

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





Visual C#: Interfaces gráficas y aplicaciones para Internet con WPF, WCF y Silverlight.

© Fco. Javier Ceballos Sierra

© De la edición: RA-MA 2012

MARCAS COMERCIALES: las marcas de los productos citados en el contenido de este libro (sean o no marcas registradas) pertenecen a sus respectivos propietarios. RA-MA no está asociada a ningún producto o fabricante mencionado en la obra, los datos y los ejemplos utilizados son ficticios salvo que se indique lo contrario.

RA-MA es una marca comercial registrada.

Se ha puesto el máximo empeño en ofrecer al lector una información completa y precisa. Sin embargo, RA-MA Editorial no asume ninguna responsabilidad derivada de su uso, ni tampoco por cualquier violación de patentes ni otros derechos de terceras partes que pudieran ocurrir. Esta publicación tiene por objeto proporcionar unos conocimientos precisos y acreditados sobre el tema tratado. Su venta no supone para el editor ninguna forma de asistencia legal, administrativa ni de ningún otro tipo. En caso de precisarse asesoría legal u otra forma de ayuda experta, deben buscarse los servicios de un profesional competente.

Reservados todos los derechos de publicación en cualquier idioma.

Según lo dispuesto en el Código Penal vigente ninguna parte de este libro puede ser reproducida, grabada en sistema de almacenamiento o transmitida en forma alguna ni por cualquier procedimiento, ya sea electrónico, mecánico, reprográfico, magnético o cualquier otro, sin autorización previa y por escrito de RA-MA; su contenido está protegido por la Ley vigente que establece penas de prisión y/o multas a quienes intencionadamente, reprodujeren o plagiaran, en todo o en parte, una obra literaria, artística o científica.

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.....	XXI
Para quién es este libro.....	XXIII
Cómo está organizado el libro.....	XXIV
Qué se necesita para utilizar este libro	XXV
Sobre los ejemplos del libro	XXV
Agradecimientos	XXV
 CAPÍTULO 1. APLICACIÓN WPF	 1
PROGRAMANDO EN WINDOWS.....	3
BIBLIOTECA WPF	5
ESTRUCTURA DE UNA APLICACIÓN.....	6
XAML	8
¿Por qué XAML?	10
Código subyacente	11
INICIO DE LA APLICACIÓN.....	13
COMPILAR Y EJECUTAR LA APLICACIÓN	16
DISEÑO DE LA INTERFAZ GRÁFICA.....	18
Información básica sobre XAML.....	18
Espacios de nombres XML	19
Propiedades como atributos	20
Propiedades como elementos	21
Propiedades de contenido	21
Extensiones de marcado	22
Propiedades asociadas.....	24
Propiedades de dependencia	25

Crear un elemento	27
Controles más comunes	27
Añadir una etiqueta y editar sus propiedades.....	28
Añadir un botón de pulsación y editar sus propiedades.....	29
Añadir una descripción abreviada a un elemento	30
Paneles de diseño	30
Canvas.....	30
StackPanel.....	31
WrapPanel.....	32
DockPanel	33
Grid	34
MANEJO DE EVENTOS	37
Asignar manejadores de eventos a un objeto	38
EVENTOS ADJUNTOS	39
INYECTAR CÓDIGO XAML DURANTE LA EJECUCIÓN	39
CICLO DE VIDA DE UNA VENTANA	42
PROPIEDADES BÁSICAS DE LA VENTANA	44
Administración de la duración	44
Administración de ventanas	44
Apariencia y comportamiento	45
CONFIGURACIÓN DE UNA APLICACIÓN.....	47
RECURSOS DE UNA APLICACIÓN	48
ATRIBUTOS GLOBALES DE UNA APLICACIÓN.....	49
CICLO DE VIDA DE UNA APLICACIÓN.....	50
Permitir una sola instancia de la aplicación	53
Cómo se genera un evento	54
Especificar cuándo se cerrará la aplicación.....	54
Pantalla de presentación.....	55
Argumentos en la línea de órdenes	56
Acceso a la aplicación actual	57
RESUMEN.....	58
EJERCICIOS PROPUESTOS.....	58

CAPÍTULO 2. INTRODUCCIÓN A WPF 59

CLASES WPF.....	59
ETIQUETAS, CAJAS DE TEXTO Y BOTONES	63
Desarrollo de la aplicación.....	65
Objetos	65
Eventos.....	65
Pasos a seguir durante el desarrollo	66
El formulario, los controles y sus propiedades	66

Tecla de acceso	69
Botón predeterminado.....	69
Propiedades comunes.....	69
EVENTOS ENRUTADOS	71
¿Cómo se definen?	73
Responder a los eventos.....	74
Eventos relacionados con el teclado	76
Eventos relacionados con el foco	79
Seleccionar el texto de una caja de texto	80
Eventos relacionados con el ratón.....	82
INTERCEPTAR LA TECLA PULSADA	86
Estado del teclado	88
VALIDACIÓN DE UN CAMPO DE TEXTO	89
ENLACE DE DATOS	91
Enlace de datos sin el motor de WPF.....	92
Notificar cuándo cambia una propiedad	95
Enlace de datos con el motor de WPF	99
La clase Binding	100
Contexto de datos.....	101
Crear un enlace	102
Origen de datos implícito.....	103
Origen de datos explícito	105
Enlaces con otros controles.....	105
Conversores.....	106
Validación de datos.....	109
Regla ExceptionValidationRule.....	110
Regla DataErrorValidationRule.....	115
Información del enlace.....	116
Regla de validación personalizada.....	118
Fuentes relativas.....	120
ESTILOS Y PLANTILLAS.....	121
Estilos.....	121
Vincular controladores de eventos	123
Desencadenadores	123
Plantillas.....	125
Plantillas de control.....	125
Plantillas de datos	128
RECURSOS	128
Recursos creados mediante código	131
Recursos del sistema	131
TEMAS Y MÁSCARAS	133
RESUMEN.....	137
EJERCICIOS PROPUESTOS.....	138

CAPÍTULO 3. MENÚS Y BARRAS DE HERRAMIENTAS..... 139

ARQUITECTURA DE UNA BARRA DE MENÚS	139
DISEÑO DE UNA BARRA DE MENÚS	140
Crear una barra de menús.....	141
Controlador de un elemento de un menú	144
Aceleradores y nemónicos	146
ÓRDENES ENRUTADAS	147
Vincular una orden enrutada con un control	148
Modelo de una orden enrutada.....	151
Cómo se ejecuta una orden enrutada.....	152
Órdenes enrutadas personalizadas.....	155
Aceleradores de teclado	157
Información adicional en las órdenes enrutadas	161
¿Dónde se aplica la orden?.....	163
Utilizar parámetros.....	165
ICommand versus RoutedCommand	168
DETALLES DE UN ELEMENTO DE UN MENÚ	177
MENÚS CONTEXTUALES	178
BARRA DE HERRAMIENTAS.....	179
Diseño de una barra de herramientas	180
Contenedor de barras de herramientas	183
BARRA DE ESTADO	183
Diseño de una barra de estado.....	184
DESARROLLO DE UN EDITOR DE TEXTOS	186
Caja de texto multilínea	186
Diseño del editor	187
El portapapeles.....	189
Clase Clipboard.....	189
Manipular la selección del texto	190
Diseño de la barra de menús	191
Diseño de la barra de herramientas	193
Asignar a un elemento de la interfaz la tarea a realizar.....	195
Archivo - Salir.....	195
Editar - Cortar	196
Editar - Copiar.....	196
Editar - Pegar	197
Opciones - Fuente	198
Opciones - Tamaño	199
Ayuda – Acerca de.....	200
Eventos comunes a todos los elementos WPF	201
Habilitar o inhabilitar los elementos de un menú.....	202
Marcar el elemento seleccionado de un menú	204

Deshacer y rehacer	205
Recordar las ediciones reversibles	205
Añadir a la interfaz la orden Deshacer	205
Añadir a la interfaz la orden Rehacer	206
Menú contextual	207
Asociar un icono a la aplicación	207
MENÚS DINÁMICOS	207
RESUMEN	211

CAPÍTULO 4. CONTROLES Y CAJAS DE DIÁLOGO 213

CAJAS DE DIÁLOGO MODALES Y NO MODALES	214
CAJAS DE MENSAJE	214
CAJAS DE DIÁLOGO PERSONALIZADAS	217
Crear una caja de diálogo	219
Mostrar una caja de diálogo	221
Gestionar los botones Aceptar y Cancelar	222
Introducción de datos y recuperación de los mismos	223
DIÁLOGO ACERCA DE	225
VENTANA PROPIETARIA	227
OTROS CONTROLES WPF	228
Casillas de verificación	229
Botones de opción	234
GroupBox y Expander	239
Listas simples	240
Diseñar la lista	243
Iniciar la lista	244
Acceder a los elementos seleccionados	244
Colección de elementos de una lista	245
Lista de elementos de tipo CheckBox	247
Listas desplegables	249
Diseñar la lista	250
Iniciar la lista	251
Acceder al elemento seleccionado	251
Colección de elementos de una lista desplegable	251
Controles de rango definido	253
ScrollBar	253
Slider	257
ProgressBar	258
Visor con barras de desplazamiento	260
Control con pestañas	261
Gestión de fechas	262

