana Edit* y *Utilizar el modo Preview* en la *Parte 1*.

En Design Mode, puede editar páginas ASP.NET, controles de usuario y páginas HTML en la vista Design y en la vista HTML (y en la vista Code para las páginas ASP.NET y controles de usuario). Sin embargo, en este modo, Web Matrix debe reformatear el código a XHTML cuando cambia a la vista Design (para que se pueda mostrar la apariencia resultante en la ventana editar). Para evitar esto, puede cambiar de Web Matrix al Modo de vista previa, en cuyo caso la ventana Edit sólo tiene las pestañas Source y Preview para las páginas ASP.NET y los controles de usuario, y las pestañas HTML y Preview para las páginas HTML. De esa manera, el contenido sólo puede ser editado en la vista Source o HTML y al cambiarlo a la vista Preview no provoca que se tenga que volver a formatear:



La instantánea anterior también muestra una lista desplegable en la cual puede especificar cuál vista (pestaña) debe ser la predeterminada cuando utiliza Design Mode. Si así lo prefiere, puede cambiar la Default View de Design (donde la pestaña de vista Design se abre en la ventana Edit por predeterminación) a Source. Con esta configuración, la ventana Edit se abre en la vista Code, All o HTML por predeterminación – dependiendo del tipo de archivo que esté editando.

La lista de opciones disponibles actualmente sin duda alguna se ampliará conforme evolucione y se desarrolle Web Matrix durante este programa Beta.

Crear y modificar plantillas de documentos

Las plantillas de documentos se utilizan para definir los elementos que aparecen en el cuadro de diálogo New File, y contienen el contenido para las páginas que crea desde este cuadro de diálogo. Puede editar las plantillas del documento que se proporcionan con Web Matrix, y también agregar las suyas para que se ajusten a los tipos de páginas de archivo que crea regularmente. Los archivos de plantillas se almacenan en una serie de subcarpetas dentro de:

```
\Program Files
 \Microsoft ASP.NET Web Matrix
  \[version]
   \Templates\
```

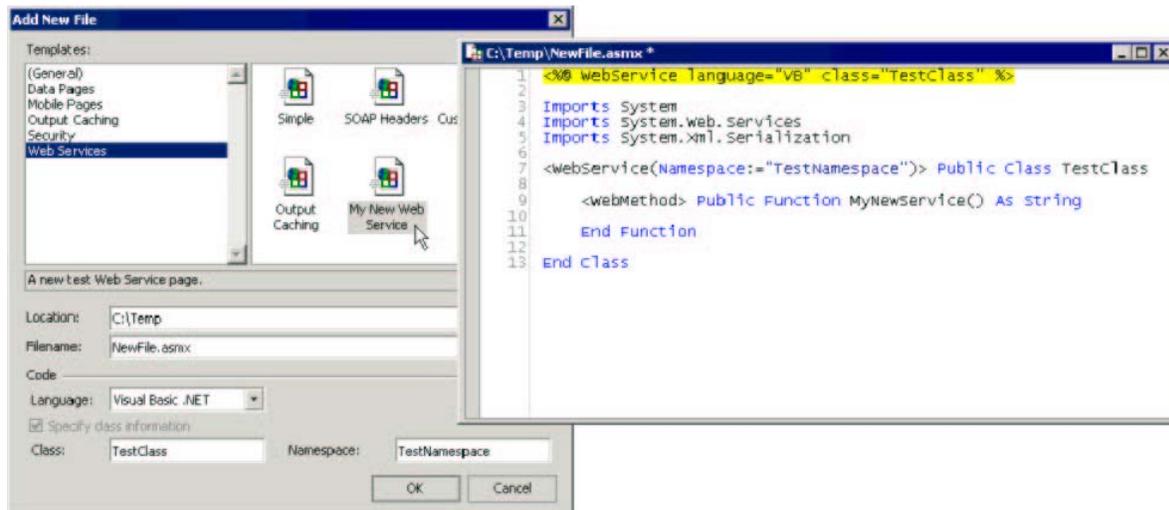
Editar plantillas de documentos directamente

En la versión actual de Web Matrix, la única manera de agregar o modificar plantillas de documentos es editar los archivos apropiados directamente, o colocar archivos nuevos en la jerarquía de la subcarpeta `Templates` de Web Matrix, y luego editar el archivo de configuración.

