Visual C#™

Interfaces gráficas y aplicaciones para Internet con WPF, WCF y Silverlight

Visual C#™

Interfaces gráficas y aplicaciones para Internet con WPF, WCF y Silverlight

Fco. Javier Ceballos Sierra
Profesor titular de la
Escuela Politécnica Superior
Universidad de Alcalá

Ra-Ma®

Editado por:
RA-MA Editorial
C/ Jarama, 3A, Polígono Industrial Igarsa
28860 PARACUELLOS DEL JARAMA, Madrid
Teléfono: 91 658 42 80
Telefax: 91 662 81 39
Correo electrónico: editorial@ra-ma.com
Internet: **www.ra-ma.es** y **www.ra-ma.com**
ISBN: 978-84-9964-203-1
Depósito Legal: M-xxxxx-2012
Autoedición: Fco. Javier Ceballos
Filmación e impresión: Closas-Orcoyen, S.L.
Impreso en España
Primera impresión: julio 2012

En la vida hay que ser positivo y cuando
los momentos negativos acechen,
hay que seguir siendo positivo.

Dedico esta obra, con mucho cariño,
a mi nieta Isabella.

CONTENIDO

PRÓLOGO

C#, pronunciado *C Sharp*, es actualmente, junto con Java, uno de los lenguajes de programación más populares en Internet. Pero, además, está disponible para el desarrollo de aplicaciones de propósito general, aplicaciones que muestren una interfaz gráfica, aplicaciones para Internet y aplicaciones para móviles. La idea fundamental de esta obra es dar a conocer estas facetas del lenguaje C#, profundizando en el alcance que tiene sobre la Web.

De forma resumida, C# es un lenguaje orientado a objetos seguro y elegante que permite a los desarrolladores construir un amplio rango de aplicaciones seguras y robustas que se ejecutan sobre .NET Framework. Podemos utilizar C# para crear aplicaciones cliente Windows tradicionales, servicios Web XML, servicios WCF, componentes distribuidos, aplicaciones cliente servidor, aplicaciones para acceso a bases de datos, y muchas otras. *Microsoft Visual C# 2010* y superiores proporcionan un editor de código avanzado, diseñadores de interfaces de usuario apropiados, depurador integrado, y muchas otras utilidades para facilitar el desarrollo rápido de aplicaciones basadas en el lenguaje C# y en .NET Framework.

La palabra "Visual" hace referencia, desde el lado del diseño, al método que se utiliza para crear la interfaz gráfica de usuario si se dispone de la herramienta adecuada (con *Microsoft Visual Studio* se utiliza el ratón para arrastrar y colocar los objetos prefabricados en el lugar deseado dentro de un formulario) y desde el lado de la ejecución, al aspecto gráfico que toman los objetos cuando se ejecuta el código que los crea, objetos que formarán la interfaz gráfica que el usuario de la aplicación utiliza para acceder a los servicios que ésta ofrece.

La palabra "NET" hace referencia al ámbito donde operarán nuestras aplicaciones (*network* - red). C# proporciona la tecnología necesaria para saltar desde el desarrollo de aplicaciones cliente-servidor tradicionales a la siguiente generación

de aplicaciones escalables para la Web, introduciendo algunos conceptos nuevos, como ensamblados, formularios Web, servicios Web, ADO.NET y el .NET Framework.

En el momento de escribir esta obra, la especificación actual publicada sobre C# era la 4.0. Hasta este momento, la evolución de C# puede resumirse así:

- C# 1.0 (VS2002) fue la primera versión de este lenguaje de programación creado específicamente para la plataforma .NET.

- C# 2.0 (VS2005) añadió más funcionalidad: tipos de datos genéricos, tipos anulables, métodos anónimos y algunas otras delicias sintácticas.

- C# 3.0 (VS2008) añadió iniciadores de objetos, expresiones lambda, árboles de expresiones y métodos de extensión entre otras cosas, todo esto para lograr soportar LINQ.

- C# 4.0 (VS2010) aporta programación dinámica, parámetros opcionales y nombrados y mejoras en la interoperabilidad COM.

- C# 5.0 (VS2011) facilita la programación asincrónica (para ello, incluye el modificador **async** y el operador **await**) haciéndola casi tan sencilla como la síncrona, y la obtención de información de quien llamó al método, usando atributos de información del llamador (*caller info attributes*), útil para el seguimiento, la depuración y la creación de herramientas de diagnóstico.

Por otra parte, .NET es una plataforma de desarrollo compuesta básicamente por el CLR (la máquina virtual), la BCL (la biblioteca básica) y un conjunto de lenguajes de programación, entre los que se encuentra C#. .NET partió con la versión 1.0 y pasó por las 1.1, 2.0, 3.0, 3.5 y 4.0 que es la versión en el momento de escribir esta obra, la cual se incluye con Visual Studio 2010 y con Windows 7, e incluye, entre otras cosas, LINQ, C# 4.0, WPF, WCF, WF, ASP.NET 4.0, AJAX, VSTO y Silverlight. ¿Y todo esto para qué? Pues para crear aplicaciones de Internet más ricas. Se trata realmente de nuevas bibliotecas que nos permiten hacer lo que ya hacíamos con .NET 2.0 pero de una forma más vistosa, más rápida y más eficiente. Si miramos hacia atrás (Win32), teníamos la biblioteca MFC. Después vino .NET y ahora vienen otras bibliotecas que tratan de facilitar al desarrollador la realización de aplicaciones más ricas en menos tiempo.

El contenido de este libro se va a centrar básicamente en las tecnologías WPF, WCF y Silverlight.

WPF (*Windows Presentation Foundation*) es una biblioteca para el desarrollo de interfaces gráficas de usuario vectoriales avanzadas. Esta biblioteca de clases

no ha sido creada para sustituir a *Windows Forms*, sino que es otra biblioteca que facilita el desarrollo de aplicaciones de escritorio en las que estén implicados diversos tipos de medios: vídeo, documentos, contenido 3D, secuencias de imágenes animadas, etc. WPF también es idóneo si lo que se necesita es crear una interfaz de usuario con un aspecto personalizado, si hay que establecer vínculos con datos, o si desea crear una aplicación de escritorio con un estilo de navegación similar a una aplicación Web. A diferencia de *Windows Forms*, utiliza el lenguaje de marcado XAML para implementar su interfaz gráfica y los lenguajes de programación administrados, como C#, para escribir el código subyacente que implemente su comportamiento. Esta separación entre la apariencia y el comportamiento permite a los diseñadores implementar la apariencia de una aplicación al mismo tiempo que los programadores implementan su comportamiento.

WCF (*Windows Communication Foundation*) es un marco de trabajo unificado para hacer fácil la comunicación entre aplicaciones diversas en cualquier plataforma (.NET, J2EE, etc.) combinando en una sola tecnología lo que en versiones anteriores de Visual Studio existía en varias tecnologías: servicios Web ASMX, .NET Remoting, Enterprise Services y Message Queue Server. WPF está orientado al servicio. Los servicios son autónomos y comparten esquemas (datos) y contratos (funcionalidad), no clases ni tipos en general y al intercambiar mensajes no tienen que asumir nada acerca de "qué es lo que hay al otro lado del extremo". Los clientes consumen servicios y los servicios ofrecen soluciones a los clientes intercambiando mensajes, con lo que un servicio puede, a su vez, ser cliente de otro servicio.

Silverlight es una tecnología multiplataforma que se ejecuta en varios exploradores. Al igual que Flash, Silverlight permite crear contenido interactivo que se ejecuta en el cliente, con soporte para gráficos dinámicos, contenido multimedia y animación, que va mucho más allá del HTML ordinario y también, al igual que Flash, Silverlight se implementa en los navegadores mediante un plug-in. Silverlight incluye una versión reducida de .NET Framework y de WPF, lo que permite a los desarrolladores escribir código de cliente utilizando C# o Visual Basic.

Para quién es este libro

Este libro está pensado para aquellas personas que quieran aprender a desarrollar aplicaciones que muestren una interfaz gráfica al usuario, aplicaciones para acceso a bases de datos y para Internet, utilizando básicamente las bibliotecas WPF, WCF, Silverlight y el entorno de desarrollo Visual Studio 2010 o superior. Para ello, ¿qué debe hacer? Pues simplemente leer ordenadamente los capítulos del libro, resolviendo cada uno de los ejemplos que en ellos se detallan.

Evidentemente, el autor asume que el lector conoce el lenguaje *C#*, ya que este libro no describe este lenguaje debido a que este tema fue expuesto pormenorizadamente en sus otros libros *Microsoft C# - Lenguaje y aplicaciones* o *Microsoft C# - Curso de programación*, ambos editados también por RA-MA, y que tiene conocimientos orientados al desarrollo de aplicaciones utilizando la biblioteca *Windows Forms* y el entorno de desarrollo ASP.NET, temática que tampoco se expone porque fue expuesta en su otro libro *Enciclopedia de Microsoft Visual C#.*

Cómo está organizado el libro

El libro se ha estructurado en 12 capítulos más algunos apéndices que a continuación se relacionan. El capítulo 1 estudia los conceptos básicos de WPF y nos introduce en el desarrollo de una aplicación WPF. En el capítulo 2 se hace una introducción a la jerarquía de clases de WPF, al uso de los controles y eventos más frecuentes, a la validación de datos y a la personalización de la apariencia de una aplicación. El capítulo 3 nos enseña cómo añadir una barra de menús, de herramientas o de estado a una ventana WPF, cómo añadir un menú contextual y a utilizar las órdenes enrutadas. El capítulo 4 explica cómo utilizar multitud de controles WPF en el diseño de interfaces gráficas y cómo apoyar estos diseños con cajas de diálogo. En el capítulo 5 se estudia el enlace a datos, uno de los pilares de WPF, y las colecciones de objetos, ya que éstas serán los orígenes de los datos que serán proporcionados por los enlaces a los controles de la interfaz gráfica del usuario y viceversa. El capítulo 6 cubre el acceso a bases de datos utilizando ADO.NET y el desarrollo de aplicaciones basado en capas. El capítulo 7 continúa con el acceso a bases de datos, pero utilizando el lenguaje de consultas integrado LINQ, para después centrarnos en el proveedor LINQ to Entities que permite consultar las entidades que define el modelo conceptual de Entity Framework. El capítulo 8 nos enseña cómo utilizar el modelo de navegación de WPF, un modelo basado en páginas. En el capítulo 9 se estudia la tecnología Silverlight y cómo desarrollar aplicaciones para la Web o no, utilizando esta tecnología. El capítulo 10 estudia los servicios WCF y expone cómo desarrollar una aplicación Silverlight de N capas y N niveles, que tiene que acceder a una base de datos a través de servicios WCF. El capítulo 11 estudia cómo implementar la autenticación y la autorización en una aplicación Silverlight. Y el capítulo 12 estudia el acceso a datos utilizando WCF RIA Services.

CAPÍTULO 10. SERVICIOS WCF
CAPÍTULO 11. AUTENTICACIÓN Y AUTORIZACIÓN
CAPÍTULO 12. ACCESO A DATOS UTILIZANDO WCF RIA SERVICES
APÉNDICE A. ENTORNO DE DESARROLLO INTEGRADO

Qué se necesita para utilizar este libro

Este libro ha sido escrito utilizando el paquete *Microsoft .NET Framework Software Development Kit* (SDK) versión 4.0 incluido en el entorno de desarrollo *Microsoft Visual Studio 2010* (o en su defecto *Visual C# 2010 Express*, *Visual Web Developer 2010 Express* y *SQL Server 2010 Express*) que incluye todo lo necesario para escribir, construir, verificar y ejecutar aplicaciones .NET. Por lo tanto, basta con que instale en su máquina el software mencionado. Las versiones *Express*, que puede descargar desde *http://www.microsoft.com/express/*, son gratuitas.

Nota: para probar las aplicaciones Web se recomienda instalar el servidor de aplicaciones IIS (*Internet Information Server*) incluido con Windows (*Inicio > Panel de control > Agregar y quitar programas > Windows*). Esto tiene que hacerlo antes de instalar *Microsoft Visual Studio 2010* o *Visual Web Developer 2010 Express*.

Sobre los ejemplos del libro

El código fuente de todos los ejemplos del libro podrá descargarse, según se indica en los apéndices, de la Web *www.ra-ma.es* desde la página Web correspondiente al libro.

Agradecimientos

He recibido ayuda de algunas personas durante la preparación de este libro, y por ello les estoy francamente agradecido. También, deseo expresar mi agradecimiento a *Microsoft Ibérica* por poner a mi disposición, en particular, y de todos los lectores, en general, el software que el estudio de esta obra requiere.

Francisco Javier Ceballos Sierra
http://www.fjceballos.es/

APÉNDICE A

ENTORNO DE DESARROLLO INTEGRADO

Se puede desarrollar una aplicación que muestre una interfaz gráfica utilizando como herramientas *Microsoft Framework SDK* (proporciona, entre otras cosas, la biblioteca de clases .NET y el compilador de C#) y un simple editor de texto, o bien utilizando un entorno de desarrollo integrado (EDI). En el primer caso hay que escribir el código fuente línea a línea, para después, desde la línea de órdenes, compilarlo, ejecutarlo y depurarlo. Lógicamente, escribir todo el código necesario para crear la interfaz gráfica de la aplicación es una tarea repetitiva que, de poder mecanizarse, ahorraría mucho tiempo en la implementación de una aplicación y permitiría centrarse más y mejor en resolver los problemas relativos a su lógica y no a su aspecto. Justamente esto es lo nuevo que aporta *Visual Studio* o, en su defecto, las versiones de *Visual C# Express* y *Visual Web Developer Express*.