ListView.....	264
TreeView.....	266
Guardar el documento XML.....	271
Recargar el documento XML.....	271
Expandir o contraer los nodos.....	272
DataGrid.....	273
Columnas del DataGrid.....	274
Inmovilizar columnas.....	276
Filas del DataGrid.....	276
Detalles de las filas.....	278
CAJAS DE DIÁLOGO ESTÁNDAR.....	280
Cajas de diálogo Abrir y Guardar.....	280
Cajas de diálogo Windows Forms estándar.....	283
Caja de diálogo Imprimir.....	284
CONTROLES DE DOCUMENTOS WPF.....	285
Documentos dinámicos.....	286
Elementos Block.....	287
Elementos Inline.....	291
Paragraph y Run.....	293
Interactuando con los elementos mediante programación.....	293
Acceso a documentos en un fichero.....	299
Editar un documento.....	300
Imprimir un documento.....	303
TEMPORIZADORES Y MODELO DE SUBPROCESOS.....	305
Timer.....	307
Resolución del temporizador.....	309
DispatcherTimer.....	310
RESUMEN.....	311
EJERCICIOS PROPUESTOS.....	312

CAPÍTULO 5. ENLACE DE DATOS EN WPF..... 319

ASPECTOS BÁSICOS.....	319
ENLACE A COLECCIONES DE OBJETOS.....	321
Cómo implementar colecciones.....	321
Vistas de colección.....	322
PLANTILLAS DE DATOS.....	324
Definir una plantilla de datos.....	326
Mejorar la presentación.....	328
Utilizar desencadenadores para aplicar valores de propiedad.....	329
XML COMO FUENTE DE DATOS.....	331
Datos jerárquicos.....	332

Islas de datos	335
Soporte .Net para trabajar con XML	335
Obtener la vista	337
Elemento actual	337
Navegar	338
Ordenar	339
Filtrar	340
Agrupar	341
Fuente de datos XML sin el proveedor	341
Vinculación maestro-detalle	343
OBJETOS COMO FUENTE DE DATOS	347
Enlace a una colección de objetos	349
Vistas de colección de objetos	351
Obtener la vista	354
Elemento actual	354
Navegar	355
Ordenar	356
Filtrar	356
Agrupar	357
Insertar y borrar elementos de la colección	358
Vinculación maestro-detalle	358
Proveedor de datos de objetos	361
Virtualización	363
Datos introducidos por el usuario	364
Solicitar datos al usuario	365
Validación	370
Visualización de los errores de validación	372
Regla de validación personalizada	375
Permanecer en la caja de diálogo si hay errores	378
Grupos de enlaces	378
DataGrid	381
Columnas del DataGrid	383
Inmovilizar columnas	384
Filas del DataGrid	384
Selección de celdas	385
Detalles de las filas	386
Filtrado, agrupación y ordenación	388
Validación	388
RESUMEN	388

CAPÍTULO 6. ACCESO A UNA BASE DE DATOS 391

SQL	392
Crear una base de datos.....	392
Crear una tabla	392
Escribir datos en la tabla	394
Modificar datos de una tabla	394
Borrar registros de una tabla	395
Seleccionar datos de una tabla	395
Crear una base de datos.....	397
Base de datos Microsoft Access.....	397
Base de datos Microsoft SQL Server	399
ADO.NET	400
Componentes de ADO.NET.....	401
Conjunto de datos.....	402
Proveedor de datos	404
Objeto conexión	405
Objeto orden	407
Objeto lector de datos	407
Adaptador de datos	408
Modos de conexión	410
Probando una conexión.....	411
Servicio de conexiones.....	413
ACCESO CONECTADO A BASE DE DATOS.....	414
ATAQUES DE INYECCIÓN DE CÓDIGO SQL.....	417
Órdenes parametrizadas	421
Procedimientos almacenados	422
TRANSACCIONES.....	423
Transacción implícita TransactionScope	424
Transacciones explícitas.....	428
CONSTRUIR COMPONENTES DE ACCESO A DATOS.....	430
Capa de presentación	432
Operaciones contra la base de datos.....	434
Objetos de negocio.....	435
Capa de acceso a datos.....	437
Capa de lógica de negocio	443
Lógica de interacción con la capa de presentación	444
Desacoplar la IU del resto de la aplicación	448
Adaptar la colección de objetos	448
Capa de lógica de negocio	452
Lógica de interacción con la capa de presentación	456
Validación	457
ACCESO DESCONECTADO A BASE DE DATOS	461

Crear la base de datos.....	465
Crear un proyecto WPF.....	465
Conectarse a la base de datos Sql Server	466
Crear la capa de acceso a datos	467
Capa de lógica de negocio	471
Lógica de interacción con la capa de presentación	473
Actualizaciones	475
Clase DataView.....	477
RESUMEN.....	480

CAPÍTULO 7. LINQ..... 481

RECURSOS DEL LENGUAJE COMPATIBLES CON LINQ.....	481
Declaración implícita de variables locales	482
Matrices de tipos definidos de forma implícita.....	482
Tipos anónimos.....	482
Propiedades auto-implementadas.....	483
Iniciadores de objetos y colecciones	483
Métodos extensores.....	484
Expresiones lambda	485
El delegado $\text{Func}\langle T, T\text{Resu}\rangle$	487
Operadores de consulta	488
Árboles de expresiones lambda.....	490
EXPRESIONES DE CONSULTA.....	493
Compilación de una expresión de consulta	497
Sintaxis de las expresiones de consulta.....	499
Cláusula group	499
Productos cartesianos.....	500
Cláusula join	500
Cláusula into	501
Cláusula let.....	502
PROVEEDORES DE LINQ	503
ENTITY FRAMEWORK	504
MARCO DE ENTIDADES DE ADO.NET	505
Consultar un modelo de objetos.....	509
ACCESO A UNA BASE DE DATOS.....	512
Conectarse a la base de datos	512
Generar el modelo de entidades	512
Las clases de entidad y el contexto de objetos	519
Propiedades de navegación	521
Mostrar datos en una interfaz gráfica.....	523
Una aplicación con interfaz gráfica.....	524

Vincular controles con el origen de datos	525
Filtros	530
Contextos de corta duración.....	532
REALIZAR CAMBIOS EN LOS DATOS.....	533
Modificar filas en la base de datos	536
Insertar filas en la base de datos.....	537
Borrar filas en la base de datos	540
Problemas de concurrencia	543
El seguimiento de cambios.....	547
EJERCICIOS RESUELTOS	550
RESUMEN.....	556
EJERCICIOS PROPUESTOS.....	557

CAPÍTULO 8. NAVEGACIÓN DE TIPO WEB..... 559

WPF, XBAP y Silverlight.....	559
NAVEGACIÓN	560
Crear la base de datos.....	561
Crear el proyecto.....	562
NavigationWindow	563
Page.....	565
Añadir páginas a la aplicación	566
Diseño de la interfaz gráfica	568
Lógica de negocio	576
Pasar datos entre páginas	577
Duración y diario de las páginas.....	578
Hyperlinks.....	580
Frame	582
Funciones de página.....	583
Diseño	586
Lógica de negocio	588
APLICACIÓN XBAP	590
Publicar la aplicación	592
Seguridad	595
ACCESO A UNA BASE DE DATOS DESDE UNA XBAP.....	597
Crear la base de datos.....	597
Conectarse a la base de datos	598
Generar el modelo de entidades	598
Interfaz gráfica	599
Vincular controles con el origen de datos	600
Controles de usuario.....	605
Modificar registros.....	607

Guardar los cambios realizados	609
Añadir un nuevo registro.....	610
Borrar un registro	616
EL CONTROL WEBBROWSER	617
RESUMEN.....	617
 CAPÍTULO 9. SILVERLIGHT	619
ARQUITECTURA.....	620
CREAR UNA APLICACIÓN SILVERLIGHT	622
Arquitectura de la aplicación Silverlight.....	623
Compilación de la aplicación Silverlight	627
Página de entrada	627
DISEÑAR UNA PÁGINA SILVERLIGHT	629
Controles Silverlight	629
Redistribuir el espacio de los elementos de un Grid	630
Texto estático	632
Imágenes	633
Controles de contenido	634
Atributos de anotación de datos.....	635
Diseño de la interfaz	636
Contexto de datos	638
TextBox	638
DescriptionViewer	639
ValidationSummary	639
Label	639
Validación de los datos	640
Origen de los datos	641
Controles de elementos	643
Controles de texto y elementos de texto	645
Controles de rango definido.....	647
Controles para gestionar fechas	648
Degradados	648
Ventanas y cajas de diálogo.....	650
Popup	650
ChildWindow.....	651
GRÁFICOS, ANIMACIÓN Y MULTIMEDIA	656
Gráficos.....	656
Transformaciones.....	658
Animaciones.....	662
Audio y vídeo.....	669
NAVEGACIÓN	680

Navegación personalizada.....	680
Navegación de Silverlight.....	681
Frame	683
Administrador de identificadores de recursos.....	687
Navegación externa.....	690
Extender el sistema de navegación	691
Compatibilidad de ejecución fuera del explorador	691
Plantilla aplicación de navegación de Silverlight	691
ACCESO A DATOS.....	692
Acceso a los datos de una colección	693
Crear la base de datos	697
Crear una aplicación Silverlight.....	699
Vincular controles con el origen de datos	700
Paginación controlada	703
Paginación personalizada	704
Filtrar los registros de la colección	709
Trabajar con imágenes	710
Cargar una nueva imagen.....	716
Guardar los cambios realizados	717
Añadir un nuevo registro.....	717
Borrar un registro	722
PUBLICAR LA APLICACIÓN.....	722
RESUMEN.....	724