Agreguemos una nueva plantilla de documento a Web Matrix. Primero, vamos a crear una carpeta nueva que recibirá el nombre `My New Web Service` dentro de la carpeta `Web Services` de Web Matrix. En esta carpeta se crean dos subcarpetas, `VB` y `C#`, que corresponden a los dos lenguajes. En cada una de estas carpetas, se crea un archivo llamado `NewFile.aspx` (el nombre de archivo puede ser cualquier nombre que usted desee, aunque todas las plantillas proporcionadas utilizan el nombre `NewFile`). Luego es simplemente cuestión de agregar lo siguiente al archivo `Web Matrix.exe.config`, dentro de las `Web Services Templates` que son parte de la sección `<documentTypes>`:

```
<templateDocumentType extension="asmx" templateCategory="Web Services"
    name="My New Web Service"
    createNewDescription="A new test Web Service page."/>
```

El elemento `<templateDocumentType>` define el nombre de la subcarpeta, la extensión del archivo, la categoría que aparecerá dentro del cuadro de diálogo `New File`, y la descripción del archivo. La siguiente instantánea muestra este cuadro de diálogo, junto con el archivo que crea nuestra plantilla nueva:



Los contenidos del archivo nuevo están, por supuesto, regidos por los contenidos de la plantilla. La siguiente lista muestra la plantilla VB que colocamos en la carpeta `Web Services\My New Web Service\`:

```
<%@ WebService language="VB" class="%%ClassName%%" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization
```

```
<WebService (Namespace:="%%NamespaceName%%") > Public Class %%ClassName%%  
  
    <WebMethod> Public Function MyNewService() As String  
  
        End Function  
  
End Class
```

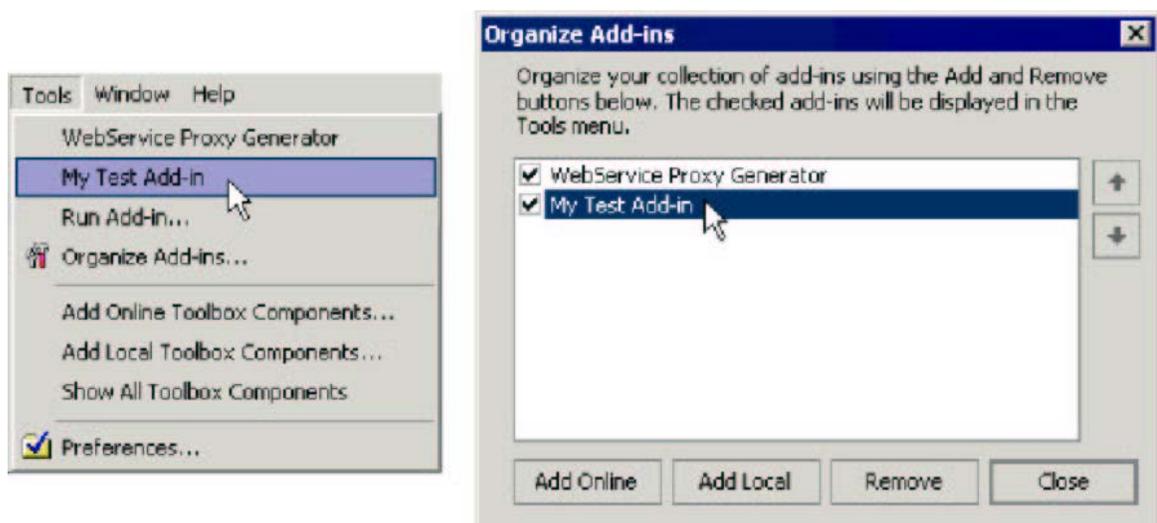
Los espaciadores para la clase y el espacio de nombre se encierran en caracteres de doble por ciento (%%). Cuando se crea el archivo nuevo a partir de la plantilla, los valores registrados en los cuadros de texto CLASS y Namespace en el cuadro de diálogo New File se sustituyen para estos espaciadores. Por supuesto, la plantilla puede contener cualquier código o contenido que usted requiera – las varias plantillas de Data Pages son buenos ejemplos de esto, y usted puede, si así lo desea, sólo editar éstos en lugar de agregar nuevos.

Instalar y utilizar complementos y desarrolladores de código

La capacidad de extensión de Web Matrix incluye Add-inns y Code Builders, lo cual significa que puede complementar los que se proporcionan por predeterminación con Web Matrix para llevar a cabo tareas específicas que realiza con regularidad.

Un complemento es un ensamble .NET que (normalmente) tiene una interfaz visual, y se puede crear en Visual Studio .NET utilizando cualquier lenguaje .NET. Crear un complemento no es una tarea trivial, pero es lo suficientemente fácil si es competente con Visual Studio .NET. Está disponible un tutorial detallado, que describe los requerimientos y las técnicas para construir e instalar un complemento a partir del sitio Web de la comunidad de Web Matrix.