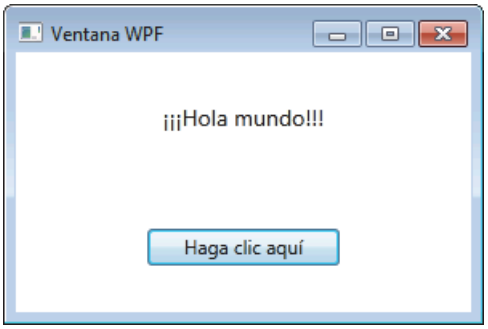
El propósito de este apéndice es mostrar cómo crear una aplicación sencilla de *Windows Presentation Foundation* (WPF) y familiarizarse con el entorno de desarrollo integrado (EDI) de *Visual Studio*/*Visual C# Express*. Al igual que las aplicaciones de formularios Windows Forms, las aplicaciones WPF se pueden diseñar arrastrando controles desde la caja de herramientas hasta el panel de diseño.

MICROSOFT VISUAL STUDIO

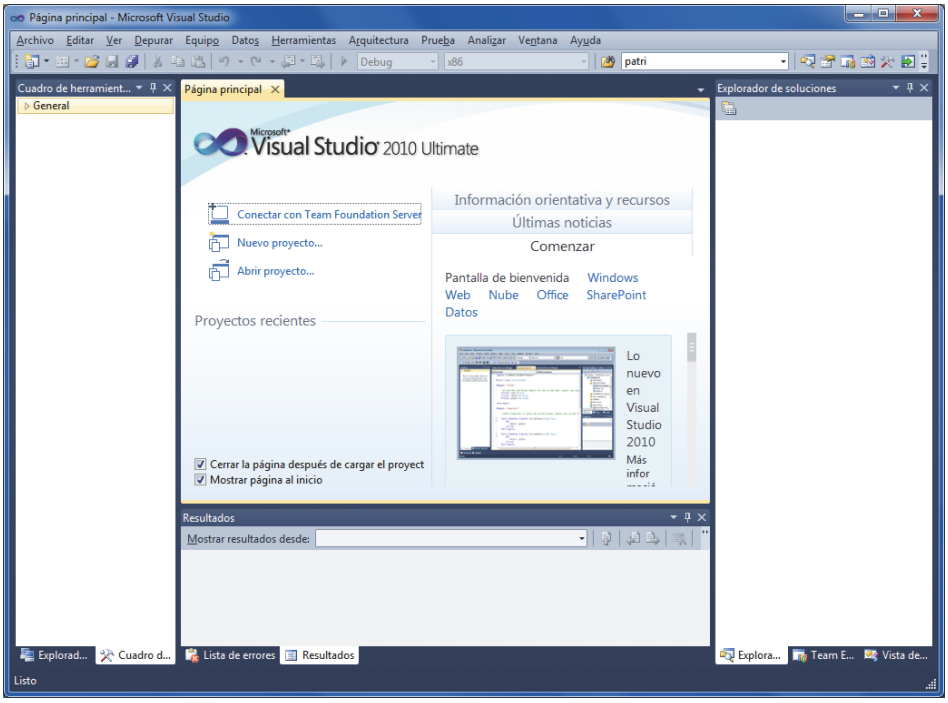
Visual Studio permite diseñar la interfaz gráfica de una aplicación WPF de manera visual, sin más que arrastrar con el ratón los controles que necesitemos sobre la ventana destino de los mismos. Unas líneas de guía o una rejilla mostrada sobre el formulario (o ventana) nos ayudarán a colocar estos controles y a darles el tamaño adecuado, y una página de propiedades nos facilitará la modificación de los valores de las propiedades de cada uno de los controles. Todo lo expuesto lo realiza-

remos sin tener que escribir ni una sola línea de código. Después, un editor de código inteligente nos ayudará a escribir el código necesario y detectará los errores sintácticos que introduzcamos, y un depurador nos ayudará a poner a punto nuestra aplicación cuando lo necesitemos.

Como ejemplo, vamos a realizar una aplicación Windows denominada *Saludo*, que presente una interfaz al usuario como la de la figura siguiente:



Para empezar, arranque Visual Studio. Se visualizará una ventana similar a la siguiente:

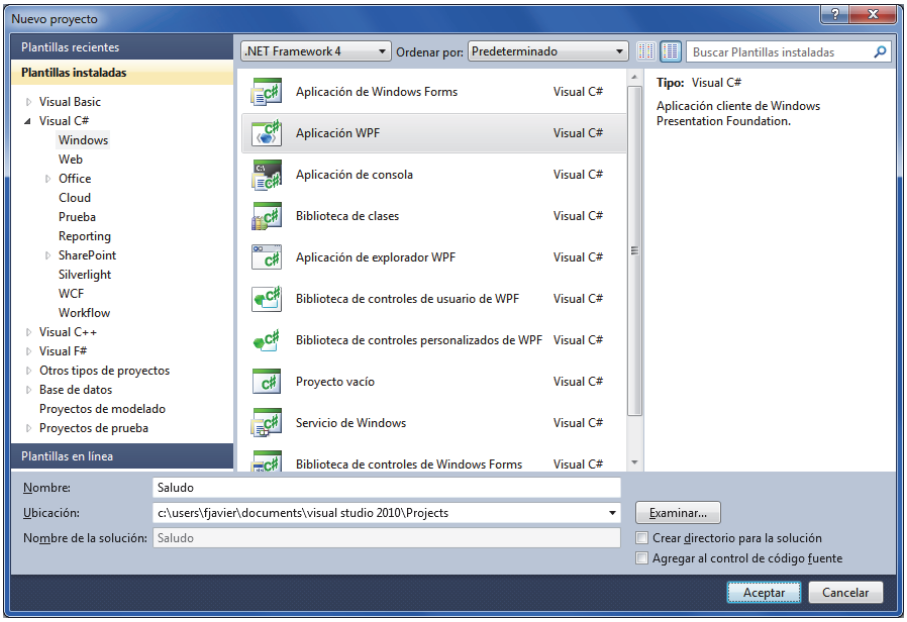


¿Cuáles son los siguientes pasos para desarrollar una aplicación WPF? En general, para construir una aplicación de este tipo con Visual Studio, siga los pasos indicados a continuación:

1. Cree un nuevo proyecto (una nueva aplicación), entendiendo por proyecto un conjunto de ficheros, normalmente distribuidos en carpetas y recursos que pueden ser compilados como una sola unidad. Visual Studio mostrará una página de diseño con un formulario vacío por omisión (una ventana).
2. Dibuje los controles sobre el formulario. Los controles serán tomados de una caja de herramientas.
3. Defina las propiedades del formulario y de los controles.
4. Escriba el código para controlar los eventos que consideremos de cada uno de los objetos.
5. Guarde, compile y ejecute la aplicación.
6. Opcionalmente, utilice un depurador para poner a punto la aplicación.

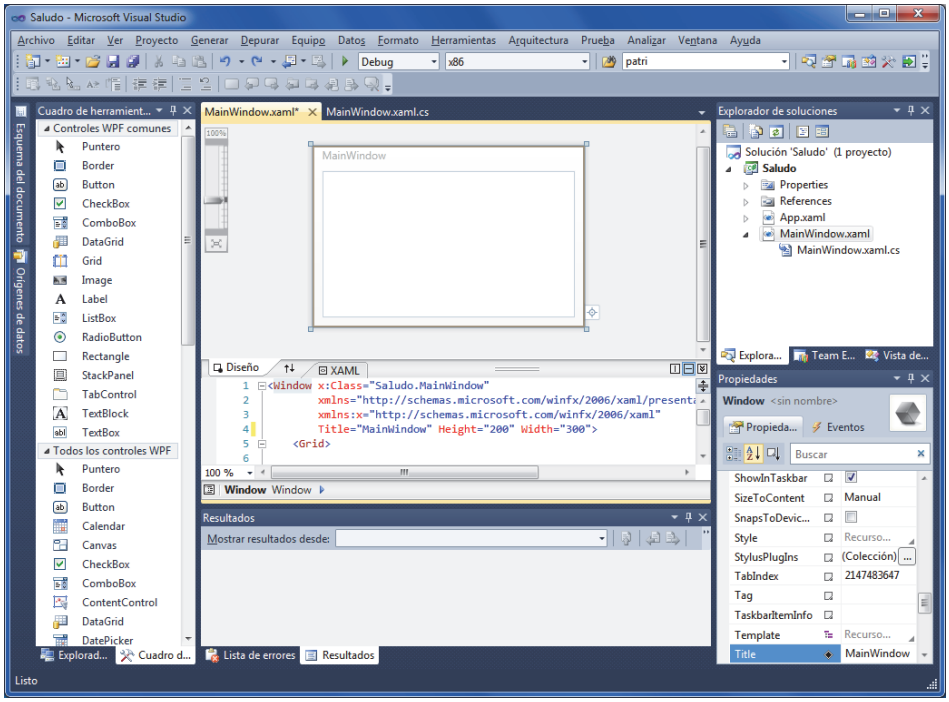
Crear un nuevo proyecto

Para crear un nuevo proyecto, diríjase a la barra de menús y ejecute *Archivo* > *Nuevo Proyecto*. En el diálogo que se visualiza, seleccione el tipo de proyecto *Visual C#* > *Windows* > *Aplicación WPF* y asígnele el nombre *Saludo*. Observe, en la parte superior de la ventana, que puede elegir la versión de *.NET Framework*. Después, para continuar, haga clic en el botón *Aceptar*:



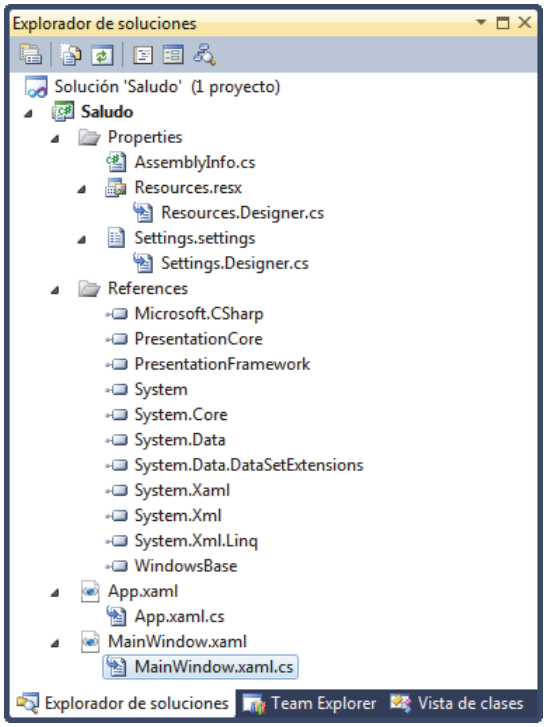
Obsérvese que se ha elegido la carpeta donde se almacenará el proyecto. Esta tarea puede posponerse sin que afecte al desarrollo de la aplicación. Ahora bien, si desea que esta tarea se realice automáticamente en el momento de crear el proyecto, caso del autor, ejecute *Herramientas > Opciones*, seleccione la opción *Proyectos y Soluciones* y marque la casilla *Guardar los proyectos nuevos al crearlos*.

Después de crear una nueva aplicación Windows, el entorno de desarrollo Visual Studio mostrará un formulario, *MainWindow*, en el diseñador. También pondrá a nuestra disposición una caja de herramientas con una gran cantidad de controles WPF listos para ser incluidos en la ventana.



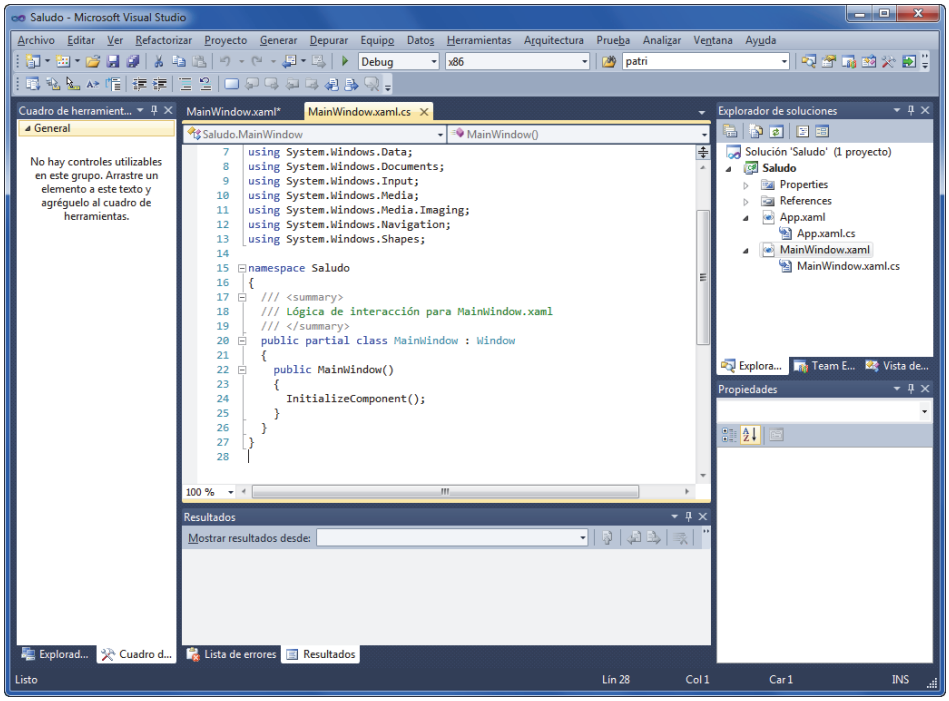
Otra característica interesante de este entorno de desarrollo es la ayuda que facilita. Basta con colocar el punto de inserción sobre una palabra clave y pulsar la tecla *F1* para que le muestre la ayuda correspondiente a la palabra seleccionada.