CAPÍTULO 10. SERVICIOS WCF..... 727

MODELO DE PROGRAMACIÓN DE WCF	728
Implementar un servicio WCF	728
Definir un contrato	730
Implementar un cliente WCF	736
Configuración del cliente	740
Obtener acceso al servicio WCF	741
Comunicación entre dominios.....	745
Publicar la aplicación	747
SERVICIOS WCF HABILITADOS PARA SILVERLIGHT	750
Crear un servicio WCF habilitado para Silverlight.....	751
Implementar un cliente WCF	754
Añadir una referencia al servicio	755
Publicar la aplicación	757
SERVICIOS WEB Y LINQ.....	760
Arquitectura de N capas lógicas y N niveles físicos	761
Crear la base de datos.....	762

Obtener acceso a la base de datos	763
Crear el servicio WCF	765
Ciente Silverlight	774
Llenar la lista	778
Mensajes para el usuario	780
Ordenar la lista	780
Mostrar datos	781
Actualizar datos	783
Actualizar la foto	784
Agregar datos	785
Borrar datos	786
Publicar el servicio WCF y la aplicación Silverlight	787
RESUMEN	793

CAPÍTULO 11. AUTENTICACIÓN Y AUTORIZACIÓN 795

SERVICIOS DE AUTENTICACIÓN	796
Autenticación de Windows	797
Autenticación mediante formularios	797
Clase FormsAuthentication	799
Autenticación mediante formularios en Silverlight	800
SERVICIOS DE APLICACIÓN DE ASP.NET	808
Crear la estructura de la aplicación	810
Asignar y configurar servicios de aplicación	812
Crear usuarios	814
Autenticación	819
Funciones (roles)	822
Perfiles	827
Autorización de ASP.NET	827
SIMPLIFICAR EL DESARROLLO DE APLICACIONES	832
Plantilla aplicación de negocios Silverlight	834
Autenticación, funciones y perfiles	838
RESUMEN	842

CAPÍTULO 12. ACCESO A DATOS UTILIZANDO RIA SERVICES 843

ACCESO A DATOS	844
Crear y configurar la solución	845
Mostrar datos utilizando la clase LoadOperation	846
Generar el modelo de entidades	846
Agregar un servicio de dominio	847
LoadOperation	850

DomainDataSource	858
Parámetros de consulta	860
Ordenar, filtrar y agrupar	860
Paginación.....	860
Actualizar la base de datos.....	861
Añadir nuevos registros	866
Borrar registros	870
RESUMEN.....	871
APÉNDICE A. ENTORNO DE DESARROLLO INTEGRADO	873
MICROSOFT VISUAL STUDIO.....	873
Crear un nuevo proyecto	875
El formulario	879
Dibujar los controles	880
Borrar un control.....	884
Propiedades de los objetos	884
Icono de la aplicación	887
Escribir los controladores de eventos.....	887
Guardar la aplicación	890
Verificar la aplicación.....	890
Propiedades del proyecto	892
Crear soluciones de varios proyectos.....	893
Opciones del EDI.....	894
Personalizar el EDI	894
SQL SERVER EXPRESS.....	894
SQL SERVER MANAGEMENT STUDIO EXPRESS.....	897
EXPLORADOR DE BASES DE DATOS.....	899
AÑADIR UN DATASET AL PROYECTO	901
Esquemas XSD	904
Base de datos XML.....	904
VISUAL WEB DEVELOPER	908
INSTALACIÓN DE ASP.NET EN WINDOWS.....	909
Registro manual de ASP.NET en IIS	909
APÉNDICE B. CD.....	911
ÍNDICE	913

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 1. APLICACIÓN WPF

CAPÍTULO 2. INTRODUCCIÓN A WPF

CAPÍTULO 3. MENÚS Y BARRAS DE HERRAMIENTAS

CAPÍTULO 4. CONTROLES Y CAJAS DE DIÁLOGO

CAPÍTULO 5. ENLACES DE DATOS EN WPF

CAPÍTULO 6. ACCESO A UNA BASE DE DATOS

CAPÍTULO 7. LINQ

CAPÍTULO 8. NAVEGACIÓN DE TIPO WEB

CAPÍTULO 9. SILVERLIGHT

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/>

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*.

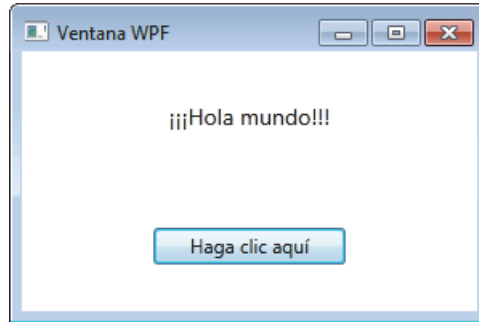
El propósito de este apéndice es mostrar cómo crear una aplicación sencilla de *Windows Presentation Foundation* (WPF) y familiarizarse con el entorno de desarrollo integrado (EDI) de *Visual Studio/Visual C# Express*. Al igual que las aplicaciones de formularios Windows Forms, las aplicaciones WPF se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

MICROSOFT VISUAL STUDIO

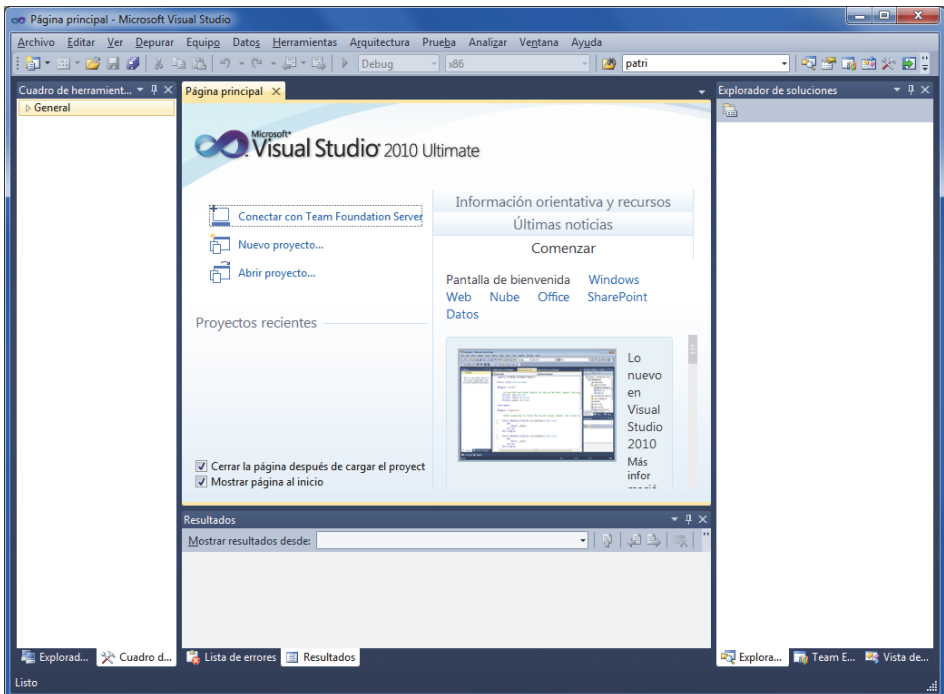
Visual Studio permite diseñar la interfaz gráfica de una aplicación WPF de manera visual, sin más que arrastrar con el ratón los controles que necesitemos sobre la ventana destino de los mismos. Unas líneas de guía o una rejilla mostrada sobre el formulario (o ventana) nos ayudarán a colocar estos controles 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:

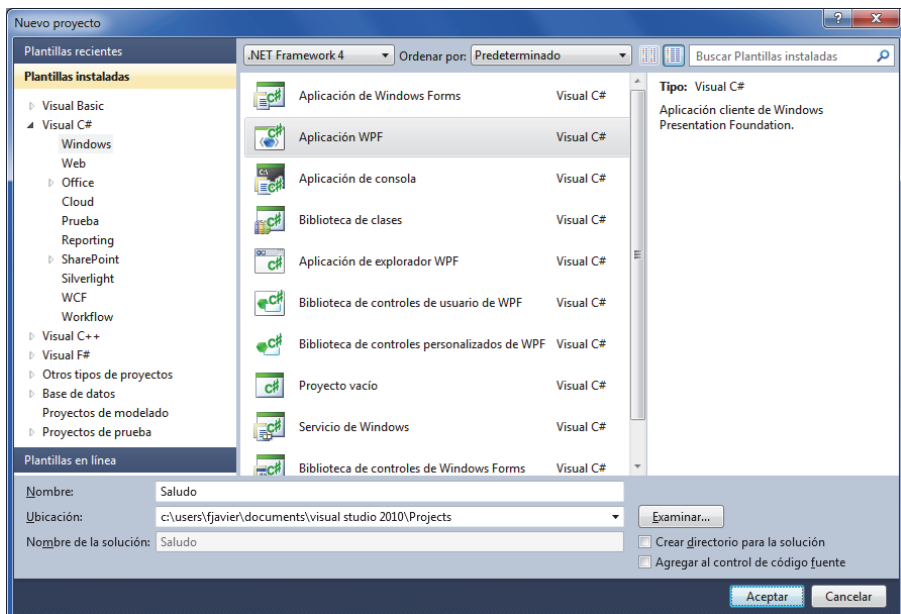


¿Cuáles son los siguientes pasos para desarrollar una aplicación WPF? En general, para construir una aplicación de este tipo con Visual Studio, siga los pasos indicados a continuación:

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.