Los complementos se pueden instalar utilizando el cuadro de diálogo Organize Add-inns, en donde utiliza el botón Add Local para agregar un ensamble. También puede hacer clic en Add Online e instalar cualquiera de los complementos disponibles de la página “galerías” del sitio de Web Matrix ASP.NET (que vimos cuando analizamos la parte de instalar complementos de la Caja de herramientas). Una vez instalado, el complemento nuevo será visible en el menú Tools:



Los complementos se pueden utilizar para generar código o HTML (tal vez con base en la información del usuario u otras funciones ambientales), crear o manipular archivos de disco, ejecutar utilidades de línea de comando e incluso automatizar otros programas fuera del ambiente de Web Matrix. Algunas de las cosas que desafortunadamente no puede hacer (por lo menos en la versión actual) son interactuar con el IDE de Web Matrix, o con los contenidos de las ventanas de edición. Sin embargo, puede copiar texto en el portapapeles, para que los usuarios puedan colocarlo posteriormente en una ventana de edición.

A diferencia de un complemento, un Code Builder no tiene una interfaz visual. En cambio, se encuentra en la sección Code Builders de Toolbox, y se utiliza al arrastrarlo hacia una página. Como los componentes, los Code Builders se crean como ensambles .NET, y se instalan al colocar el ensamble en la subcarpeta CodeBuilders de la carpeta principal del programa de Web Matrix y al agregar un elemento <toolboxItem> apropiado al archivo de configuración de la sección <toolbox>.

Las instrucciones detalladas para crear Add-ins y Code Builders están fuera del alcance de este documento, pero puede encontrar mayores informes y ejemplos en la página principal del Proyecto de Web Matrix ASP.NET de Microsoft en <http://www.asp.net/WebMatrix/>.

Personalizar la interfaz de Web Matrix

La ventana Classes y la herramienta separada Class Browser proporcionan una lista de clases de marcos útiles, de los cuales puede obtener información acerca de los miembros de estas clases. También existen listas con base en carpetas colapsables de las clases ASP.NET principales en la ventana Classes, que facilita encontrar rápidamente detalles de estas clases que se utilizan con regularidad.

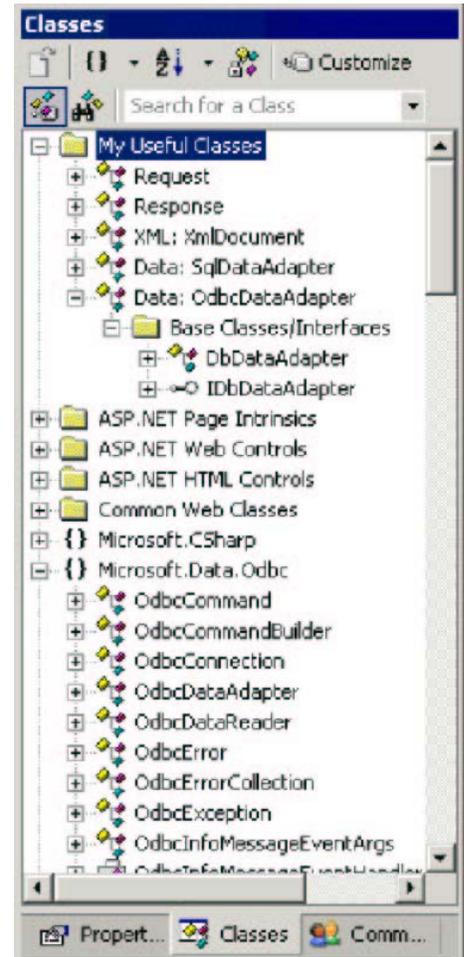
El botón Customize en la parte superior derecha de la ventana Classes se puede utilizar para agregar ensambles que ya están en la GAC (Memoria caché global del ensamble), de manera que las clases que contienen se muestran en la lista. Sin embargo, esto no modifica las carpetas colapsables de clases, y las modificaciones que realiza utilizando el cuadro de diálogo Customize se pierden si elimina el archivo WebMatrix.settings. Sin embargo, editar el archivo de configuración WebMatrix.exe.config proporciona una manera para cambiar estas listas de carpetas y crear nuevas, así como para agregar clases permanentes a la lista.

Personalizar la ventana Classes

En la instantánea puede ver que se ha agregado una nueva carpeta llamada My Useful Classes a la ventana Classes. Esta carpeta ha sido llenada con algunas clases de la Biblioteca de clases de .NET Framework. Las primeras cuatro son clases que aparecen en las otras carpetas (existentes), aunque también se ha agregado un elemento nuevo a la lista (la clase OdbcDataAdapter):

Agregar una carpeta nueva que contiene clases de los espacios de nombre existentes en la ventana Classes simplemente implica agregar un nuevo elemento `<group>` en el archivo de configuración. Para editar la lista de clases en una carpeta existente, sólo edite el contenido del elemento `<group>` apropiado.