En la esquina superior derecha también se localiza otra ventana con varias páginas: explorador de soluciones, vista de clases, etc.; en la figura siguiente vemos el *Explorador de soluciones*:



El *Explorador de soluciones* muestra el nombre de la solución (una solución engloba uno o más proyectos), el nombre del proyecto (un proyecto administra los ficheros que componen la aplicación) y el de todos los formularios y demás módulos que intervienen en la aplicación; en nuestro caso, observamos un formulario, denominado *MainWindow*, descrito por los ficheros de código *MainWindow.xaml* y *MainWindow.xaml.cs*; el primero es el utilizado por el diseñador de formularios y el segundo, el utilizado por el programador para escribir el código. También se observa un nodo *References* que agrupa las referencias a las bibliotecas de clases de objetos que utilizará la aplicación en curso; podemos añadir nuevas referencias a otras bibliotecas haciendo clic con el botón secundario del ratón sobre ese nodo o bien eliminarlas.

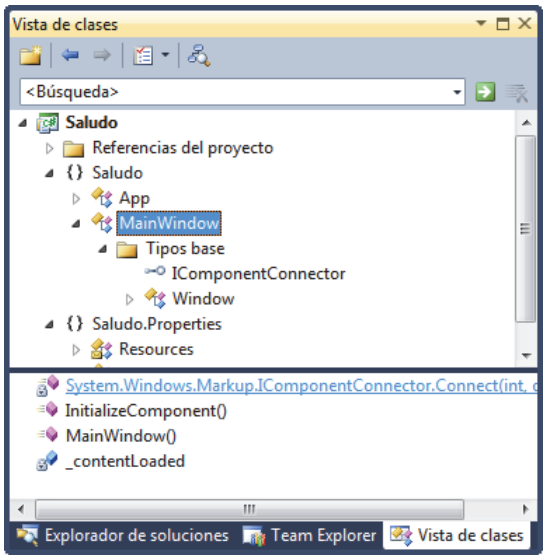
Así mismo, en su parte superior, muestra una barra de botones que permiten ver el *código*, el *diseñador* de formularios, la ventana de *propiedades*, etc. Por ejemplo, si estamos viendo el diseñador de formularios y hacemos clic en el botón *Ver código*, la página de diseño será sustituida por el editor de código, como se puede observar en la figura siguiente:



Una característica digna de resaltar del editor de Visual Studio es la incorporación de bloques de código contraíbles. En la figura superior podemos ver varios de estos bloques; si hacemos clic en un nodo –, contraeremos el bloque y ese nodo se convertirá en otro + que permitirá expandir de nuevo el bloque.

Otra característica del editor es la finalización y el formato de código automáticos. Por ejemplo, al escribir un método, el editor mostrará automáticamente la ayuda en línea de la palabra clave (**public**, **void**, **int**, etc.) que intenta escribir; si escribimos una sentencia **if**, exactamente igual. Puede personalizar las características del editor ejecutando *Herramientas* > *Opciones* > *Editor de texto*.

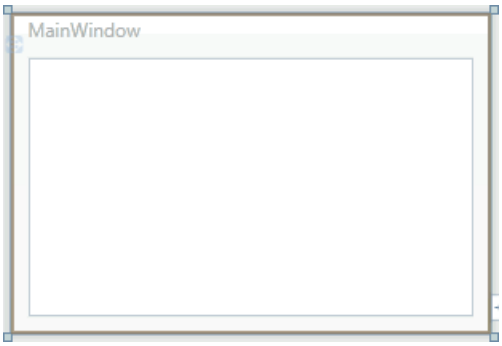
Si cuando se está visualizando el explorador de soluciones desea mostrar la vista de clases de su aplicación, sólo tiene que hacer clic en la pestaña *Vista de clases*. Esta ventana en su parte superior muestra las clases que componen la aplicación y en su parte inferior los métodos pertenecientes a la clase seleccionada.



Expandiendo el nodo del proyecto, vemos, en primer lugar, el espacio de nombres al que pertenecen las clases que definen la aplicación: *Saludo* (un espacio de nombres define un ámbito). Si ahora expandimos este otro nodo, veremos que incluye las clases mencionadas, la que define el objeto aplicación, *App*, y la que define la ventana principal, *MainWindow*, y si expandimos a su vez este nodo, podremos observar su clase base, **Window**. Si seleccionó el nodo *MainWindow*, en el panel inferior de la figura podemos observar los métodos *MainWindow*, es el constructor de la clase del mismo nombre (la clase seleccionada), y el método *InitializeComponent*, entre otros.

El formulario

El formulario, objeto de la clase **Window** (una ventana), es el plano de fondo para los controles. Después de crear un nuevo proyecto, la página de diseño muestra uno como el de la figura siguiente. Lo que ve en la figura es el aspecto gráfico de un objeto de la clase *MainWindow*. Para modificar su tamaño ponga el cursor del ratón sobre alguno de los lados del cuadrado que lo rodea y arrastre en el sentido deseado.



Si ahora ejecutamos esta aplicación, para lo cual podemos pulsar las teclas *Ctrl+F5*, o bien elegir la orden correspondiente del menú *Depurar*, aparecerá sobre la pantalla el formulario (un objeto ventana), con el tamaño asignado, y podremos actuar sobre cualquiera de sus controles, o bien sobre las órdenes del menú de control, para minimizarlo, maximizarlo, moverlo, ajustar su tamaño, etc. Ésta es la parte que el diseñador de Visual Studio realiza por nosotros y para nosotros; pruébelo. Finalmente, para cerrar la ejecución de la aplicación disponemos de varias posibilidades:

1. Hacer clic en el botón que cierra la ventana.
2. Hacer un doble clic en el icono situado a la izquierda en la barra de título de la ventana.
3. Activar el menú de control de la ventana *MainWindow* y ejecutar *Cerrar*.
4. Pulsar las teclas *Alt+F4*.

Dibujar los controles

En Visual Studio disponemos fundamentalmente de dos tipos de objetos: *ventanas* y *controles*. Las ventanas son los objetos sobre los que se dibujan los controles como cajas de texto, botones o etiquetas, dando lugar a la interfaz gráfica que el usuario tiene que utilizar para comunicarse con la aplicación y que genéricamente denominamos formulario.

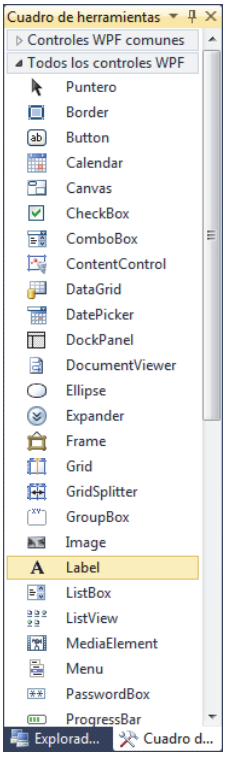
Para añadir un control a un formulario, utilizaremos la caja de herramientas que se muestra en la figura siguiente. Cada herramienta de la caja crea un único control. El significado de algunos de los controles más comunes se expone a continuación.

Puntero. El puntero no es un control. Se utiliza para seleccionar, mover y ajustar el tamaño de los objetos.

Label. Una *etiqueta* permite mostrar un texto de una o más líneas que no puede ser modificado por el usuario. Son útiles para dar instrucciones al usuario.

Button. Un *botón de pulsación* normalmente tendrá asociada una orden con él. Esta orden se ejecutará cuando el usuario haga clic sobre el botón.

TextBox. Una *caja de texto* es un área dentro del formulario en la que el usuario puede escribir o visualizar texto.



Menu. Permite añadir un menú de opciones para que el usuario seleccione una.

CheckBox. Una *casilla de verificación* se utiliza para seleccionar una opción. Utilizando estos controles se pueden elegir varias opciones de un grupo.

RadioButton. El control *botón de opción* se utiliza para seleccionar una opción entre varias. Utilizando estos controles se puede elegir una sola opción de un grupo de ellas.

GroupBox. Un *marco* se utiliza para realzar el aspecto del formulario. También los utilizamos para formar grupos de botones de opción, o bien para agrupar controles relacionados entre sí.

Grid. Control que actúa como contenedor de otros controles.

DataGrid. Proporciona una tabla para visualizar los datos de un origen de datos de una forma personalizada.

ListBox. El control *lista fija* (lista desplegada) contiene una lista de elementos de la que el usuario puede seleccionar uno o varios.

ComboBox. El control *lista desplegable* combina una caja de texto y una lista desplegable. Permite al usuario escribir lo que desea seleccionar o elegir un elemento de la lista.

ListView. El control *vista de lista* muestra una colección de elementos que se pueden visualizar mediante una de varias vistas distintas.

TreeView. Representa un control que muestra datos jerárquicos en una estructura de árbol con nodos que se pueden expandir y contraer.

TabControl. Es un control que agrupa un conjunto relacionado de fichas.

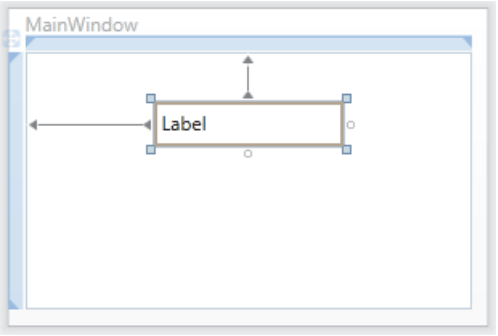
DatePicker. Control que permite seleccionar la fecha y hora.

ScrollBar. Representa una *barra de desplazamiento horizontal* o *vertical* que permite seleccionar un valor dentro de un rango de valores. Estos controles son utilizados independientemente de otros objetos, y no son lo mismo que las barras de desplazamiento de una ventana.

Otros controles de interés son la barra de progreso (*ProgressBar*), la caja de texto enriquecido (*RichTexBox*), la barra de estado (*StatusBar*), etc.

Observe que el formulario *MainWindow* ya contiene un **Grid** que actuará como contenedor de los controles que arrastremos sobre el mismo.

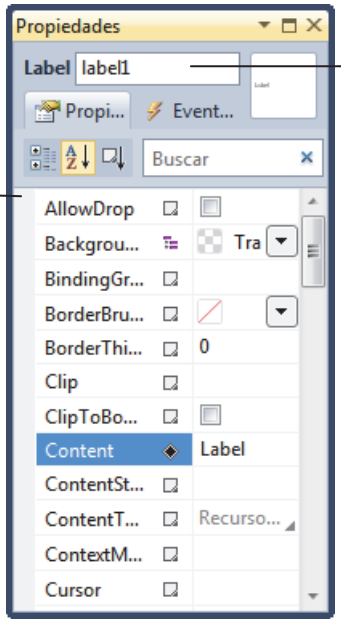
Siguiendo con nuestra aplicación, seleccionamos de la caja de herramientas que acabamos de describir los controles que vamos a utilizar. En primer lugar vamos a añadir al formulario una etiqueta. Para ello, hacemos clic sobre la herramienta etiqueta (*Label*) y, sin soltar el botón del ratón, la arrastramos sobre el formulario. Cuando soltemos el botón del ratón aparecerá una etiqueta de un tamaño predefinido, según se muestra en la figura siguiente:



Observe en la página de propiedades del entorno de desarrollo, mostrada en la figura siguiente, las propiedades **Name**, nombre, y **Content**, contenido. La primera tiene asignado el valor *label1* que es el nombre por defecto dado al control *Label*, y la segunda tiene asignado por defecto el valor "Label", contenido que mostrará la etiqueta.

Si la ventana propiedades no está visible, ejecute la orden *Ventana de propiedades* en el menú *Ver* o pulse *F4*.

El nombre de un control se utiliza para referirnos a dicho control en el código de la aplicación.

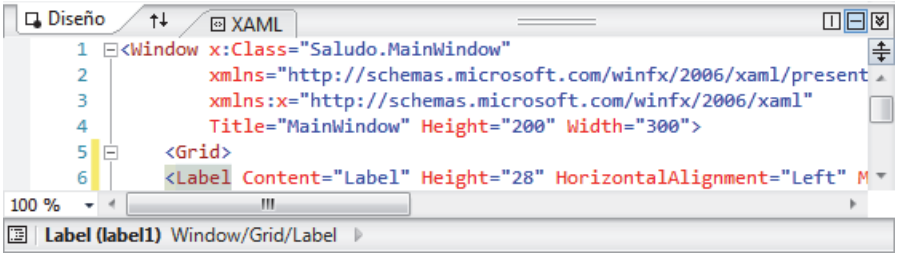


Estando la etiqueta seleccionada se observa sobre la misma un rectángulo con unos cuadrados distribuidos a lo largo de su perímetro, que reciben el nombre de

modificadores de tamaño, indicando que se puede modificar el tamaño del control que estamos dibujando. Para ello, primero selecciónelo haciendo clic sobre él, después apunte con el ratón a alguno de los lados del rectángulo que lo envuelve, observe que aparece una doble flecha, y, entonces, con el botón izquierdo del ratón pulsado, arrastre en el sentido que desee ajustar el tamaño.

También puede mover el control a un lugar deseado dentro del formulario. Para mover un control, primero selecciónelo haciendo clic sobre él y después apunte con el ratón a alguna zona perteneciente al mismo y, con el botón izquierdo del ratón pulsado, arrastre hasta situarlo en el lugar deseado.