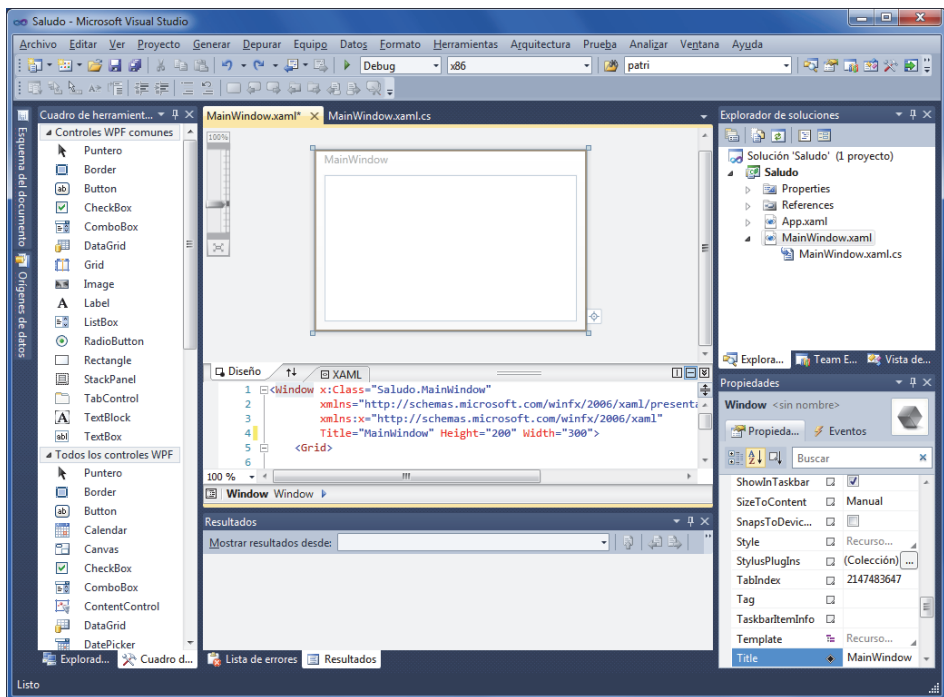
Crear un nuevo proyecto

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*:



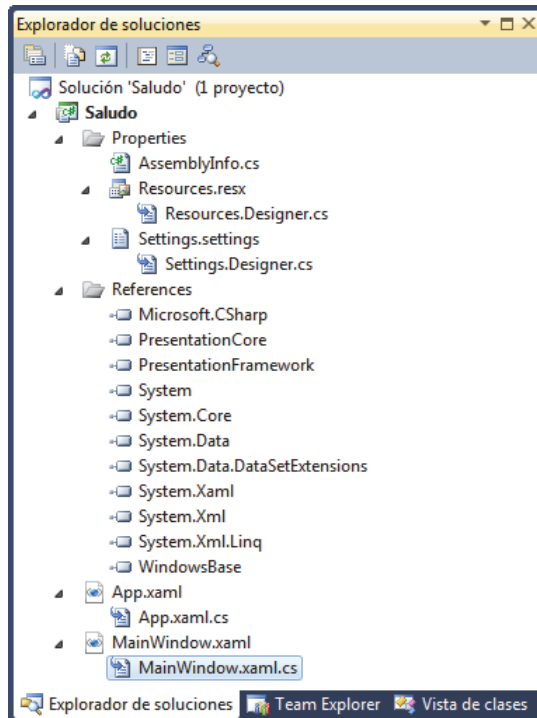
Obsérvese que se ha elegido la carpeta donde se almacenará el proyecto. Esta tarea puede posponerse sin que afecte al desarrollo de la aplicación. Ahora bien, si desea que esta tarea se realice automáticamente en el momento de crear el proyecto, caso del autor, ejecute *Herramientas > Opciones*, seleccione 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.



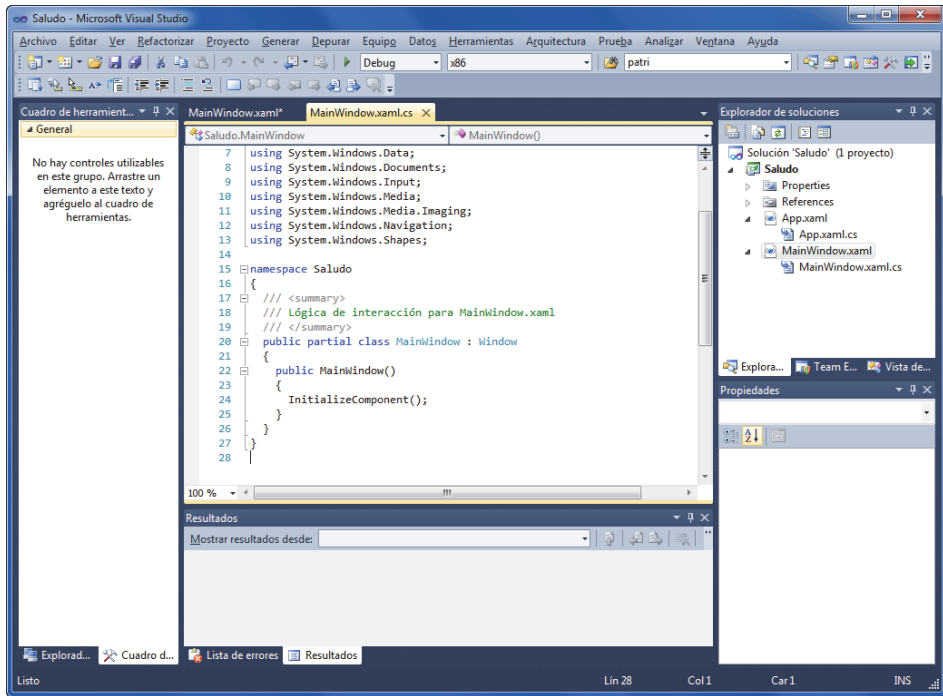
Otra característica interesante de este entorno de desarrollo es la ayuda que facilita. Basta con colocar el punto de inserción sobre una palabra clave y pulsar la tecla *F1* para que le muestre la ayuda correspondiente a la palabra seleccionada.

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.

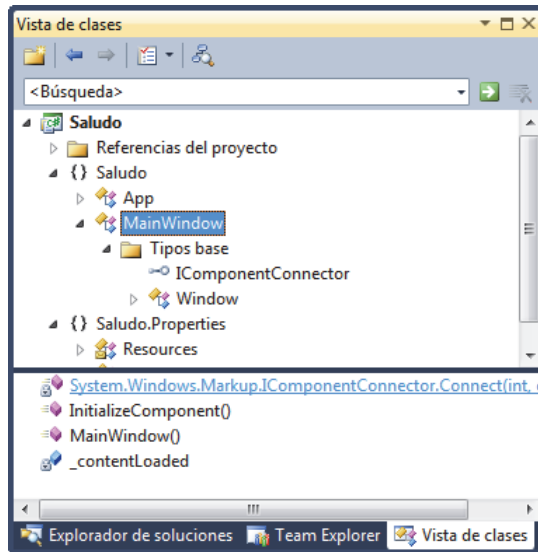
Así mismo, en su parte superior, muestra una barra de botones que permiten ver el *código*, el *diseñador* de formularios, la ventana de *propiedades*, etc. Por ejemplo, si estamos viendo el diseñador de formularios y hacemos clic en el botón *Ver código*, la página de diseño será sustituida por el editor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

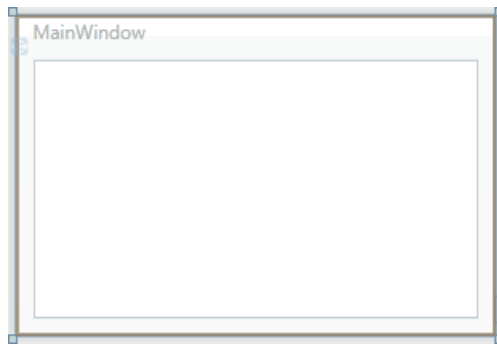
Si cuando se está visualizando el explorador de soluciones desea mostrar la vista de clases de su aplicación, sólo tiene que hacer clic en la pestaña *Vista de clases*. Esta ventana en su parte superior muestra las clases que componen la aplicación y en su parte inferior los métodos pertenecientes a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

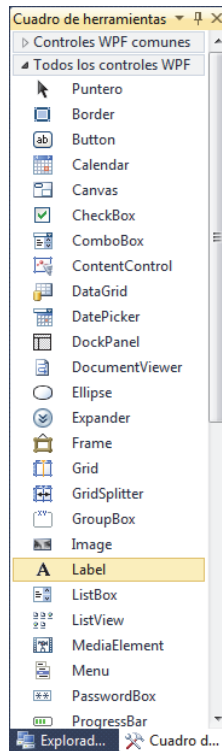
Para añadir un control a un formulario, utilizaremos la caja de herramientas que se muestra en la figura siguiente. Cada herramienta de la caja crea un único control. El significado de algunos de los controles más comunes se expone a continuación.

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.

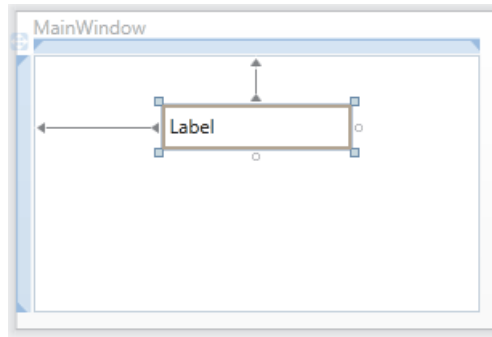
DatePicker. Control que permite seleccionar la fecha y hora.

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 (*RichTextBox*), 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.