Encuentre el elemento `<classView>` en el archivo de configuración `Web Matrix.exe.config`. Este elemento define qué clases, de las que ya están calificadas en la sección `<runtime>` del archivo, aparecerán en la ventana Classes. En la siguiente lista, puede ver los ensamblados predeterminados, el ensamblado `Microsoft.Data.Odbc` que agregamos nosotros, y el grupo personalizado que agregamos para crear la carpeta My Useful Classes (y el hecho de que utilizamos un nombre de pantalla personalizada en el atributo de nombre):



```
<classView>
  <assembly name="mscorlib" displayName="mscorlib"/>
  <assembly name="System" displayName="System"/>
  <assembly name="System.Data" displayName="System.Data"/>
  <assembly name="System.Drawing" displayName="System.Drawing"/>
  <assembly name="System.Web" displayName="System.Web"/>
  <assembly name="System.Web.Mobile" displayName="System.Web.Mobile"/>
  <assembly name="System.Web.Services" displayName="System.Web.Services"/>
  <assembly name="System.Xml" displayName="System.Xml"/>
  <assembly name="Microsoft.Data.Odbc" displayName="Microsoft.Data.Odbc"/>
  <group name="My Useful Classes">
    <groupItem name="Request" type="System.Web.HttpRequest, System.Web"/>
    <groupItem name="Response" type="System.Web.HttpResponse, System.Web"/>
    <groupItem name="XML: XmlDocument" type="System.Xml.XmlDocument, System.Xml"/>
  </group>
</classView>
```

```

type="System.Xml.XmlDocument, System.Xml"/>
  <groupItem name="Data: SqlDataAdapter"
    type="System.Data.SqlClient.SqlDataAdapter, System.Data"/>
  <groupItem name="Data: OdbcDataAdapter"
    type="Microsoft.Data.Odbc.OdbcDataAdapter, Microsoft.Data.Odbc"/>
</group>
<group name="ASP.NET Page Intrinsic">
  ... etc ...
</group>
...

```

Para agregar una clase a partir de un ensamble que todavía no se encuentra en la lista predeterminada, también tiene que agregar ese ensamble a la sección `<runtime>` del archivo de configuración. Esto también asegura que el ensamble se incluirá en la lista incluso si elimina el archivo `WebMatrix.settings` en algún momento en el futuro (por ejemplo, al instalar sus propios Add-ins o Code Builders). En la siguiente lista, puede ver el ensamble `Microsoft.Data.Odbc` que agregamos aquí para que las clases que contiene aparezcan permanentemente en la lista de la ventana `Classes`:

```

<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <probing privatePath="IDE;Packages;AddIns;Components;CodeBuilders"/>
    <qualifyAssembly partialName="mscorlib" fullName="mscorlib,
      Version=1.0.3300.0, Culture=neutral,
      PublicKeyToken=b03f5f7f11d50a3a"/>
    ... etc ...
    <qualifyAssembly partialName="Microsoft.Data.Odbc" fullName="Microsoft.Data.Odbc,
      Version=1.0.3300.0, Culture=neutral,
      PublicKeyToken=b77a5c561934e089"/>
  </assemblyBinding>
</runtime>

```

Los atributos `fullName` de estos elementos deberán estar en una sola línea, y no envueltos como lo hemos tenido que hacer en este listado. Esto también aplica al atributo `type` de los elementos `<groupItem>` que mostramos en la lista anterior.

Para obtener la información que requiere para agregar un ensamble a la lista “calificada”, puede abrir el ensamble en la herramienta `ildasm.exe` que se proporciona con el SDK de .NET Framework (en la carpeta `Framework SDK\Bin`). Una vez que se abre el ensamble, puede hacer doble clic en la entrada `MANIFEST` para ver la versión y los valores de `token` de la clave pública. El ensamble `Microsoft.Data.Odbc` que agregamos complementa al Proveedor ODB de .NET, que no se incluye en la instalación predeterminada del SDK de .NET Framework. Puede obtenerlo del sitio Web de Microsoft Data Access en <http://www.microsoft.com/data/>. Por predeterminación, instale el ensamble que complementa al proveedor ODBC en:

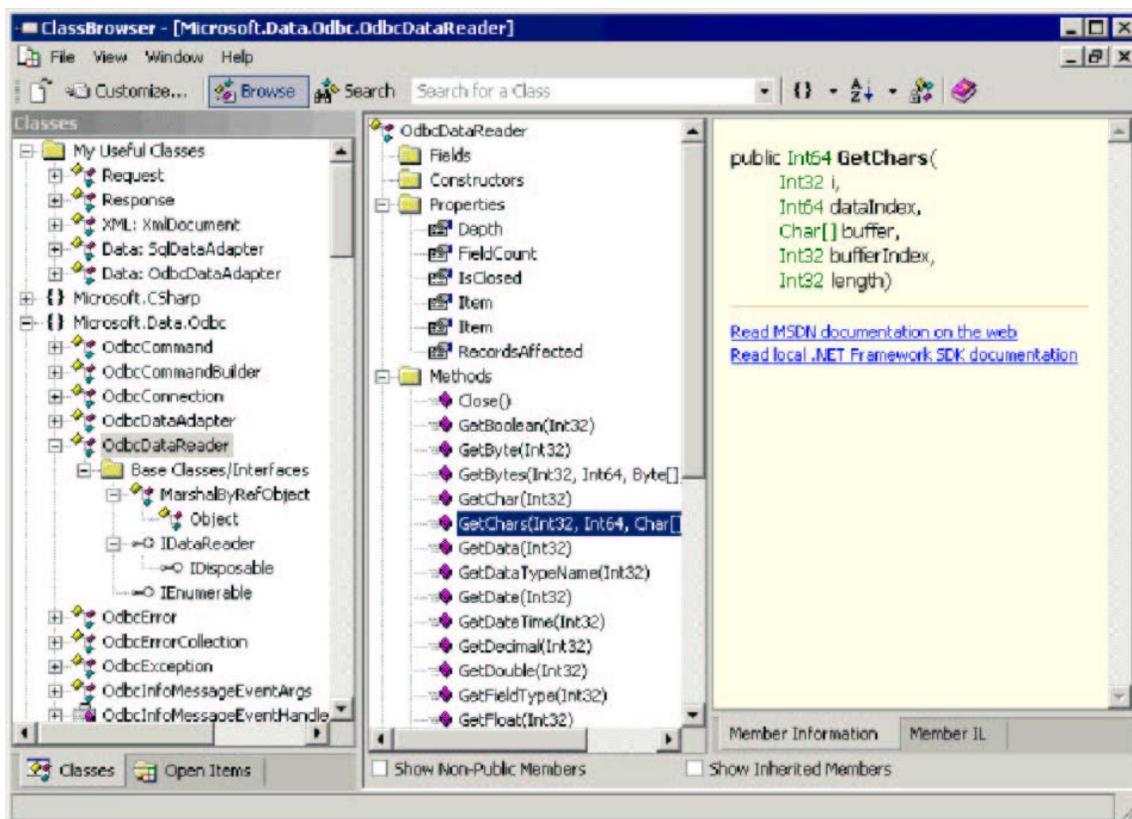
```

\Program Files
\Microsoft.NET
\Odbc.Net
\Microsoft.Data.Odbc.dll

```

Personalizar la lista de ensamble del explorador de clases

También es posible editar la lista de clases que aparece en la herramienta Class Browser, y agregar carpetas a la lista como lo hicimos para la ventana Classes en el IDE de Web Matrix. El archivo `ClassBrowser.exe.config` contiene los mismos elementos `<classView>` y `<runtime>` que el archivo de configuración de Web Matrix, por lo que la técnica es la misma. En la siguiente instantánea, puede ver que la carpeta My Useful Classes se ha agregado al Class Browser al simplemente copiarla de la sección `<classView>` del archivo `Web Matrix.exe.config`:



El Class Browser almacena sus configuraciones actuales en su carpeta `Documents y Settings` de la misma manera en que lo hace Web Matrix. Así como con Web Matrix, si no se registran los cambios al archivo `ClassBrowser.exe.config` la siguiente vez que abre el Class Browser, debe eliminar manualmente el archivo `Classbrowser.settings` para obligar que se vuelva a leer el archivo `ClassBrowser.exe.config` y la generación de un archivo nuevo `Classbrowser.settings` cuando vuelva a cerrar el Class Browser.