Después de haber arrastrado un control sobre el formulario, Visual Studio crea automáticamente el código XAML que hará que el control se muestre cuando se ejecute el programa. De manera predeterminada, el editor XAML aparece bajo el diseñador. En la parte superior de este editor hay botones que le permitirán ver el marcado XAML en modo pantalla completa, en modo pantalla compartida horizontalmente o verticalmente, en la parte superior, etc., y en la parte inferior izquierda hay otro botón que muestra el esquema del documento que le permitirá navegar por el árbol de elementos de la ventana.



Borrar un control

Para borrar un control, primero se selecciona haciendo clic sobre él y, a continuación, se pulsa la tecla *Supr* (*Del*). Para borrar dos o más controles, primero se seleccionan haciendo clic sobre cada uno de ellos, al mismo tiempo que se mantiene pulsada la tecla *Ctrl*, y después se pulsa *Supr*.

Se pueden seleccionar también dos o más controles contiguos, pulsando el botón izquierdo y arrastrando el ratón hasta rodearlos.

Propiedades de los objetos

Cada clase de objeto tiene predefinido un conjunto de propiedades, como nombre, tamaño, color, etc. Las propiedades de un objeto representan todos los atributos que por definición están asociados con ese objeto.

Cuando se selecciona más de un objeto, la página de propiedades visualiza las propiedades comunes a esos objetos.

Cada propiedad de un objeto tiene un valor por defecto que puede ser modificado *in situ* si se desea. Por ejemplo, la propiedad **Title** del formulario del ejemplo que nos ocupa tiene el valor *MainWindow*.

Para cambiar el valor de una propiedad de un objeto, siga los pasos indicados a continuación:

1. Seleccione el objeto. Para ello, haga clic sobre el objeto o pulse sucesivamente la tecla *Tab* hasta que esté seleccionado (el control seleccionado aparecerá rodeado por un rectángulo modificador de tamaño).

2. Seleccione en la lista de propiedades la propiedad que desea cambiar.

3. Modifique el valor que actualmente tiene la propiedad seleccionada. El valor actual de la propiedad en cuestión aparece escrito a continuación del nombre de la misma. Para cambiar este valor, sobrescriba el valor actual o, si es posible, seleccione uno de la lista que se despliega haciendo clic sobre la misma. Para algunas propiedades, esta lista es sustituida por una caja de diálogo.

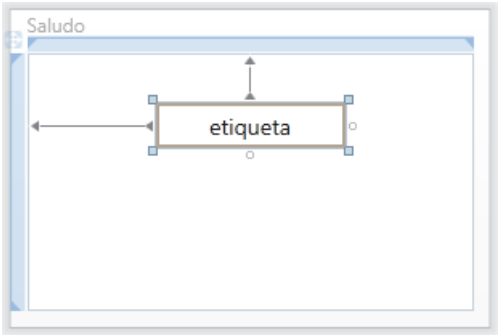
Se puede también modificar una propiedad durante la ejecución de la aplicación. Esto implica añadir el código necesario en el método que deba realizar la modificación.

Para verificar el valor de una misma propiedad en varios objetos, se selecciona ésta en la página de propiedades para uno de ellos, y a continuación se pasa de un objeto al siguiente haciendo clic con el ratón sobre cada uno de ellos, o simplemente pulsando la tecla *Tab*.

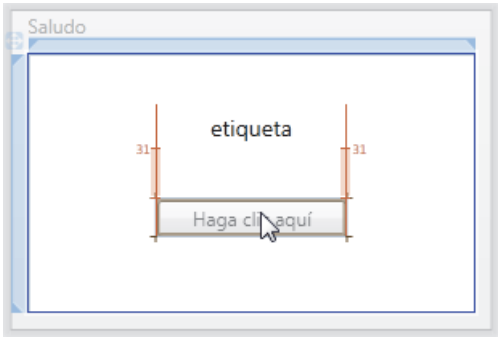
Siguiendo con nuestro ejemplo, vamos a cambiar el título del formulario. Para ello, seleccione el formulario y a continuación la propiedad **Title** en la página de propiedades. Después, sobrescriba el texto “MainWindow” con el texto “Saludo”.

Veamos ahora las propiedades de la etiqueta. Seleccione la etiqueta y observe la lista de propiedades. Algunas de estas propiedades son **Background** (color del fondo de la etiqueta), **Name** (identificador de la etiqueta para referirnos a ella en el código) y **Content** (contenido de la etiqueta). Siguiendo los pasos descritos anteriormente, cambie el valor actual de la propiedad **Name** al valor *etSaludo*, el contenido “Label” de la propiedad **Content** a “etiqueta” y alinee este texto para que se muestre centrado tanto horizontal como verticalmente; esto requiere asignar a las propiedades **HorizontalContentAlignment** y **VerticalContentAlignment** el valor *Center*. A continuación, vamos a modificar el tamaño de la letra de

la etiqueta; para ello, seleccione la propiedad **FontSize** en la página de propiedades, despliegue la lista de la derecha y elija como tamaño, por ejemplo, 14. El resto de las propiedades las dejamos como están.



El paso siguiente será añadir un botón. Para ello, hacemos clic sobre la herramienta *Button* de la caja de herramientas y arrastramos el botón sobre el formulario. Modificamos sus propiedades y asignamos a **Content** el valor *Haga clic aquí*, y a **Name**, el valor *btSaludo*. Después, movemos el botón y ajustamos su tamaño para conseguir el diseño que observamos en la figura siguiente.



También observamos que al colocar el control aparecen unas líneas indicando la alineación de éste con respecto a otros controles. Es una ayuda para alinear los controles que coloquemos dentro del formulario. Puede elegir entre los modos *SnapLines* (líneas de ayuda), es el modo que estamos utilizando, o *SnapToGrid* (rejilla de ayuda; se visualizan los puntos que dibujan la rejilla). Para elegir el modo de ayuda, ejecute *Herramientas* > *Opciones*, seleccione la opción *Diseñador de formularios Windows* y asigne a la propiedad **LayoutMode** el modo deseado. Para que las opciones elegidas tengan efecto, tiene que cerrar el diseñador y volverlo a abrir.

Icono de la aplicación

Todos los formularios visualizan un icono en la esquina superior izquierda que generalmente ilustra la finalidad de la aplicación y que también aparece cuando se minimiza el formulario. Por omisión, Visual Studio utiliza un icono genérico.

Para utilizar su propio icono (de 16 × 16 o de 32 × 32 píxeles), sólo tiene que asignarlo a la propiedad **Icon** del formulario; esto es, seleccione el formulario, vaya a la página de propiedades, elija la propiedad **Icon**, pulse el botón que se muestra a la derecha y asigne el fichero *.ico* que contiene el icono.

Escribir los controladores de eventos

Sabemos que el nombre de un objeto, propiedad **Name**, nos permite referirnos a él dentro del código de la aplicación; por ejemplo, en las líneas de código siguiente, la primera asigna el valor "¡¡¡Hola mundo!!!" a la propiedad **Content** del objeto *etSaludo* y la siguiente obtiene el valor de la etiqueta y lo almacena en la variable *sTexto*:

```
etSaludo.Content = "¡¡¡Hola mundo!!!";
string sTexto = etSaludo.Content.ToString();
```

En C# la forma general de referirse a una propiedad de un determinado objeto es:

Objeto.Propiedad

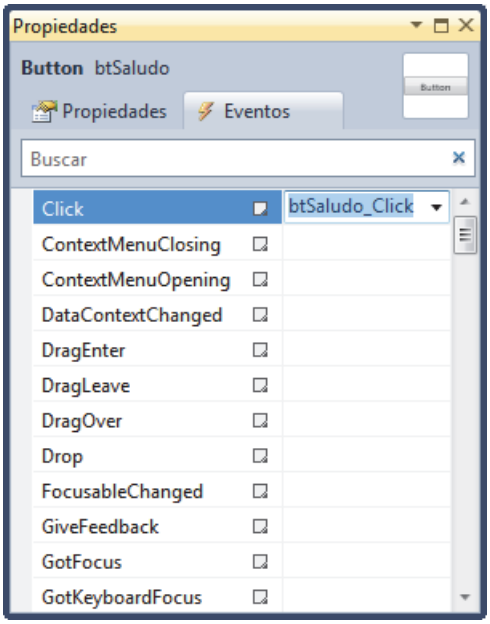
donde *Objeto* es el nombre del formulario o control y *Propiedad* es el nombre de la propiedad del objeto cuyo valor queremos asignar u obtener.

Una vez que hemos creado la interfaz o medio de comunicación entre la aplicación y el usuario, tenemos que escribir los métodos para controlar, de cada uno de los objetos, aquellos eventos que necesitemos manipular.

Hemos dicho que una aplicación en Windows es conducida por *eventos* y *orientada a objetos*. Esto es, cuando sobre un objeto ocurre un suceso (por ejemplo, el usuario hizo clic sobre un botón) se produce un evento (por ejemplo, el evento **Click**); si nosotros deseamos que nuestra aplicación responda a ese evento, tendremos que escribir un método que incluya el código que debe ejecutarse y vincularlo con el objeto que genera el evento. El método pertenecerá a la interfaz del objeto o del objeto padre. Por ejemplo, el método que responda al evento **Click** de un botón pertenecerá a la interfaz de su ventana padre, esto es, a su contenedor.

¿Dónde podemos ver la lista de los eventos a los que puede responder un objeto de nuestra aplicación? En la ventana de propiedades.

Por ejemplo, seleccione el botón *btSaludo* en la ventana de diseño, vaya a la ventana de propiedades y muestre la lista de eventos para el control seleccionado, haciendo clic en el botón *Eventos*. Haga doble clic en el evento **Click**, o bien escriba manualmente el nombre del controlador y pulse *Entrar*.



El resultado es que se añade a la clase *MainWindow* un manejador para este evento (fichero *MainWindow.xaml*),

```
<Window x:Class="Saludo.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Saludo" Height="200" Width="300" >
  <Grid>
    <Label Name="etSaludo" Content="etiqueta" FontSize="14"
      Height="28" Width="118" Margin="80,31,0,0"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      HorizontalContentAlignment="Center"
      VerticalContentAlignment="Center" />
    <Button Name="btSaludo" Content="Haga clic aquí"
      Height="23" Width="118" Margin="80,96,0,0"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      Click="btSaludo_Click" />
  </Grid>
</Window>
```

y el método *btSaludo_Click* que responderá a ese evento **Click** cada vez que se genere (fichero *MainWindow.xaml.cs*):

```
private void btSaludo_Click(object sender, RoutedEventArgs e)
{
  // Escriba aquí el código que tiene que ejecutarse para responder
  // al evento Click que se genera al pulsar el botón
}
```

El primer parámetro del método anterior hace referencia al objeto que generó el evento y el segundo contiene datos relacionados con el evento.

Una vez añadido el controlador para el evento **Click** del botón *btSaludo*, ¿cómo lo completamos? Lo que deseábamos era que la etiqueta mostrara el mensaje "¡¡¡Hola mundo!!!" cuando el usuario hiciera clic en el botón. Según esto, complete este controlador así:

```
private void btSaludo_Click(object sender, RoutedEventArgs e)
{
  etSaludo.Content = "¡¡¡Hola mundo!!!";
}
```

Para añadir el controlador anterior, también podríamos habernos dirigido a la página de diseño y haber hecho doble clic sobre el botón de pulsación.

Además del evento **Click**, hay otros eventos asociados con un botón de pulsación, según se puede observar en la figura anterior.