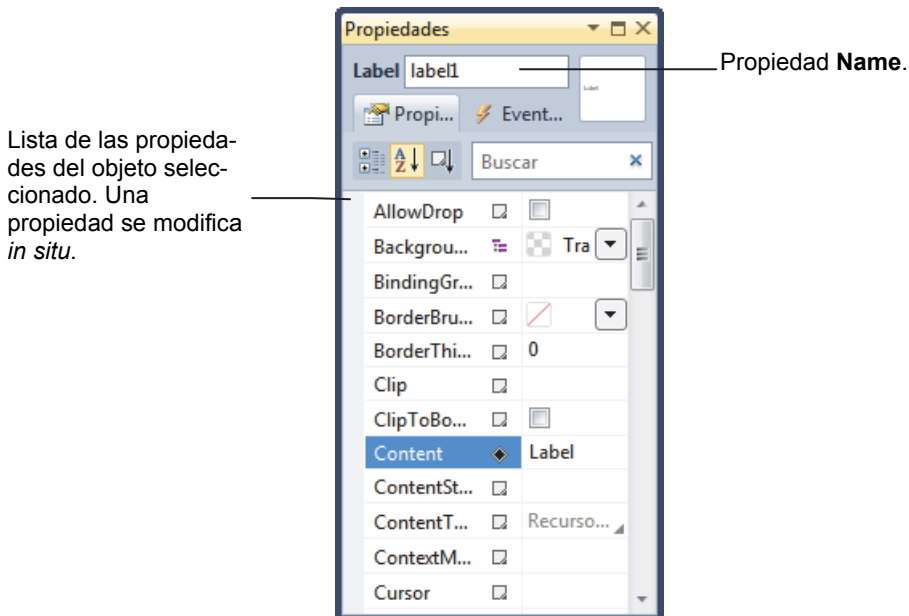
Siguiendo con nuestra aplicación, seleccionamos de la caja de herramientas que acabamos de describir los controles que vamos a utilizar. En primer lugar vamos a añadir al formulario una etiqueta. Para ello, hacemos clic sobre la herramienta etiqueta (*Label*) y, sin soltar el botón del ratón, la arrastramos 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.

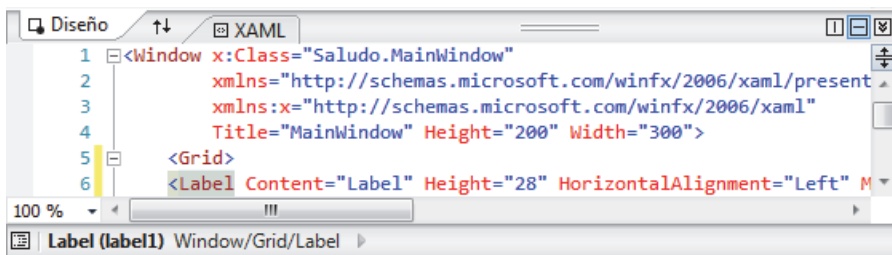


Estando la etiqueta seleccionada se observa sobre la misma un rectángulo con unos cuadrados distribuidos a lo largo de su perímetro, que reciben el nombre de

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.

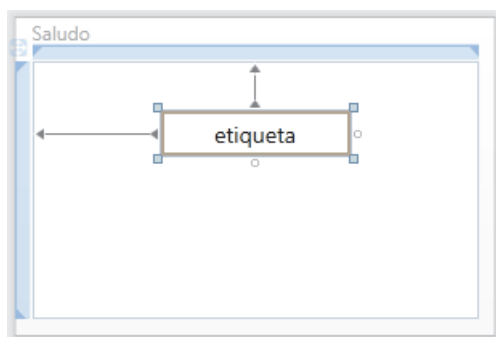
Se puede también modificar una propiedad durante la ejecución de la aplicación. Esto implica añadir el código necesario en el método que deba realizar la modificación.

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*.

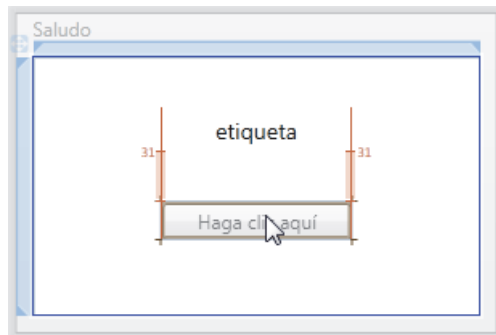
Siguiendo con nuestro ejemplo, vamos a cambiar el título del formulario. Para ello, seleccione el formulario y a continuación la propiedad **Title** en la página de propiedades. Después, sobrescriba el texto “*MainWindow*” con el texto “*Saludo*”.

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

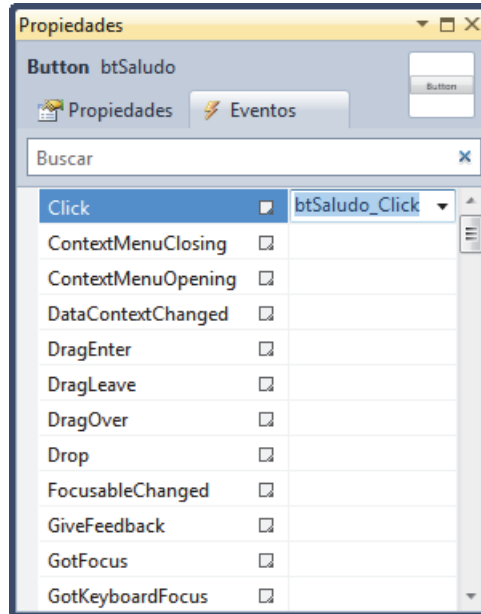
donde *Objeto* es el nombre del formulario o control y *Propiedad* es el nombre de la propiedad del objeto cuyo valor queremos asignar u obtener.

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*.

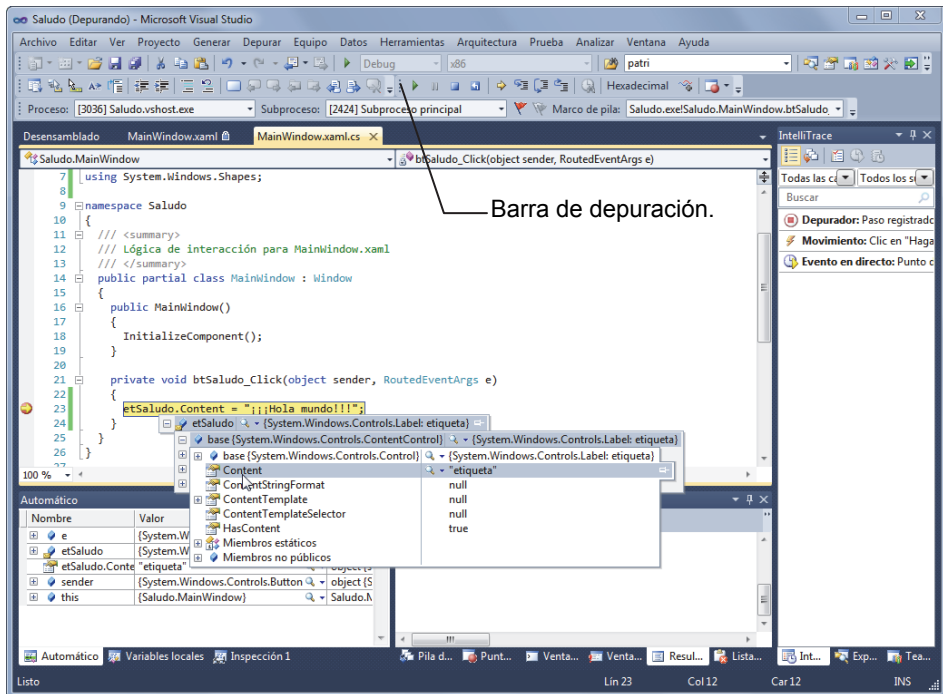
Si durante la ejecución encuentra problemas o la solución no es satisfactoria y no es capaz de solucionarlos por sus propios medios, puede utilizar, fundamentalmente, las órdenes *Paso a paso por instrucciones* (F11), *Paso a paso por procedimientos* (F10), *Alternar puntos de interrupción* (F9), todas 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.

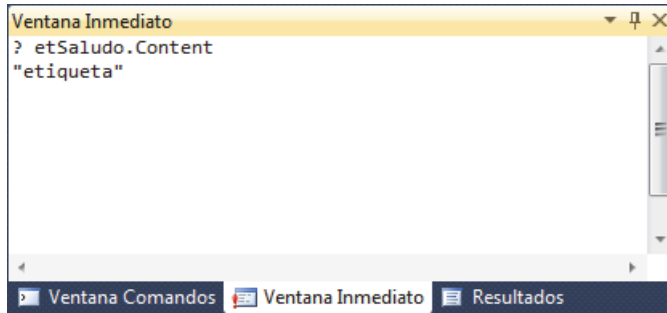


También puede utilizar el ratón para arrastrar el puntero de ejecución (observe la flecha en el margen izquierdo de la ventana anterior) a otro lugar dentro del mismo método con la intención de alterar el flujo normal de ejecución.

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.