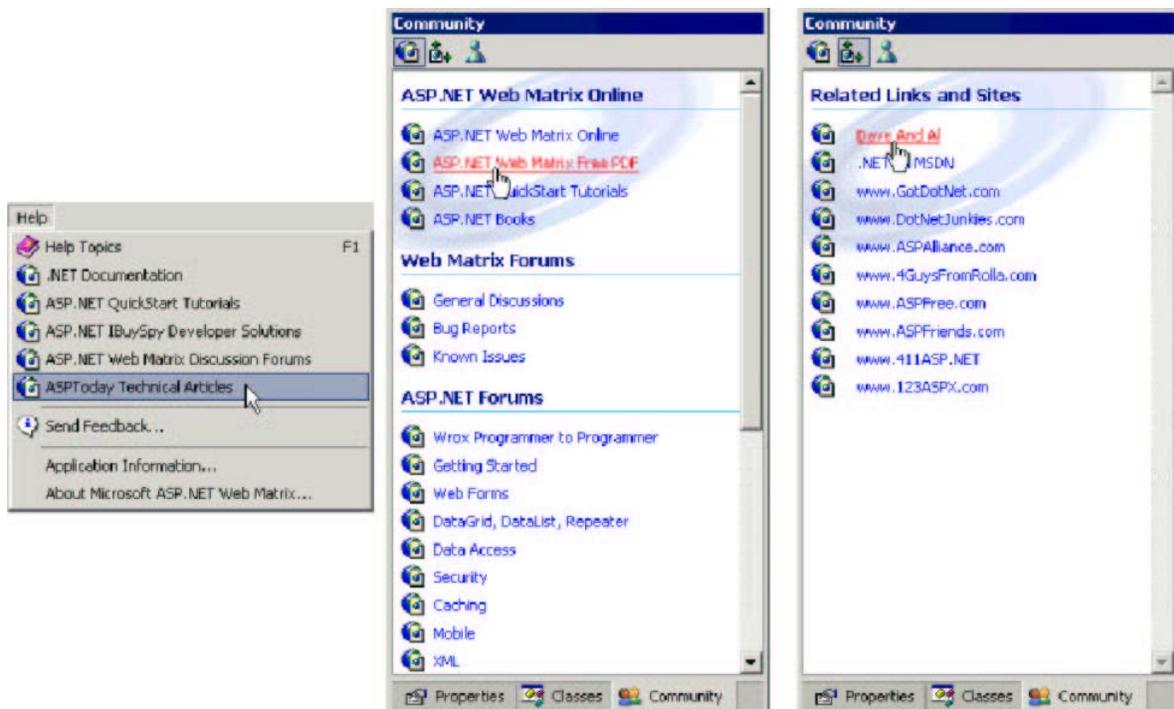
Personalizar ayuda y vínculos de recursos

Web Matrix proporciona vínculos a recursos útiles en varios lugares dentro del IDE. Existen vínculos al equipo de Web Matrix a los que puede enviar correo electrónico, vínculos para obtener ayuda en el menú Ayuda, y un grupo de vínculos en la ventana Community. Todos estos se pueden cambiar el archivo `Web Matrix.exe.config`.

La sección <WebLinks> del archivo de configuración contiene una lista de elementos <WebLink>. El atributo `name` para cada elemento <WebLink> define dónde se utiliza el vínculo. Las opciones son:

- ❑ `Help` – el vínculo aparece en el menú principal de Ayuda
- ❑ `CommunityLink` – el vínculo aparece en la sección superior de la primera página en la ventana **Community** bajo el encabezado **Web Matrix ASP.NET Online**
- ❑ `MxForumLink` – el vínculo aparece en la primera página en la ventana **Community** bajo el encabezado **Web Matrix Forums**
- ❑ `ForumLink` – el vínculo aparece en la sección superior de la primera página en la ventana **Community** bajo el encabezado **ASP.NET Forums**
- ❑ `NewsgroupLink` – el vínculo aparece bajo el encabezado **Newsgroups** en la primera página de la ventana **Community**
- ❑ `ListservLink` – el vínculo aparece bajo el encabezado **Listservs** en la primera página de la ventana **Community**
- ❑ `RelatedLink` – el vínculo aparece bajo el encabezado **Related Links and Sites** en la segunda página de la ventana **Community**

Puede ver el menú **Help** y algunos de los encabezados que se enumeran arriba en las siguientes instantáneas, que muestran que la lista predeterminada ha sido editada y se han agregado algunos de nuestros propios vínculos:



Cada nombre de tipo de vínculo (tal como `Help`) tiene un número anexo cuando se utiliza en el atributo `name` del elemento <WebLink>, por lo que los vínculos del menú **Help** que se muestran en la instantánea anterior son creados por elementos mostrados en la siguiente lista – nosotros agregamos la última:

```

<webLinks>
  <webLink name="Help0" title=".NET Documentation"
    url="http://msdn.microsoft.com/net"/>
  <webLink name="Help1" title="ASP.NET QuickStart Tutorials"
    url="http://www.asp.net/Tools/redirect.aspx?path=quickstart"/>
  <webLink name="Help2" title="ASP.NET IBuySpy Developer Solutions"
    url="http://www.asp.net/Tools/redirect.aspx?path=ibuyspy"/>
  <webLink name="Help3" title="ASP.NET Web Matrix Discussion Forums"
    url="http://www.asp.net/Forums/ShowForumGroup.aspx?tabindex=1&ForumGroupID=4"/>
  <webLink name="Help4" title="ASPToday Technical Articles"
    url="http://www.asptoday.com"/>
  ...