Un detalle de estilo a la hora de escribir el código. Observe que *Visual Studio*, para no anteponer a los nombres de las clases y otras estructuras de datos el nombre del espacio de nombres al que pertenecen (por ejemplo **System.Object** en lugar de escribir solamente el nombre de la clase **Object**), añade al principio del código fuente las sentencias **using**, que se muestran a continuación, que los especifican.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Shapes;
```

Análogamente a como las carpetas o directorios ayudan a organizar los ficheros en un disco duro, los espacios de nombres ayudan a organizar las clases en grupos para facilitar el acceso a las mismas y proporcionan una forma de crear ti-

pos globales únicos, evitando conflictos en el caso de clases de igual nombre pero de distintos fabricantes, ya que se diferenciarán en su espacio de nombres.

Guardar la aplicación

Una vez finalizada la aplicación, se debe guardar en el disco para que pueda tener continuidad; por ejemplo, por si más tarde se quiere modificar. Esta operación puede ser que se realice automáticamente cuando se compila o se ejecuta la aplicación y si no, puede requerir guardar la aplicación en cualquier instante ejecutando la orden *Guardar todo* del menú *Archivo*.

Si desplegamos el menú *Archivo*, nos encontraremos, además de con la orden *Guardar todo*, con dos órdenes más: *Guardar nombre-fichero* y *Guardar nombre-fichero como...* La orden *Guardar nombre-fichero* guarda en el disco el fichero actualmente seleccionado y la orden *Guardar nombre-fichero como...* realiza la misma operación, y además nos permite cambiar el nombre, lo cual es útil cuando el fichero ya existe.

No es conveniente que utilice los nombres que Visual C# asigna por defecto, porque pueden ser fácilmente sobrescritos al guardar aplicaciones posteriores.

Verificar la aplicación

Para ver cómo se ejecuta la aplicación y los resultados que produce, hay que seleccionar la orden *Iniciar sin depurar* del menú *Depurar* o pulsar *Ctrl+F5*.

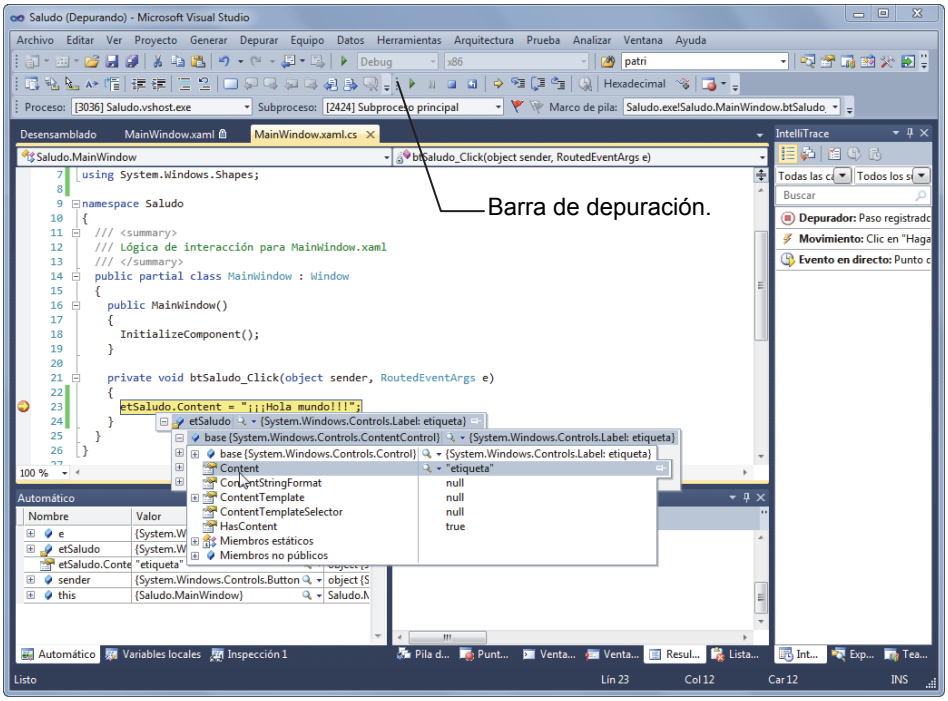
Si durante la ejecución encuentra problemas o la solución no es satisfactoria y no es capaz de solucionarlos por sus propios medios, puede utilizar, fundamentalmente, las órdenes *Paso a paso por instrucciones* (*F11*), *Paso a paso por procedimientos* (*F10*), *Alternar puntos de interrupción* (*F9*), todas ellas del menú *Depurar*, para hacer un seguimiento paso a paso de la aplicación, y las opciones del menú *Depurar > Ventanas* para observar los valores que van tomando las variables y expresiones de la aplicación.

La orden *Paso a paso por instrucciones* permite ejecutar cada sentencia de cada método de la aplicación paso a paso. Esta modalidad se activa y se continúa pulsando *F11*. Cuando una sentencia se corresponde con una llamada a un método y quiere que éste se ejecute en un solo paso, utilice la tecla *F10* (*Paso a paso por procedimientos*). Para detener la depuración pulse las teclas *Mayús+F5*.

La orden *Alternar un punto de interrupción* (*F9*) permite colocar una pausa en cualquier línea. Esto permite ejecutar la aplicación hasta la pausa en un solo paso (*F5*), y ver en la ventana *Automático* los valores que tienen las variables en

ese instante. Para poner o quitar una pausa, se coloca el cursor donde se desea que tenga lugar dicha pausa y se pulsa *F9*, o bien se hace clic con el ratón sobre la barra situada a la izquierda del código.

Alternativamente al menú de depuración, puede utilizar la barra de herramientas de depuración. La figura siguiente muestra esta barra dentro de la ventana de Visual Studio en un proceso de depuración. La línea de código sombreada es la siguiente sentencia a ejecutar.

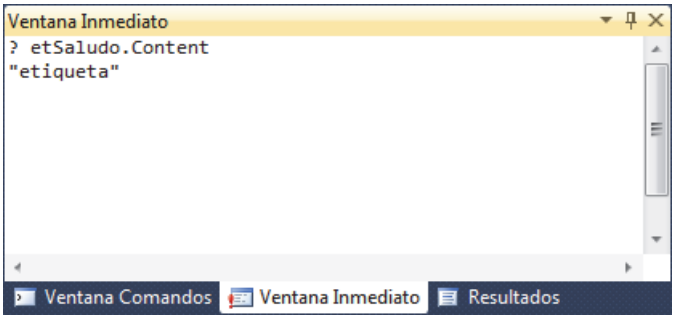


También puede utilizar el ratón para arrastrar el puntero de ejecución (observe la flecha en el margen izquierdo de la ventana anterior) a otro lugar dentro del mismo método con la intención de alterar el flujo normal de ejecución.

Durante el proceso de depuración, puede ver en la ventana *Automático* los valores de las variables y expresiones que desee. Además, en la ventana *Inspección* puede escribir la expresión cuyo resultado desea ver.

También, puede seleccionar en la ventana de código el objeto o propiedad cuyo valor quiere inspeccionar y ejecutar *Inspección rápida...* del menú *Depurar*. Una forma más rápida de hacer esto último es dirigirse al código y situar el puntero del ratón sobre el identificador del objeto o propiedad; le aparecerá una etiqueta con el valor, como se puede observar en la ventana de código anterior.

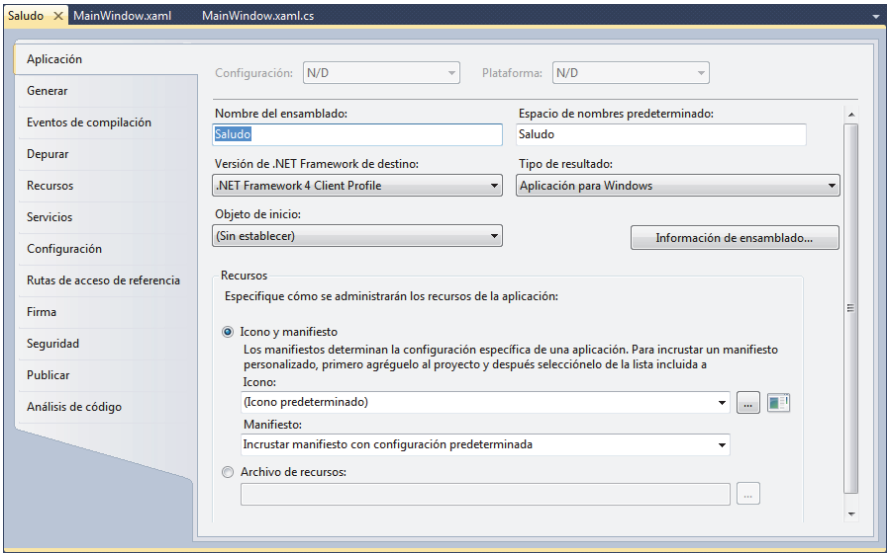
Así mismo, según se observa en la figura siguiente, puede ejecutar en la *Ventana Inmediato* cualquier sentencia de una forma inmediata. Para mostrar u ocultar esta ventana ejecute la orden *Ventanas > Inmediato* del menú *Depurar*. La figura siguiente muestra un ejemplo de cómo se puede inspeccionar el valor de la propiedad **Content** de la etiqueta *etSaludo* (observe el uso del símbolo ?).



Una vez iniciada la ejecución de la aplicación, si se pulsa la tecla *F5*, la ejecución continúa desde la última sentencia ejecutada en un método hasta finalizar ese método o hasta otro punto de parada.

Propiedades del proyecto

Cuando sea necesario establecer determinadas propiedades del proyecto actual o revisar las actuales hay que ejecutar la orden *Proyecto > Propiedades de nombre-proyecto...* Se le mostrará una ventana con varios paneles. Seleccione el deseado y modifique las propiedades que considere.



Crear soluciones de varios proyectos

Una solución agrupa uno o más proyectos. Por omisión, cuando se crea un nuevo proyecto, en la misma carpeta física se crea la solución (fichero con extensión *.sln*) a la que pertenece, con el mismo nombre que el proyecto. Esta solución permite que los ficheros que forman parte del proyecto se almacenen bajo una estructura de directorios que facilite su posterior localización así como las tareas de compartir la solución con otros desarrolladores de un supuesto equipo.

¿Qué tenemos que hacer si necesitamos agrupar varios proyectos bajo una misma solución? Crear una solución vacía y añadir nuevos proyectos a la solución o añadir nuevos proyectos a la solución existente. Asegúrese de que se va a mostrar siempre el nombre de la solución en el explorador de soluciones. Para ello, ejecute *Herramientas > Opciones > Proyectos y Soluciones > Mostrar siempre la solución*.

Para crear una nueva solución vacía:

1. Ejecute la orden *Archivos > Nuevo > Proyecto*.

2. Como tipo de proyecto, seleccione *Otros Tipos de Proyectos*.

3. Y como plantilla, seleccione *Solución en Blanco*.

4. Finalmente, introduzca el nombre que desea dar a la solución. Se creará un fichero *.sln* con el nombre dado, almacenado en una carpeta con el mismo nombre. Puede elegir, si lo desea, la posición que ocupará esa carpeta en el sistema de ficheros de su plataforma.

Para añadir un nuevo proyecto a una solución existente:

1. Diríjase al explorador de soluciones y haga clic sobre el nombre de la solución utilizando el botón secundario del ratón. Del menú contextual que se visualiza, ejecute la orden *Añadir > Nuevo Proyecto...*

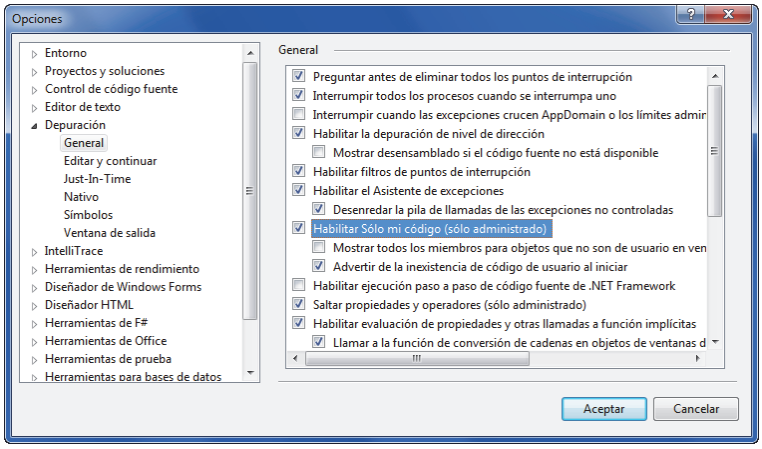
2. Seleccione el tipo de proyecto y la plantilla que va a utilizar para crearlo.

3. Para añadir nuevos proyectos repita los pasos anteriores.

4. Para activar el proyecto sobre el que va a trabajar, haga clic sobre el nombre del proyecto utilizando el botón secundario del ratón y del menú contextual que se visualiza, ejecute la orden *Establecer como proyecto de inicio*.

Opciones del EDI

Las opciones del entorno que se pueden establecer en el entorno de desarrollo integrado (EDI) son mostradas por la ventana *Opciones* que observa en la figura siguiente. Para mostrar esta ventana tiene que ejecutar la orden *Herramientas > Opciones...* Se puede observar que desde esta ventana es posible establecer opciones para el entorno de desarrollo, para los proyectos y soluciones, para el diseñador, para el depurador, etc. Por ejemplo, para que el depurador sólo navegue a través del código escrito por el usuario, no sobre el código añadido por los asistentes, tiene que estar activada la opción "habilitar sólo mi código"; *Herramientas > Opciones > Depuración > General > Habilitar sólo mi código*.



Personalizar el EDI

Para personalizar el entorno de desarrollo tiene que ejecutar la orden *Herramientas > Personalizar...* Desde esta ventana podrá añadir o quitar elementos de un menú, añadir o quitar una barra de herramientas, añadir o quitar un botón de una barra de herramientas, etc.

SQL SERVER EXPRESS

SQL Server Express es el motor de base de datos gratuito, potente, pero sencillo, que se integra perfectamente con el resto de productos Express. Se trata de una versión aligerada de la nueva generación de SQL Server.

Este producto tiene el mismo motor de base de datos que toda la familia SQL Server y utiliza el mismo lenguaje SQL, pero tiene ciertas limitaciones como por ejemplo, sólo puede usar una UCP y un máximo de 1GB de RAM, y las bases de

datos no pueden sobrepasar los 4GB. Otra característica interesante es la movilidad de las bases de datos de un servidor a otro con **XCOPY**. Con esta utilidad podemos mover un fichero MDF de una máquina a otra a cualquier ubicación dentro de su sistema de ficheros, quedando la base de datos movida lista para trabajar. Para utilizar esta base de datos deberemos hacer uso de la opción **AttachDBFilename** en la cadena de conexión, según se muestra a continuación:

```
connectionString="Data Source=.\sqlexpress;Initial Catalog=;
Integrated Security=True;AttachDBFileName=C:\bd\bd_telefonos.mdf"
```

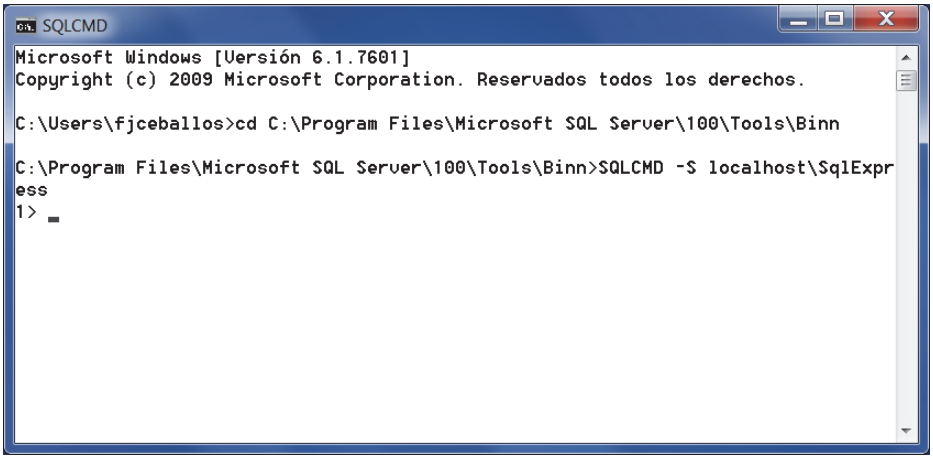
La entrada *Data Source* especifica el servidor de base de datos que vamos a utilizar y **AttachDBFilename**, la localización del fichero de base de datos. Obsérvese que la entrada *Initial Catalog* está vacía. O también, si la base de datos la copiamos en el directorio *App_Data* de una aplicación Web, la cadena de conexión sería similar a esta otra:

```
connectionString = "Data Source=.\sqlexpress;" +
"Integrated Security=True;" +
"AttachDBFilename=|DataDirectory|\bd_telefonos.mdf;" +
"User Instance=True";
```

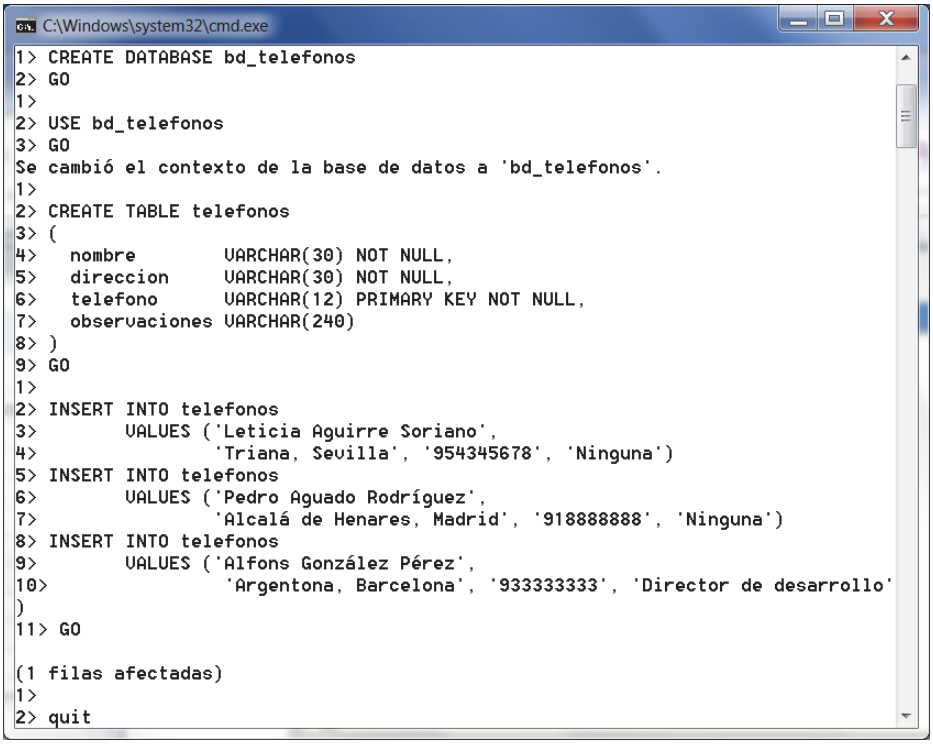
El nombre del fichero comienza con |*DataDirectory*|. Esto automáticamente apunta a la carpeta *App_Data* del proyecto Web. La entrada *User Instance* especifica que la base de datos es una instancia del usuario porque ésta no está registrada en el catálogo maestro del SQL Server. Esto es, cada base de datos que generamos mediante un *script*, por ejemplo, su nombre es registrado en el catálogo maestro, nombre que después será utilizado en la cadena de conexión por *Initial Catalog*, pero si copiamos una base de datos, ésta no es registrada. En este caso Sql Server Express permite acceder a ella tratándola como una instancia del usuario, característica que no está disponible en Sql Server.

Para crear una base de datos utilizando SQL Server Express tiene que hacerlo desde la línea de órdenes (véase también el capítulo titulado *Acceso a una base de datos*). Para iniciar la consola que le permita trabajar contra el motor de base de datos SQL Server, localice en su instalación el fichero SQLCMD.EXE, cambie a ese directorio y ejecute la orden:

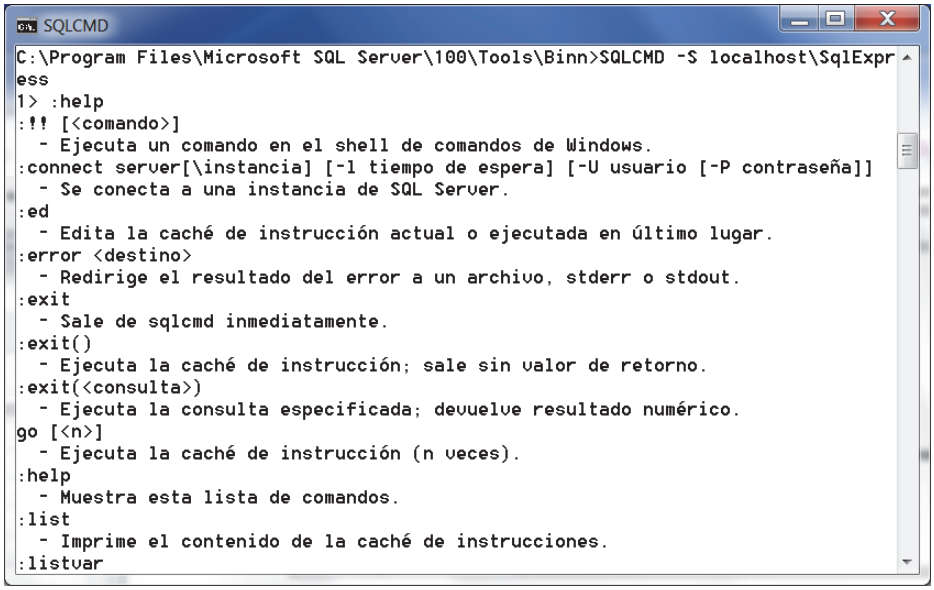
```
SQLCMD -S nombre-del-ordenador\SqlExpress
```



Una vez iniciada la consola, puede escribir órdenes SQL a continuación del símbolo ">". Para ejecutar un bloque de sentencias escriba GO. Para salir, escriba QUIT. Por ejemplo, el guión que muestra la figura siguiente crea la base de datos *bd_telefonos* con una tabla *telefonos*, y añade tres filas a la tabla:



Para ver la relación de órdenes que puede utilizar a través de la aplicación *SQLCMD* ejecute la orden **help** como se muestra en la figura siguiente. Obsérvese que cada orden va precedida por dos puntos (:).



SQL SERVER MANAGEMENT STUDIO EXPRESS

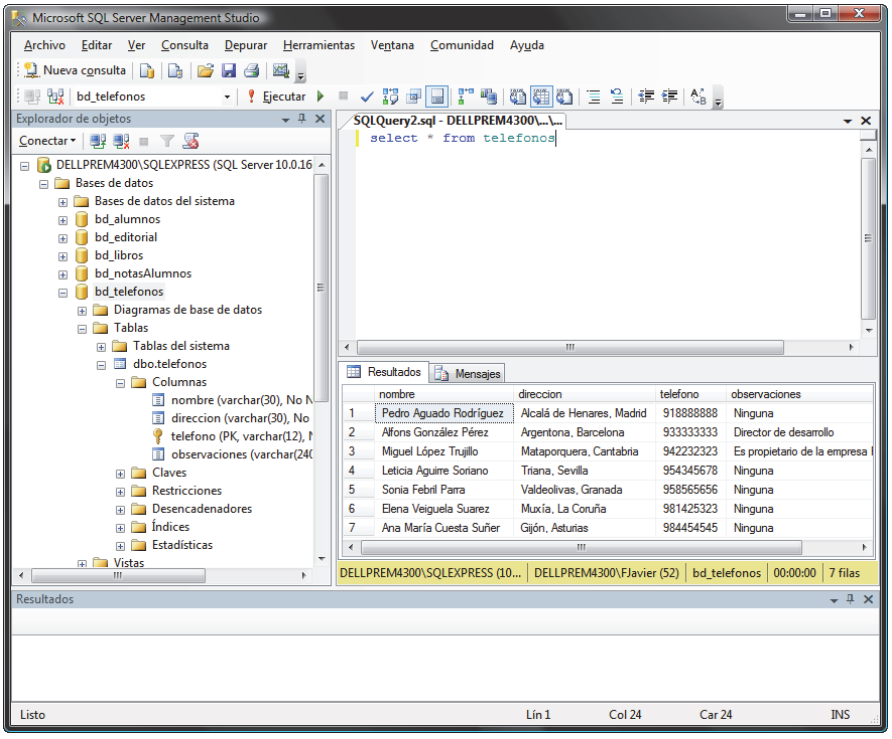
Si instaló SQL Server, habrá comprobado que la única herramienta que proporciona al usuario es *SQL Computer Manager* que sirve para gestionar los servicios básicos de SQL Server y para configurar los protocolos de red. Por tal motivo, Microsoft también ha desarrollado una nueva aplicación para gestionar bases de datos que puede obtener de forma gratuita de Internet en la dirección especificada a continuación:

http://www.microsoft.com/downloads/es-es/

Esta aplicación presenta una interfaz gráfica, muy sencilla de utilizar, para realizar tareas típicas como crear bases de datos, gestionar las tablas de la base, los procedimientos almacenados, crear usuarios, etc.

Cuando inicie *SQL Server Management Studio Express*, le serán solicitados el nombre del servidor de bases de datos, el tipo de autenticación, y el usuario y la contraseña sólo si eligió autenticación SQL Server:

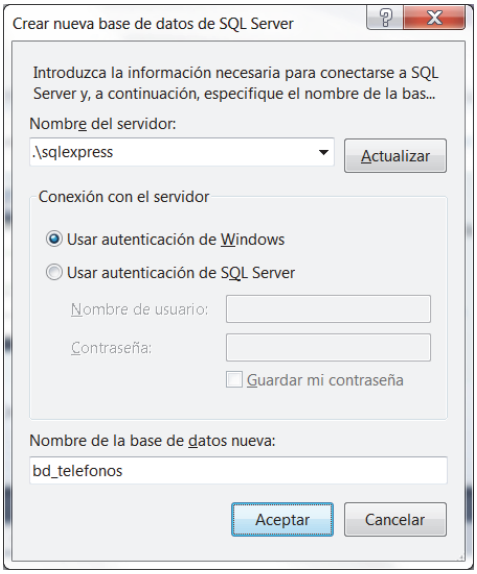
Una vez realizada la conexión con el gestor de bases de datos, le será mostrada la ventana de la figura siguiente. Seleccione en la lista del panel de la izquierda la base de datos con la que desea trabajar, haga clic en el botón *Nueva consulta* de la barra de herramientas y, después, escriba en el mismo las sentencias SQL que desee ejecutar. Para ejecutar una sentencia SQL haga clic en el botón *Ejecutar* de la barra de herramientas.



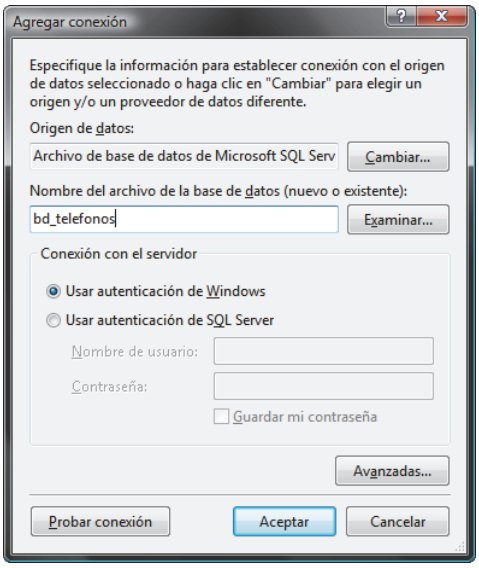
EXPLORADOR DE BASES DE DATOS

Otra forma de crear bases de datos es a través del explorador de bases de datos del EDI de Visual Studio. Si instaló Visual Studio o Visual C# Express y SQL Server Express, entonces es posible añadir al proyecto elementos nuevos de tipo base de datos utilizando el explorador de bases de datos. Para ello, abra el *Explorador de bases de datos* (*Ver* > *Explorador de servidores* en Visual Studio o *Ver* > *Otras ventanas* > *Explorador de bases de datos* en Visual C# Express). Haga clic con el botón secundario del ratón sobre el nodo *Conexiones de datos* y seleccione *Crear nueva base de datos SQL Server* (o *Nueva conexión* en Visual C# Express).

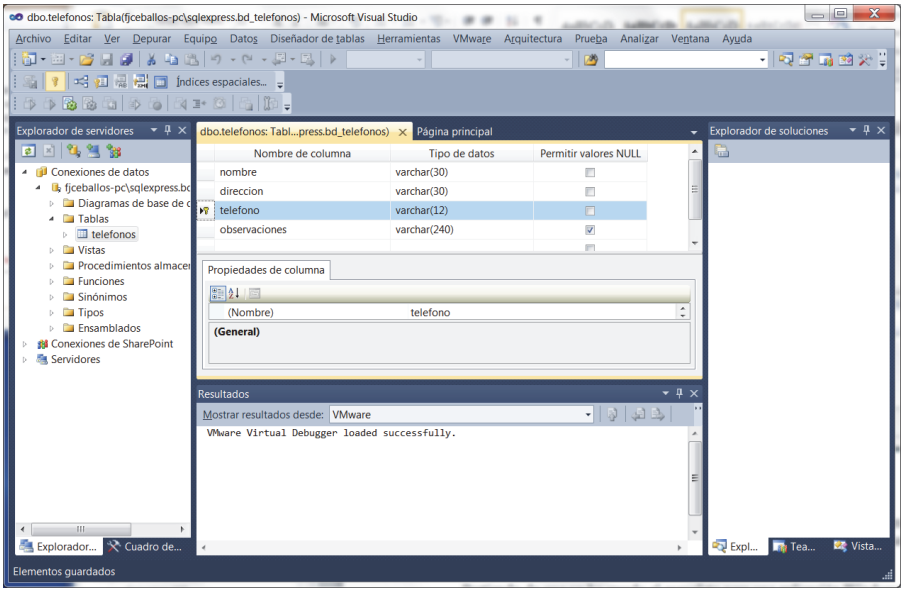
Si está utilizando Visual Studio se visualizará la ventana siguiente. Elija como nombre del servidor .*sqlexpress*, escriba el nombre de la base de datos y haga clic en *Aceptar*.



Si está utilizando Visual C# Express se visualizará esta otra ventana. Elija como origen de datos *Microsoft SQL Server*, escriba el nombre de la base de datos y haga clic en *Aceptar*.



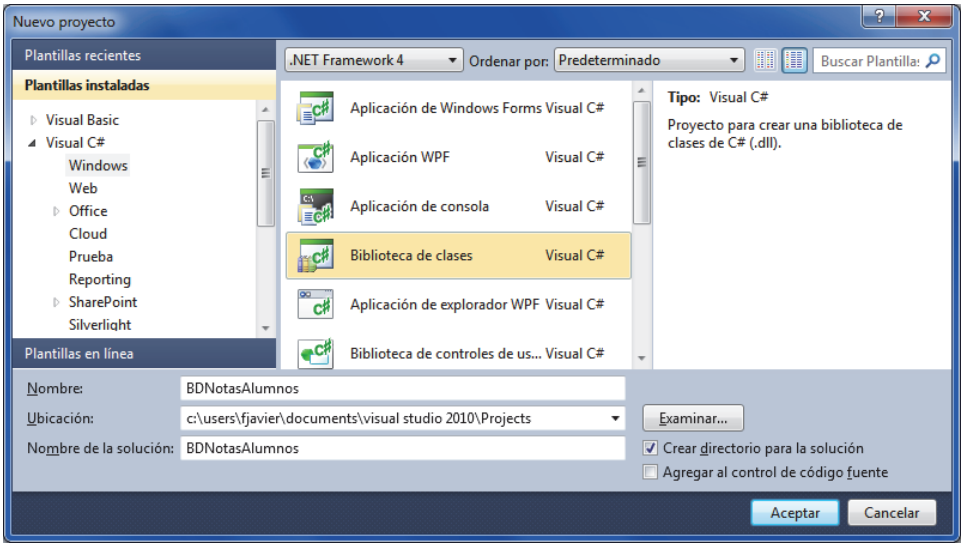
Una vez creada la base de datos, el paso siguiente es añadir las tablas. Despliegue el árbol correspondiente a la nueva conexión, haga clic con el botón secundario del ratón sobre el nodo *Tablas* y agregue una nueva tabla. Después complete el proceso de creación de la misma.



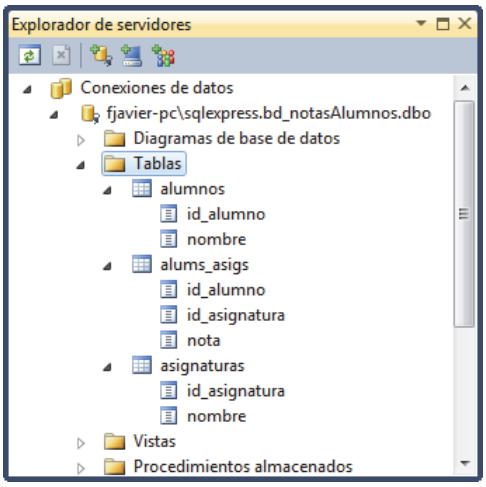
AÑADIR UN DATASET AL PROYECTO

Para acceder a una base de datos podemos escribir código basado en los objetos que proporciona ADO.NET, o bien podemos añadir un **DataSet** (conjunto de datos) al proyecto. Cuando se añade un **DataSet** al proyecto lo que sucede es que el asistente de diseño de Visual Studio crea una capa de acceso (DAL) a las tablas seleccionadas de la base de datos que nos abstrae de SQL. Dicha capa incluye todo un conjunto de clases con la funcionalidad suficiente para modificar, añadir o borrar filas en las tablas de la base de datos.

Como ejemplo, vamos a crear un proyecto *BDNotasAlumnos* de tipo *Biblioteca de clases*, que nos proporcione una interfaz de clases C# para acceder a una base de datos formada por tres tablas, *alumnos*, *asignaturas* y *alums_asigs*, para gestionar las notas de las asignaturas en las que se ha matriculado cada uno de los alumnos (puede ver más detalles acerca de esta base de datos en el capítulo 6). Esta forma de proceder nos permitirá utilizar esta biblioteca en cualquier proyecto donde sea necesaria, simplemente añadiendo al proyecto en cuestión una referencia a la misma.



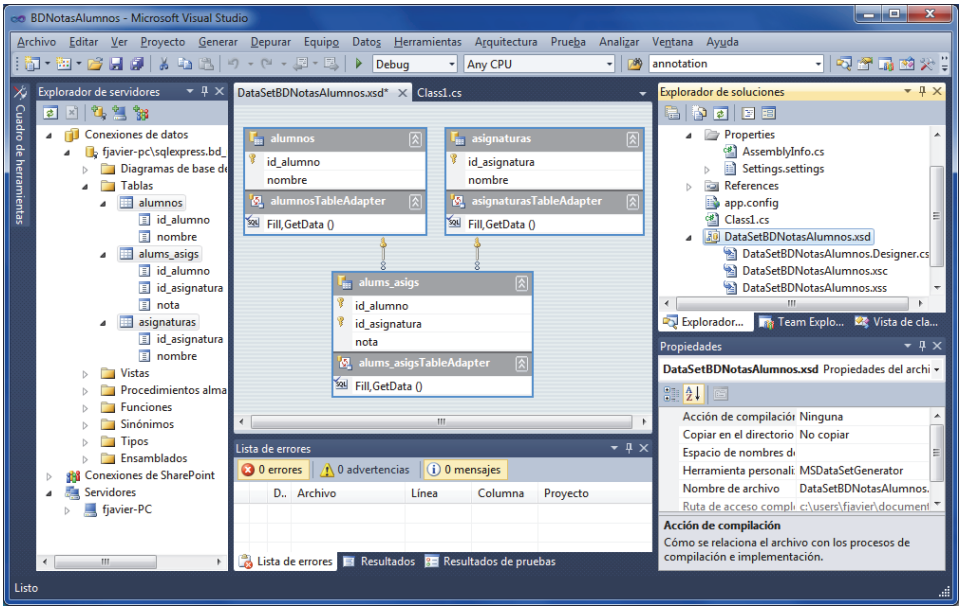
Suponiendo que tenemos creada la base de datos SQL, abrimos el explorador de servidores (explorador de base de datos) y añadimos una conexión a esta base de datos:



Para crear la capa de acceso a la base de datos, añadimos un **DataSet** al proyecto. Para ello, haga clic con el botón secundario del ratón sobre el nombre del proyecto, y seleccione *Agregar* > *Nuevo Elemento* > *Conjunto de datos*, elija un nombre, por ejemplo *DataSetBDNotasAlumnos.xsd*, y haga clic en *Agregar*. Observará en el *Explorador de soluciones* que se ha añadido un esquema XSD inicialmente vacío. También observará que la ventana de edición muestra el diseñador de **DataSet** para crear y editar visualmente conjuntos de datos con tipo.

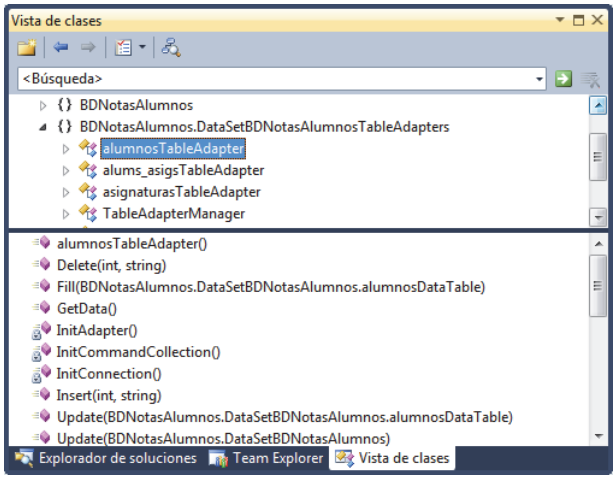
También puede realizar estas operaciones desde el menú *Datos* de Visual Studio.

Como siguiente paso, arrastre desde el *Explorador de servidores* los elementos de la base de datos con los que vaya a trabajar. En nuestro caso seleccionaremos las tres tablas.



También puede agregar nuevos elementos haciendo clic con el botón secundario del ratón sobre la superficie de diseño.

Si ahora muestra la vista de clases, podrá observar la capa de acceso a datos, formada por clases C#, que le permitirá realizar operaciones sobre los elementos de la base de datos que seleccionó y añadió al diseñador.



Esquemas XSD

Se pueden crear vistas XML de datos relacionales utilizando el lenguaje de definición de esquemas XML (XSD). Estas vistas pueden consultarse después utilizando consultas XPath (*lenguaje de rutas XML*).

Un esquema XML describe la estructura de un documento XML y también describe las distintas restricciones en los datos del documento. Todas las declaraciones de elementos deben incluirse dentro del elemento *<xsd:schema>* o *<xs:schema>*:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="DataSetBDNotasAlumnos"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

</xs:schema>
```