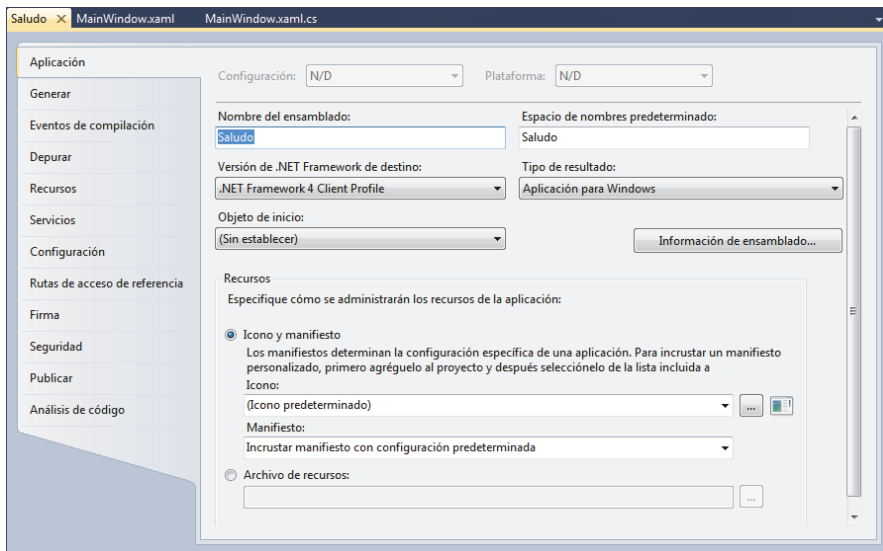
Así mismo, según se observa en la figura siguiente, puede ejecutar en la *Ventana Inmediato* cualquier sentencia de una forma inmediata. Para mostrar u ocultar esta ventana ejecute la orden *Ventanas > Inmediato* del menú *Depurar*. La figura siguiente muestra un ejemplo de cómo se puede inspeccionar 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:

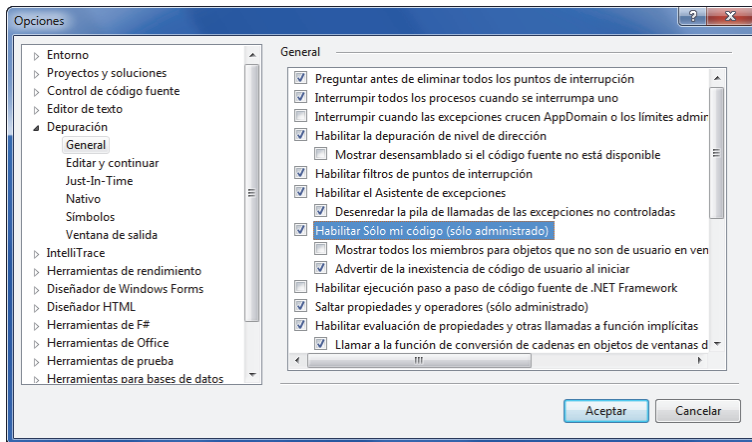
1. Ejecute la orden *Archivos > Nuevo > Proyecto*.
2. Como tipo de proyecto, seleccione *Otros Tipos de Proyectos*.
3. Y como plantilla, seleccione *Solución en Blanco*.
4. Finalmente, introduzca el nombre que desea dar a la solución. Se creará un fichero *.sln* con el nombre dado, almacenado en una 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"
```

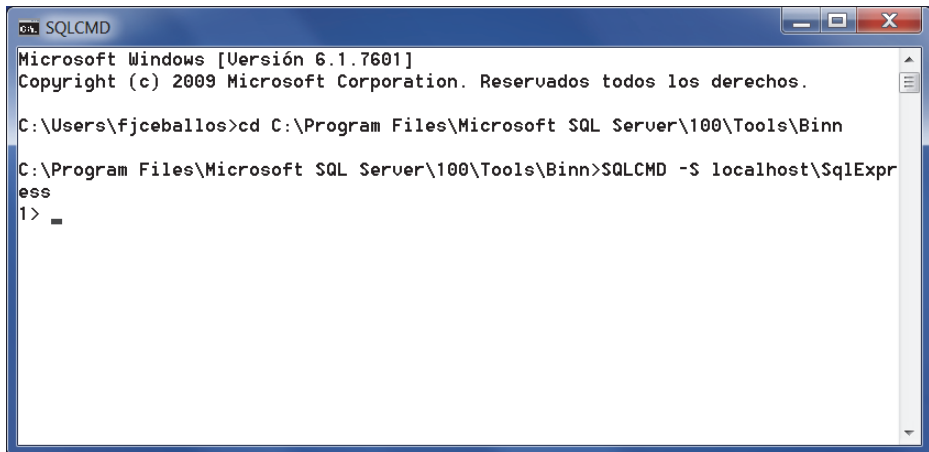
La entrada *Data Source* especifica el servidor de base de datos que vamos a utilizar y **AttachDBFilename**, la localización del fichero de base de datos. Obsérvese que la entrada *Initial Catalog* está vacía. O también, si la base de datos la copiamos en el directorio *App_Data* de una aplicación Web,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

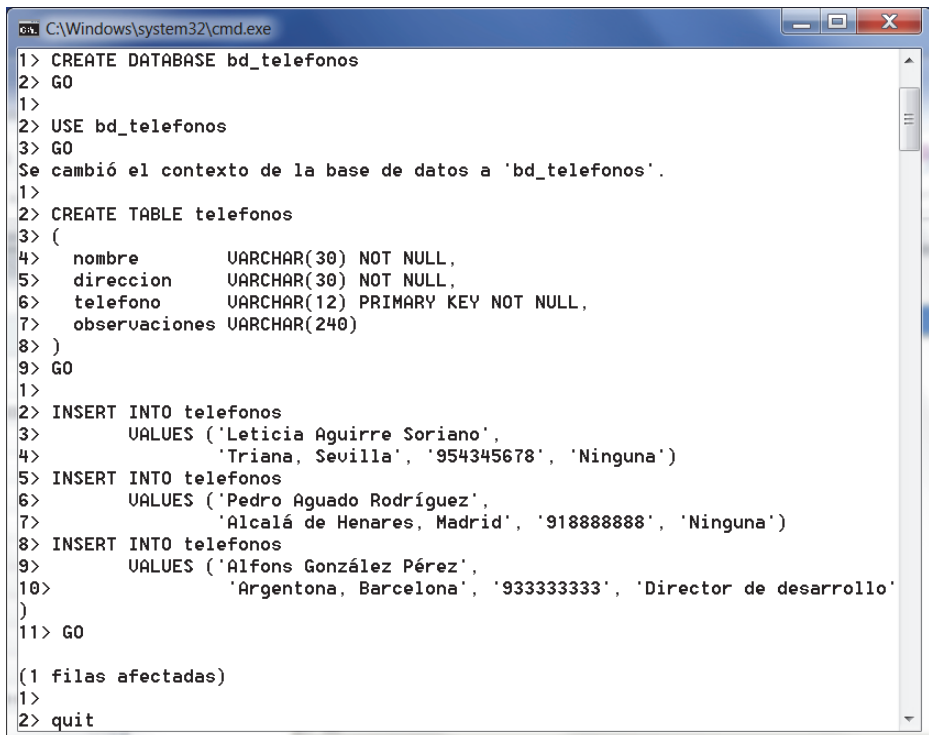


```

C:\Users\fjceballos>cd C:\Program Files\Microsoft SQL Server\100\Tools\Binn
C:\Program Files\Microsoft SQL Server\100\Tools\Binn>SQLCMD -S localhost\Sq1Expr
ess
1> _

```

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:

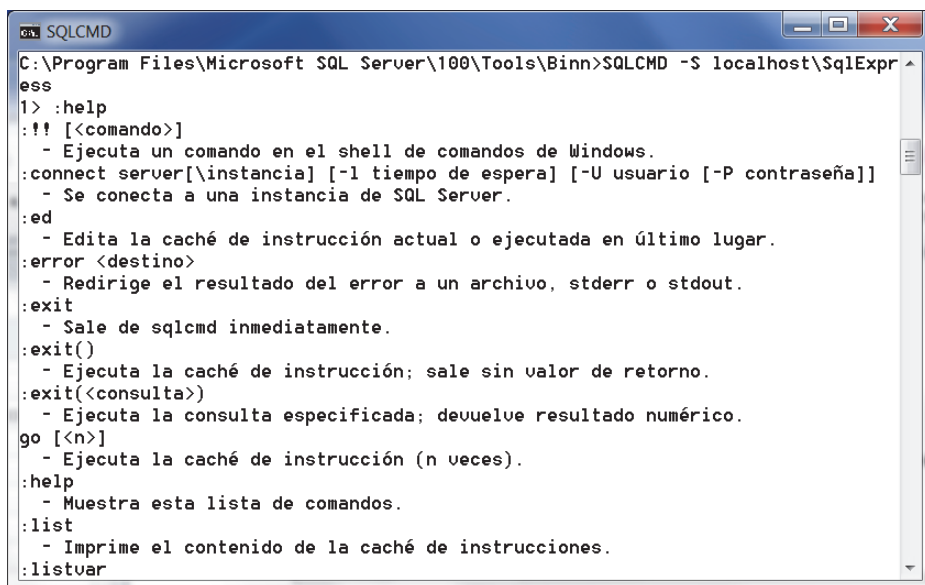


```

C:\Windows\system32\cmd.exe
1> CREATE DATABASE bd_telefonos
2> GO
1>
2> USE bd_telefonos
3> GO
Se cambió el contexto de la base de datos a 'bd_telefonos'.
1>
2> CREATE TABLE telefonos
3> (
4>     nombre          VARCHAR(30) NOT NULL,
5>     direccion        VARCHAR(30) NOT NULL,
6>     telefono         VARCHAR(12) PRIMARY KEY NOT NULL,
7>     observaciones    VARCHAR(240)
8> )
9> GO
1>
2> INSERT INTO telefonos
3>     VALUES ('Leticia Aguirre Soriano',
4>             'Triana, Sevilla', '954345678', 'Ninguna')
5> INSERT INTO telefonos
6>     VALUES ('Pedro Aguado Rodríguez',
7>             'Alcalá de Henares, Madrid', '918888888', 'Ninguna')
8> INSERT INTO telefonos
9>     VALUES ('Alfons González Pérez',
10>             'Argentona, Barcelona', '933333333', 'Director de desarrollo')
11> GO
(1 filas afectadas)
1>
2> quit

```

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 (:).