```

De igual forma, los vínculos adicionales en las dos páginas de la ventana Community se crean con elementos `<weblink>` adicionales, que también insertamos nosotros en la sección `<weblinks>`:

```

...
  <weblink name="CommunityLink1" title="ASP.NET Web Matrix Free PDF"
    url="http://www.asp.net/WebMatrix/download/pdf"/>
  ...
  <weblink name="ForumLink0" title="Wrox Programmer to Programmer"
    url="news://p2p.wrox.com/aspnet"/>
  ...
  <weblink name="NewsgroupLink0" title="news.wrox.com"
    url="news://news.wrox.com/aspnet"/>
  ...
  <weblink name="ListservLink0" title="ASP.NET Articles"
    url="news://ls.p2p.wrox.com/aspnetnews"/>
  ...
  <weblink name="RelatedLink0" title="Dave And Al"
    url="http://www.daveandal.com"/>
  ...
</weblinks>

```

Esto le permite insertar sus vínculos favoritos, así como cambiar el orden en el cual aparecen.

Resumen

Como puede ver, incluso desde el recorrido de inicio que proporcionamos en este documento, Web Matrix ASP.NET ya es una herramienta poderosa, intuitiva y extremadamente útil para crear sitios Web y páginas Web, Controles de usuario, Servicios Web y muchos otros tipos de archivos de proyecto relacionados con ASP.NET. Combina la facilidad de uso de las herramientas como Visual Studio .NET con la sencillez de los archivos creados con un editor de texto u otras herramientas de terceros. Al evitar el enfoque de código detrás de Visual Studio .NET, también produce páginas ASP.NET que generalmente son más fáciles de depurar, modificar y ampliar en el futuro sin requerir la herramienta de desarrollo original que se va a utilizar.

En este documento, analizamos las diferencias fundamentales entre Web Matrix y Visual Studio .NET, y después recorrimos el IDE mismo. Explicamos las funciones proporcionadas, las plantillas y asistentes que se incluyen y la manera en que puede obtener ayuda y soporte directamente desde dentro del IDE. Recuerde que Web Matrix es un “proyecto de comunidad” que evolucionará con el tiempo junto con las solicitudes de funciones y retroalimentación de los usuarios. Los vínculos que se proporcionan hacia otros recursos le ayudan a interactuar con la comunidad en general (fuera de Microsoft) que ha impulsado de manera consistente la adopción de ASP y ASP.NET.

Después pusimos Web Matrix a trabajar y creamos una aplicación de ejemplo que demuestra muchas de sus funciones. Mientras que nuestra aplicación, bajo ninguna circunstancia es una aplicación comercial del mundo real, sí nos permitió mostrar varios tipos de archivos, asistentes y otras funciones que son parte de Web Matrix. También indica cuánto tiempo y esfuerzo puede ahorrarle Web Matrix cuando trabaja con ASP.NET.

Por último, vimos cómo puede modificar y ampliar Web Matrix para ajustarla a sus propios requerimientos. Esto incluye preferencias de configuración, así como modificar el contenido y el diseño de la interfaz. También le mencionamos cómo puede crear sus propios componentes y otras herramientas que se integran directamente con Web Matrix. Habrá un número cada vez mayor de herramientas ya creadas y componentes disponibles conforme madure Web Matrix y crezca la comunidad de los usuarios. El enfoque para todo esto se encuentra en el sitio de Web Matrix ASP.NET de Microsoft en <http://www.asp.net/WebMatrix>.

Tenga en mente que este documento toma en cuenta la versión Beta 1 de Web Matrix, liberada en verano de 2002. No existe un programa fijo para la liberación “final” del producto en este momento, de hecho tal vez nunca exista una versión “final”. Sólo usted, como miembro de la comunidad de usuarios, puede ayudar a influir y decidir la fase última de Web Matrix. Úsela y disfrútela...



Más recursos para el programador ASP de WROX

Libros

Existe una gran cantidad de libros de ASP y ASP.NET impresos de Wrox Press ~ visite <http://www.wrox.com> para resúmenes detallados de los libros y descargas de códigos muestra. Nuestros títulos más recientes les proporciona aspectos invaluable a los profesionales y a los novatos en programación de Active Server Page:

En línea

ASP Today
www.asptoday.com

Todos los días, miles de desarrolladores profesionales de ASP tratan de salir de la encrucijada al utilizar el código más respetado y el recurso de métodos en el Web:

www.ASPToday.com

Este sitio "Today" se ha mantenido sólido por 2 años y está lleno de artículos técnicos, casos de estudio, códigos y asesorías prácticas. ASPToday.com es el recurso de opción profesional, recomendado por anillos Web, grupos de usuarios, Microsoft, compañías de capacitación y programadores que trabajan.