Un esquema XSD con anotaciones que describan la base de datos y las operaciones sobre la misma, puede utilizarlo para consultar la base de datos y devolver los resultados en forma de documento XML.

Base de datos XML

Para realizar pruebas rápidas en diferentes proyectos puede ser interesante simular una determinada base de datos con el esquema XSD correspondiente a las tablas del conjunto de datos construido, con los datos de las distintas tablas especificados en XML y con el código C# personalizado necesario para construir a partir de la información anterior, un **DataSet** que modele la base de datos. Después, la funcionalidad proporcionada por la clase **DataSet** nos permitirá acceder a la base de datos.

Como ejemplo, partiendo de la aplicación anterior, vamos a construir en primer lugar los ficheros XML que definan el **DataSet** y los datos almacenados en las distintas tablas que incluimos en el conjunto de datos. Una forma fácil de hacer esto es construir una nueva solución con un nuevo proyecto de tipo *Aplicación WPF*, denominado, por ejemplo, *EnlaceDeDatosXML*, que incluya una referencia a la biblioteca que creamos anteriormente (para mayor facilidad, añada el proyecto *BDNotasAlumnos* a esta solución) y ejecutar desde el mismo, para cada una de las tablas, un código análogo al siguiente:

```
BDNotasAlumnos.DataSetBDNotasAlumnos.alumnosDataTable dt1 =
  new BDNotasAlumnos.DataSetBDNotasAlumnos.alumnosDataTable();
```

```
(new BDNotasAlumnos.DataSetBDNotasAlumnosTableAdapters.
  alumnosTableAdapter()).Fill(dt1);
dt1.WriteXmlSchema("xsd_alumnos.xml");
dt1.WriteXml("datos_alumnos.xml");
```

Una vez obtenido el esquema XSD de cada una de las tablas, fusionamos todas ellas en un único fichero; por ejemplo en *bd_notasAlumnos.xsd*:

```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="bd_notasAlumnos" xmlns=""
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="bd_notasAlumnos"
    msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="alumnos">

        </xs:element>
        <xs:element name="asignaturas">

        </xs:element>
        <xs:element name="alums_asigs">

        </xs:element>
      </xs:choice>
    </xs:complexType>

    <xs:unique name="Constraint1" msdata:PrimaryKey="true">

    </xs:unique>
    <xs:unique name="Constraint2" msdata:PrimaryKey="true">

    </xs:unique>
    <xs:unique name="Constraint3" msdata:PrimaryKey="true">

    </xs:unique>
  </xs:element>
</xs:schema>
```

Hacemos lo mismo con los datos de cada una de las tablas; por ejemplo fusionamos los datos en *bd_notasAlumnos.xml*. Observe que la etiqueta que envuelve todo el documento, *bd_notasAlumnos*, coincide con la identificación del esquema XSD.

```
<?xml version="1.0" standalone="yes"?>
<bd_notasAlumnos>
  <alumnos>
    <id_alumno>1234567</id_alumno>
```

```
    <nombre>Leticia Aguirre Soriano</nombre>
  </alumnos>
  ...
  <asignaturas>
    <id_asignatura>20590</id_asignatura>
    <nombre>PROGRAMACION AVANZADA</nombre>
  </asignaturas>
  ...
  <alums_asigs>
    <id_alumno>1234569</id_alumno>
    <id_asignatura>33693</id_asignatura>
    <nota>9.5</nota>
  </alums_asigs>
</bd_notasAlumnos>
```

Ahora ya podemos eliminar de la biblioteca *BDNotasAlumnos* el **DataSet** que agregamos inicialmente y, en su lugar, añadimos a este proyecto estos dos ficheros que acabamos de crear: *bd_notasAlumnos.xsd* y *bd_notasAlumnos.xml*. En realidad no es necesario añadirlos; lo hacemos simplemente a efectos de modificaciones. Donde sí tienen que estar es en la carpeta desde la que se ejecute la aplicación WPF que haga referencia a esta biblioteca que hemos generado; en nuestro caso en *EnlaceDeDatosXML*.