```

C:\Program Files\Microsoft SQL Server\100\Tools\Binn>SQLCMD -S localhost\SqLExpr
ess
1> :help
:!! [<comando>]
- Ejecuta un comando en el shell de comandos de Windows.
:connect server[<instancia>] [-I tiempo de espera] [-U usuario [-P contraseña]]
- Se conecta a una instancia de SQL Server.
:ed
- Edita la caché de instrucción actual o ejecutada en último lugar.
:error <destino>
- Redirige el resultado del error a un archivo, stderr o stdout.
:exit
- Sale de sqlcmd inmediatamente.
:exit()
- Ejecuta la caché de instrucción; sale sin valor de retorno.
:exit(<consulta>)
- Ejecuta la consulta especificada; devuelve resultado numérico.
go [<n>]
- Ejecuta la caché de instrucción (n veces).
:help
- Muestra esta lista de comandos.
:list
- Imprime el contenido de la caché de instrucciones.
:listvar
  
```

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:

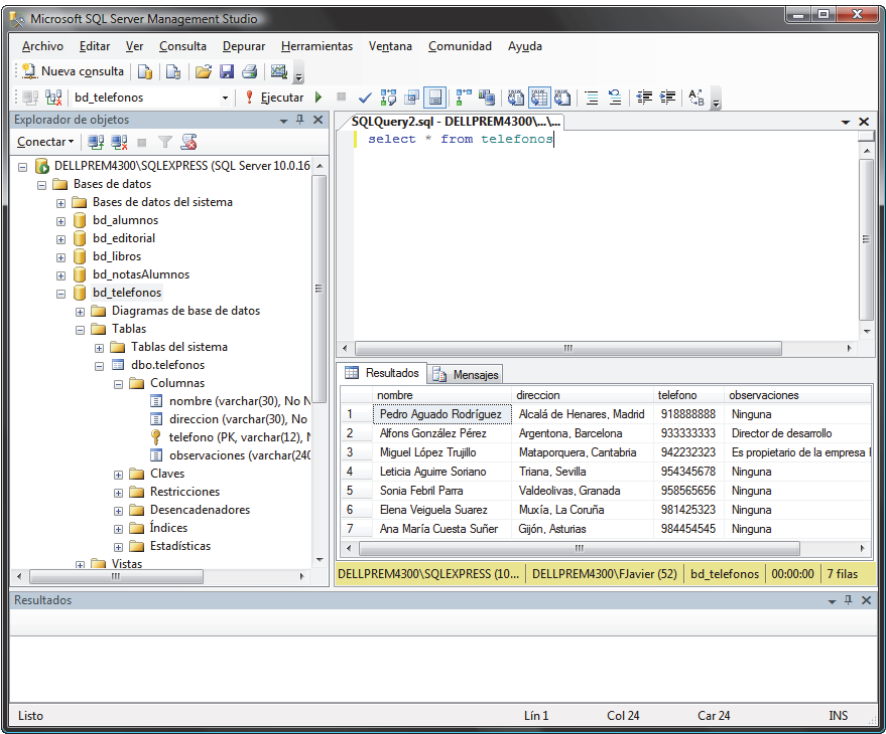
<http://www.microsoft.com/downloads/es-es/>

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*.

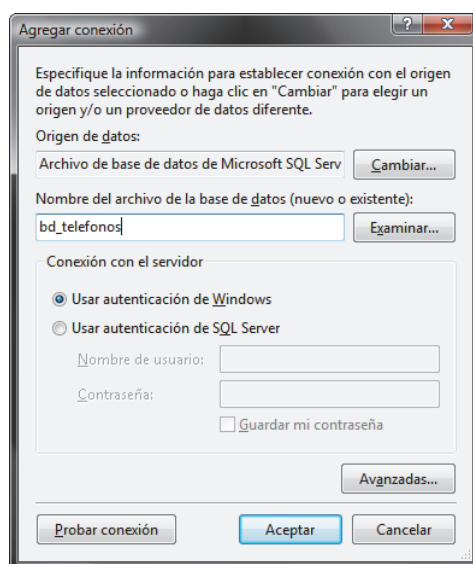
La imagen muestra una ventana de diálogo titulada "Crear nueva base de datos de SQL Server". El texto principal indica: "Introduzca la información necesaria para conectarse a SQL Server y, a continuación, especifique el nombre de la bas...".

La ventana está organizada en secciones:

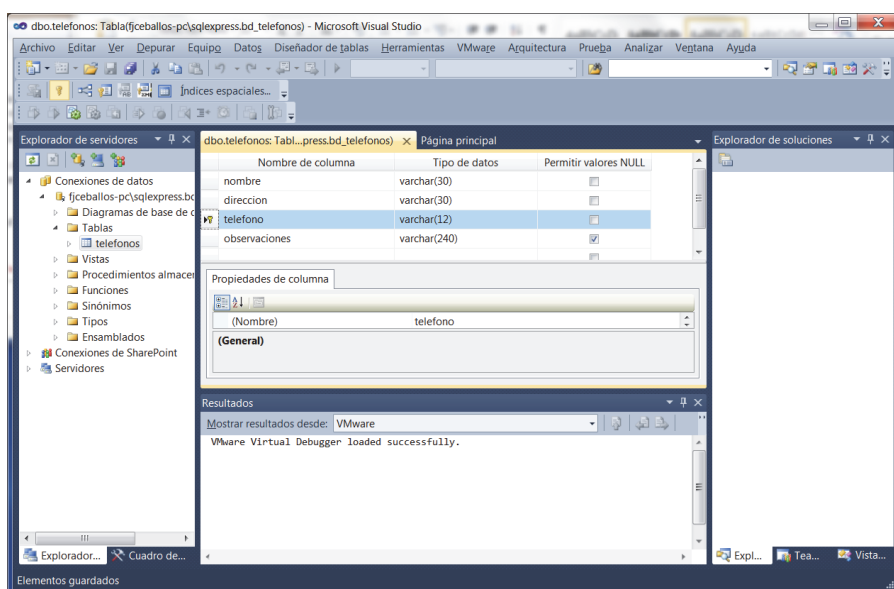
- Nombre del servidor:** Un campo de texto con un menú desplegable que muestra ".\sqlexpress" y un botón "Actualizar" a su derecha.
- Conexión con el servidor:** Una sección con dos opciones de radio:
 - ☒ Usar autenticación de Windows
 - ☐ Usar autenticación de SQL ServerDebajo de estas opciones hay dos campos de texto etiquetados como "Nombre de usuario:" y "Contraseña:", y un checkbox "Guardar mi contraseña" desmarcado.
- Nombre de la base de datos nueva:** Un campo de texto que contiene el texto "bd_telefonos".

En la parte inferior de la ventana hay dos botones: "Aceptar" y "Cancelar".

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*.



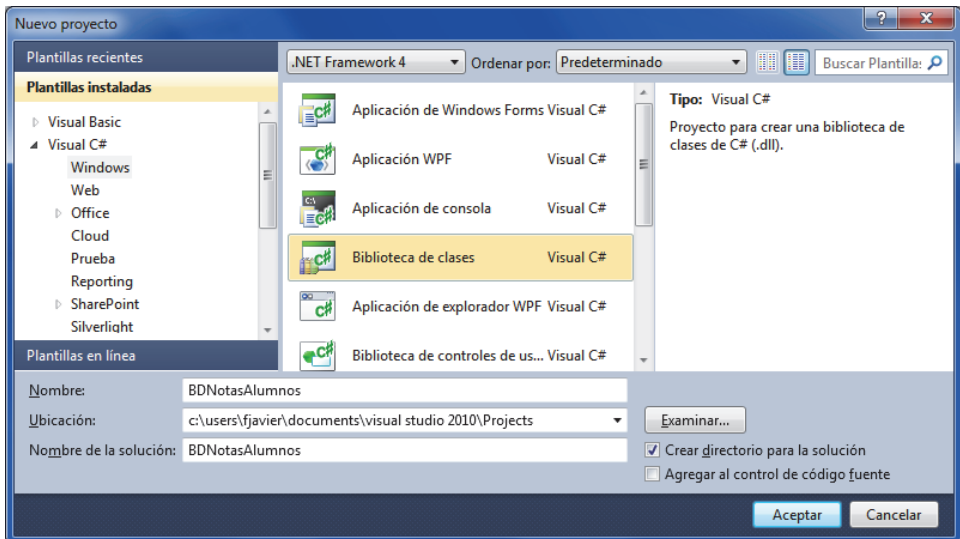
Una vez creada la base de datos, el paso siguiente es añadir las tablas. Despliegue el árbol correspondiente a la nueva conexión, haga clic con el botón secundario del ratón sobre el nodo *Tablas* y agregue una nueva tabla. Después complete el proceso de creación de la misma.



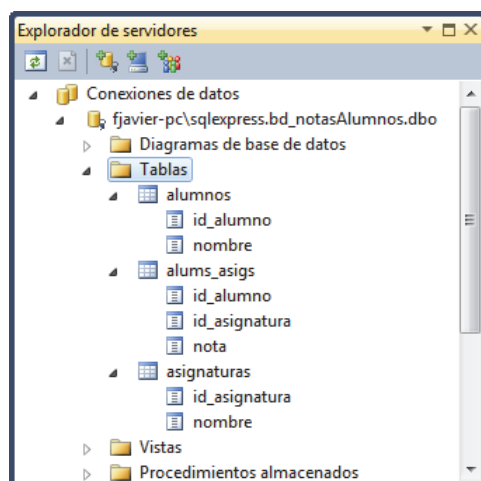
AÑADIR UN DATASET AL PROYECTO

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.



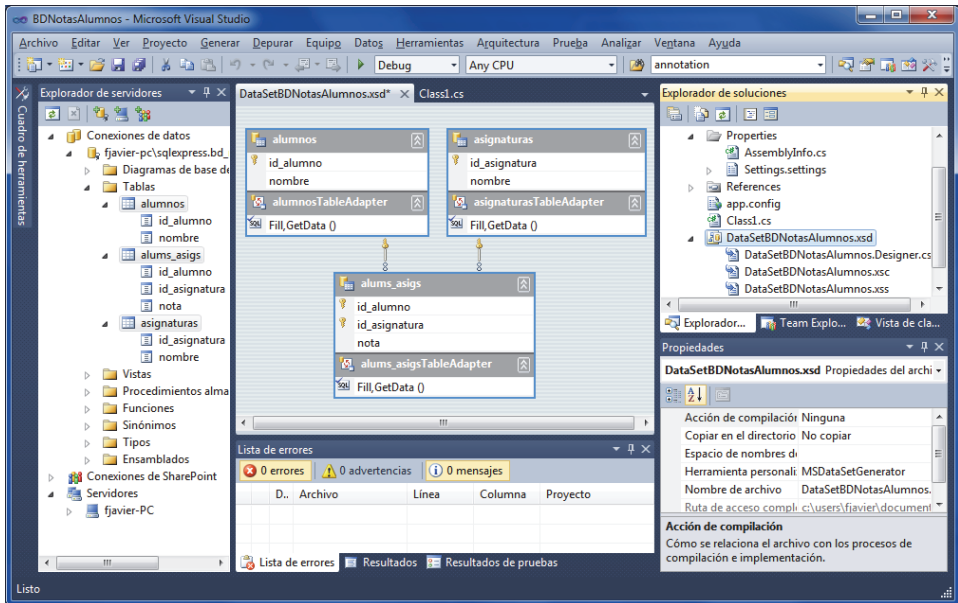
Suponiendo que tenemos creada la base de datos SQL, abrimos el explorador de servidores (explorador de base de datos) y añadimos una conexión a esta base de datos:



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.

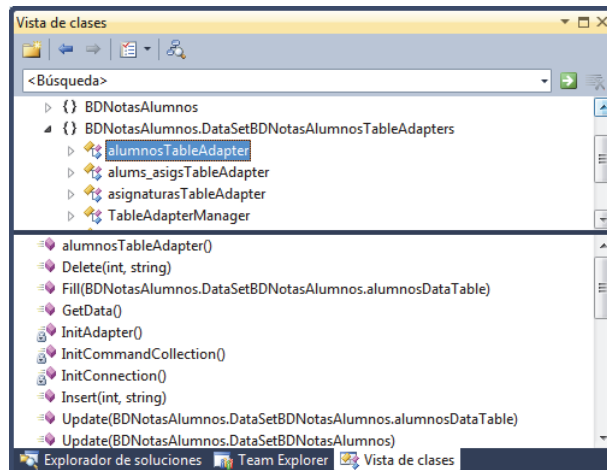
También puede realizar estas operaciones desde el menú *Datos* de Visual Studio.

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();

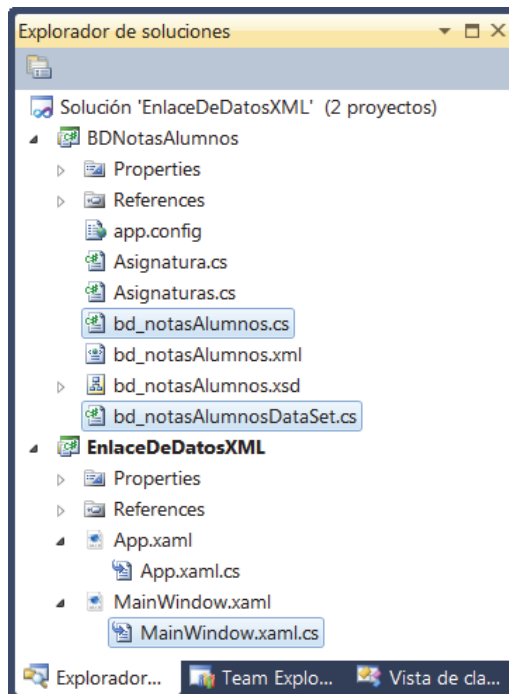
        ObservableCollection<Asignatura> asigs =
            new ObservableCollection<Asignatura>();

        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.xxxx* y ejecute:

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

Esto habilitará IIS para la ejecución de ASP.NET, registrará las extensiones de ASP.NET (*aspx*, *asmx*, *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 (teclea 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

#

#line, 12, 15

@

@, 635

A

AcceptsReturn, 187

AcceptsTab, 187

acceso a datos con RIA Services, 844

acceso conectado a bases de datos, 414