**Entre y obtenga un artículo o dos gratis... hay uno nuevo
¡TODOS LOS DÍAS HÁBILES!**

Cuando se convenza de que ASP Today cuenta con el detalle para ayudarle en su carrera – aproveche una OFERTA ESPECIAL disponible para las personas que descargan Web Matrix de:

**20% de descuento en la suscripción anual a ASPToday
(US\$79 en lugar de US\$99 estándar)**

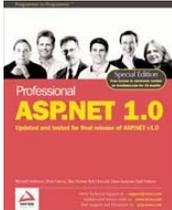


www.ASPToday.com

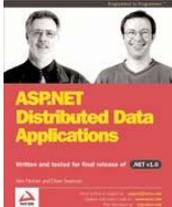
Obtener su descuento

Para obtener su descuento por suscripción anual, necesita primero:

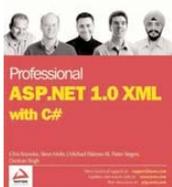
- 1) **Registrar** una dirección de correo electrónico y contraseña con la **Comunidad de desarrolladores Wrox (WDC)**. Los miembros de la WDC tienen acceso a contenido gratuito en los sitios Web de Wrox. Usted es miembro si se ha registrado en ASPToday, CSharpToday o Wrox.com (para el registro de descarga de códigos). Regístrese aquí <http://www.wrox.com>
- 2) **Conéctese** al hacer clic en el siguiente vínculo. Introduzca sus detalles nuevos de la WDC y luego escriba **matrix@wrox.com** en el indicador para 'referrer', siga las instrucciones en línea para comprar acceso durante 12 meses a ASPToday por US\$79.
OFERTA ESPECIAL PARA LA MATRIZ



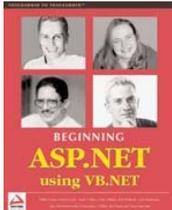
ISBN: 5644



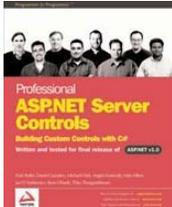
ISBN: 7450



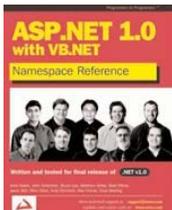
ISBN: 5040



ISBN: 7248



ISBN: 4923



ISBN: 7035

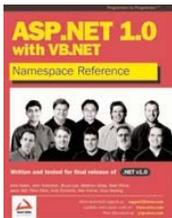
Listas para los programadores

p2p.wrox.com
The programmer's resource center

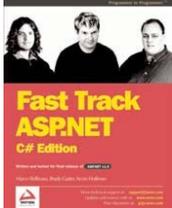
Todos los desarrolladores están invitados a unirse a las listas de análisis Wrox P2P (Programador a programador). ASP es un tema favorecido y se ha dividido en docenas de sublistas especializadas en donde los programadores y los lectores de Wrox comparten experiencias, responden a preguntas y mesas de análisis que pueden probar que son ahorradores de tiempo reales. Todas las tecnologías que cubre Wrox sirven en una lista de análisis... de ADO hasta XSLT. P2P que es su comunidad de recursos gratuita - ¡Web Matrix ya ha empezado!

Hay mucho más con Wrox que páginas... también debe verificar nuestra Biblioteca en línea, el Correo de diario de los desarrolladores y Documentos electrónicos de Amazon.

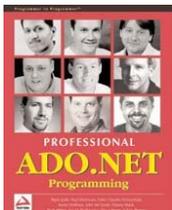
wroxbase
www.wroxbase.com



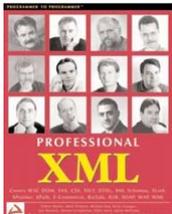
ISBN: 7450



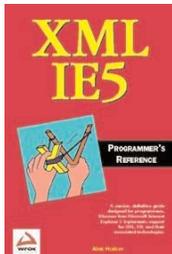
ISBN: 7191



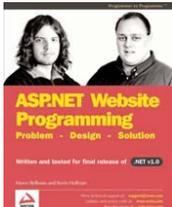
ISBN: 527X



ISBN: 3110



ISBN: 1576

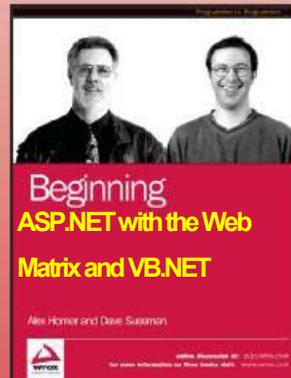


ISBN: 6934

Amazon E-Docs
Documentos electrónicos de Amazon
Nuestros casos de estudio y títulos de listas populares

Diario del desarrollador, nuestro notificador regular de correo electrónico para 100,000 programadores

Coming Soon



ISBN: 7922