Para construir un objeto **DataSet** a partir de los ficheros *bd_notasAlumnos.xsd* y *bd_notasAlumnos.xml*, añadimos a la biblioteca *BDNotasAlumnos* una nueva clase, *DataSetBDNotasAlumnos*, que almacenamos en el fichero *bd_notasAlumnosDataSet.cs*:

```
public class DataSetBDNotasAlumnos
{
  // Construcción del DataSet a partir de
  // bd_notasAlumnos.xsd y bd_notasAlumnos.xml
  public static DataTable ObtenerTablaAsignaturas()
  {
    return ReadDataSet().Tables["asignaturas"];
  }

  public static DataSet ObtenerAlumsAsigsNotas()
  {
    return ReadDataSet();
  }

  internal static DataSet ReadDataSet()
  {
    try
    {
      DataSet ds = new DataSet();
      ds.ReadXmlSchema("../../bd_notasAlumnos.xsd");
```

```
      ds.ReadXml("../../bd_notasAlumnos.xml");
      return ds;
    }
    catch (Exception exc)
    {
      System.Diagnostics.Debug.WriteLine(exc.Message);
      return null;
    }
  }
}
```

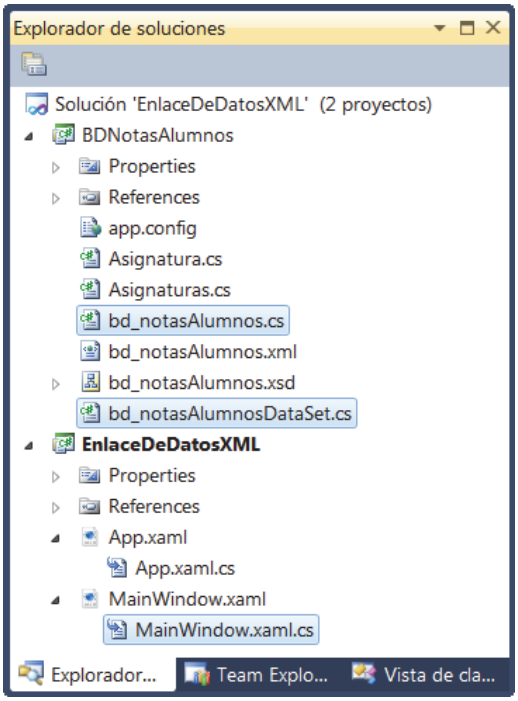
Y para implementar la lógica de negocio que permita acceder a la base de datos añadimos otra clase más, *bd_notasAlumnos*, que almacenamos en el fichero *bd_notasAlumnos.cs*:

```
public class bd_notasAlumnos
{
  public Asignatura ObtenerAsignatura(int ID)
  {
    try
    {
      DataTable tablaAsigs =
        DataSetBDNotasAlumnos.ObtenerTablaAsignaturas();
      DataRow filaAsignaturas =
        tablaAsigs.Select("id_asignatura = " + ID.ToString())[0];
      Asignatura asig =
        new Asignatura((int)filaAsignaturas["id_asignatura"],
                       (string)filaAsignaturas["nombre"]);
      return asig;
    }
    catch (Exception exc)
    {
      System.Diagnostics.Debug.WriteLine(exc.Message);
      return null;
    }
  }

  public ICollection<Asignatura> ObtenerAsignaturas()
  {
    DataTable tablaAsigs =
      DataSetBDNotasAlumnos.ObtenerTablaAsignaturas();

    ObservableCollection<Asignatura> asigs =
      new ObservableCollection<Asignatura>();

    foreach (DataRow fila in tablaAsigs.Rows)
    {
      asigs.Add(new Asignatura((int)fila["id_asignatura"],
                               (string)fila["nombre"]));
    }
```

```
        return asigs;
    }

    // ...
}
```

Esta clase requiere añadir también la clase *Asignatura* que implementaremos de la misma forma que lo hicimos en el capítulo *Enlace de datos en WPF*.

En la figura siguiente se puede observar ambos proyectos, el correspondiente a la biblioteca, *BDNotasAlumnos*, y el correspondiente a la aplicación WPF, *EnlaceDeDatosXML*, que utilizará esa biblioteca.



VISUAL WEB DEVELOPER

Visual Web Developer Express (VWD) es una herramienta enfocada exclusivamente al desarrollo de aplicaciones Web dinámicas con ASP.NET. Para ello, proporciona:

- Diseñadores visuales para crear las páginas Web.

- Un editor de código potente que muestra referencias del lenguaje a medida que se escribe código, sin tener que dejar el editor. Con este editor podrá escribir código en los diferentes lenguajes .NET (Visual VB o C#) y código HTML.

- Desarrollo rápido de aplicaciones Web integradas con bases de datos. Para ello proporciona acceso integrado a SQL Server Express.

Crear un sitio Web con VWD y probar su funcionamiento es algo sencillo porque integra un servidor Web, por lo que no es necesario desplegar la aplicación en el servidor IIS de Windows (*Internet Information Server*). En realidad, cualquier carpeta con páginas *.aspx* puede ser considerada virtualmente como un sitio Web. Este servidor Web integrado será utilizado automáticamente cuando elijamos como tipo de sitio Web "sistema de archivos". Otros tipos de sitios Web son HTTP y FTP, ambos necesitan tener instalado IIS.

La forma de trabajar con esta herramienta es la misma que hemos explicado en los últimos capítulos de este libro para crear sitios y servicios Web con Visual Studio, ya que esta herramienta está integrada en Visual Studio.

INSTALACIÓN DE ASP.NET EN WINDOWS

ASP.NET queda instalado automáticamente cuando instaló Visual Studio o, en su defecto, Visual Web Developer. Ahora bien, para crear y ejecutar aplicaciones para Internet necesitará un servidor de aplicaciones, en el caso de la plataforma Windows éste es IIS (*Internet Information Server*). La instalación de IIS debe ser anterior a la de .NET Framework. Si no se hizo así, ASP.NET no estará habilitado en IIS y no podremos realizar aplicaciones ASP.NET. En este caso, la solución es registrar manualmente ASP.NET en IIS.

Registro manual de ASP.NET en IIS

Para registrar manualmente ASP.NET en IIS, abra una ventana de consola, sitúese en la carpeta *C:\WINDOWS\Microsoft.NET\Framework\vx.x.xxxxx* y ejecute:

```
aspnet_regiis.exe -i -enable
```

Esto habilitará IIS para la ejecución de ASP.NET, registrará las extensiones de ASP.NET (*aspx*, *amsx*, *asax*, etc.) y ya podremos empezar a trabajar con ASP.NET.

Si echa una ojeada a todas las opciones disponibles (*aspnet_regiis.exe -help*) podrá observar algunas muy interesantes, como *-c*, que instala las secuencias de

órdenes del cliente de esta versión en el subdirectorio *aspnet_client* de todos los directorios de sitios IIS.

Si la operación anterior no solucionó el problema, pruebe a reparar la instalación actual a partir del CD de instalación del producto. Si esto tampoco solucionara el problema, sólo queda desinstalar el producto y volverlo a instalar.

APÉNDICE B

© F.J.Ceballos/RA-MA

CD

La imagen del CD de este libro, con las aplicaciones desarrolladas y el software para reproducirlas, puede descargarlo desde *http://www.fjceballos.es* (sección *Mis publicaciones* > *C#* > *CD*) o desde *http://www.ra-ma.com* (en la página correspondiente al libro). La descarga consiste en un fichero ZIP con una contraseña `ddd-dd-dddd-ddd-d` que se corresponde con el ISBN de este libro localizado en la página IV del mismo (teclee los dígitos y los guiones).

Cuando descomprima el fichero obtendrá la imagen ISO del CD que puede grabar en un soporte convencional ejecutando la orden *Grabar imagen*, o equivalente, de su programa de grabación de CD/DVD.

ÍNDICE

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

W

X

Y

Ra-Ma®

Del mismo autor

• Curso de programación con **PASCAL**	ISBN: 978-84-86381-36-3 224 págs.
• Curso de programación **GW BASIC/BASICA**	ISBN: 978-84-86381-87-5 320 págs.
• Manual para **TURBO BASIC** Guía del programador	ISBN: 978-84-86381-43-1 444 págs.
• Manual para **Quick C 2** Guía del programador	ISBN: 978-84-86381-65-3 540 págs.
• Manual para **Quick BASIC 4.5** Guía del programador	ISBN: 978-84-86381-74-5 496 págs.
• Curso de programación **Microsoft COBOL**	ISBN: 978-84-7897-001-8 480 págs.
• Enciclopedia del lenguaje **C**	ISBN: 978-84-7897-053-7 888 págs.
• Curso de programación **QBASIC y MS-DOS 5**	ISBN: 978-84-7897-059-9 384 págs.
• Curso de programación **RM/COBOL-85**	ISBN: 978-84-7897-070-4 396 págs.
• El abecé de **MS-DOS 6**	ISBN: 978-84-7897-114-5 224 págs.
• Microsoft **Visual C ++** (ver. 1.5x de 16 bits) Aplicaciones para Windows	ISBN: 978-84-7897-180-0 846 págs. + 2 disquetes
• Microsoft **Visual C ++** Aplicaciones para Win32 (2ª edición)	ISBN: 978-84-7897-561-7 792 págs. + disquete
• Microsoft **Visual C ++** Programación avanzada en Win32	ISBN: 978-84-7897-344-6 888 págs. + CD-ROM
• **Visual Basic 6** Curso de programación (2ª edición)	ISBN: 978-84-7897-357-6 528 págs. + disquete
• Enciclopedia de Microsoft **Visual Basic 6**	ISBN: 978-84-7897-386-6 1.072 págs. + CD-ROM
• El lenguaje de programación **Java**	ISBN: 978-84-7897-485-6 320 págs. + CD-ROM
• El lenguaje de programación **C#**	ISBN: 978-84-7897-500-6 320 págs. + CD-ROM

Del mismo autor

• El lenguaje de programación **Visual Basic.NET**	ISBN: 978-84-7897-525-9 464 págs. + CD-ROM
• **Java 2** Lenguaje y aplicaciones	ISBN: 978-84-7897-745-1 392 págs. + CD-ROM
• Programación orientada a objetos con **C ++** (4ª edición)	ISBN: 978-84-7897-761-1 648 págs. + CD-ROM
• **C/C++** Curso de programación (3ª edición)	ISBN: 978-84-7897-762-8 708 págs. + CD-ROM
• **Microsoft C#** Lenguaje y aplicaciones (2ª edición)	ISBN: 978-84-7897-813-7 520 págs. + CD-ROM
• **Java 2**. Interfaces gráficas y aplicaciones para Internet (3ª edición)	ISBN: 978-84-7897-859-5 718 págs. + CD-ROM
• **Aplicaciones .Net multiplataforma** (Proyecto Mono)	ISBN: 978-84-7897-880-9 212 págs. + CD-ROM
• Enciclopedia del lenguaje **C ++** (2ª edición)	ISBN: 978-84-7897-915-8 902 págs. + CD-ROM
• Enciclopedia de Microsoft **Visual C#** (3ª edición)	ISBN: 978-84-7897-986-8 1.110 págs. + CD-ROM
• Enciclopedia de Microsoft **Visual Basic** (2ª edición)	ISBN: 978-84-7897-987-5 1.090 págs. + CD-ROM
• **Microsoft Visual Basic .NET** Lenguaje y aplicaciones (3ª edición)	ISBN: 978-84-9964-020-4 520 págs. + CD-ROM
• **Java 2** Curso de programación (4ª edición)	ISBN: 978-84-9964-032-7 820 págs. + CD-ROM
• **Microsoft C#** Curso de programación (2ª edición)	ISBN: 978-84-9964-068-6 850 págs. + CD-ROM

INSTALACIÓN

Para instalar el kit de desarrollo de C# y los ejemplos de este libro descargue la plataforma de desarrollo de la dirección: *http://www.microsoft.com/express/*.

PLATAFORMA WINDOWS

Instalación de .NET Framework SDK

Hay que instalar .NET Framework Redistributable Package antes de instalar .NET Framework SDK. El primer paquete incluye todo lo necesario para ejecutar aplicaciones desarrolladas con .NET Framework. El segundo paquete incorpora todo lo necesario para escribir, construir, verificar y desplegar aplicaciones desarrolladas con.NET Framework.

Para realizar la instalación, siga las instrucciones mostradas por el asistente de instalación. Esta instalación se realiza en la carpeta *Microsoft.NET* de Windows y en la carpeta *Microsoft.NET* de archivos de programas.

Instalación de Microsoft Visual Studio

La instalación de Visual Studio (el paquete completo o las versiones *Express* que necesite) no requiere la instalación previa del SDK porque está incluido en éste. Descargue de Internet los paquetes Visual C# 2010 Express y Visual Web Developer 2010 Express (ambos incluyen SQL Server Express y la ayuda en línea), o bien, si tiene acceso, Visual Studio 2010, e instálelos.

Ejemplos del libro

Los ejemplos del libro puede instalarlos en la carpeta *Projects* de Visual Studio o los puede recuperar directamente desde el CD cuando los quiera consultar.

LICENCIA

WEB DEL AUTOR: http://www.fjceballos.es

En esta Web podrá echar una ojeada a mis publicaciones más recientes y acceder a la descarga del software necesario para el estudio de esta obra así como a otros recursos.