acceso desconectado a datos, 461

Access, 406

AccessText, 69

aceleradores, 146

Acerca de, 200, 225

Activate, 43, 44

Activated, 43, 50

actualizar datos, 783

actualizar la base de datos, 861

actualizar las filas, 536

adaptador, 408

Add, colección, 208

Added, 403

AddedItems, 446

AddHandler, 73, 678

AddNew, 785

AddObject, 537

administración de sitios Web, 816

ADO.NET, 400

AdornedElementPlaceholder, 126

agregar datos, 785

agrupación de los datos, 860

agrupar los elementos de la vista, 357

alineación del texto, 28

AllowsTransparency, 45

ámbito de foco, 151

animación, 330

animaciones, 662

animar Opacity, 330

animar una transformación, 665

anónimo, 531

añadir imagen a un elemento de un menú, 177

añadir nuevos objetos, 717

añadir texto a una caja de texto, 82

añadir un nuevo registro, 866

aplicación

 crear, 875

 de negocios de Silverlight, 834

 depurar, 890

 desarrollo, 65

 instanciar sólo una vez, 53

 mínima, 6

aplicaciones XBAP, 590

app.config, 48

AppendText, 191

Application, 14, 42, 60

Application.Current, 57

ApplicationServices, 809

árbol con nodos, 882

árboles de expresiones, 490

área cliente, 44

área de trabajo, 3
argumentos en la línea de órdenes, 56
asincrónico, modelo, 740
ASP.NET
 acceso a los recursos, 763
 arquitectura, 796
 instalación, 909
aspecto de un control, 125
aspnet_regiis, 909
Assembly, 50
atajo, 157
ataques de inyección SQL, 417
atributos de anotación de datos, 635
atributos globales de una aplicación, 49, 226
Attach, 771
AttachDBFilename, 895
audio, 669
autenticación, 795, 838
 de Windows, 796, 797, 836
 mediante formularios, 797, 836
 y autorización, 691
Authenticate, 805
AuthenticationService, 809, 812, 819
AutoCompleteBox, 645
AutoPlay, 670
autorización, 795, 827
ayuda dinámica, 876

BeginInit, 306
BeginTime, 665
binary, 763
BinaryExpression, 493
Binding, 22, 24, 100, 320, 693
BindingExpression, 117
BindingGroup, 379, 459
BindingListCollectionView, 324
Bitmap, 711
BitmapImage, 711
Block, 287
BlockUIContainer, 288
bool?, 222
Border, 27, 63, 328, 631
borrar datos, 786
borrar objetos, 722
borrar registros en una tabla, 395
borrar tabla de la base de datos, 395
borrar un control, 884
borrar un registro, 870
botón de opción, 234, 881
botón de pulsación, 29, 64, 881
botón predeterminado, 69
bucle de mensajes, 4, 306
Button, 27, 62, 64
ButtonBase, 62
byte[], 713

B

BAML, 18
barra de desplazamiento, 2, 253, 882
 eventos, 256
barra de estado, 183
barra de herramientas, 179
 añadir botón, 181
 diseño, 193
 orientación, 183
barra de menús, 1, 141
 crear, 141
 diseño, 191
barra de progreso, 258
barra de tareas, botón de la ventana, 43
barra de título, 2
base de datos, 391
 crear, 895, 899
 mover, 895
 XML, 904
BasedOn, 122
BeginEdit, 449

C

caché, 595
cadena de conexión, 830
caja de diálogo
 Abrir, 280
 Color, 283
 crear, 219
 Guardar, 282
 Imprimir, 284
 modal o no modal, 214
 mostrar, 221
 para mostrar un mensaje, 214
 recuperar datos, 223
caja de herramientas, 880
caja de texto, 881
 multilínea, 186
 seleccionar el contenido, 80
cajas de diálogo estándar, 280
cajas de diálogo personalizadas, 217
calculadora, 138
Calendar, 263

- cambios en los datos, 533
- cambios, seguimiento, 547
- campos, 391
- CancelEdit, 449
- CanExecute, 153, 176
- CanExecuteChanged, 154, 176
- CanUndo, 191, 205
- Canvas, 30, 63
- capa de acceso a datos, 437
- capa de lógica de negocio, 443
- capa de presentación, 432
 - desacoplar, 448
 - lógica, 444
- CaptureSource, 680
- carpeta virtual, 592
- cascada, operaciones en, 542
- casilla de verificación, 229, 881
- CenterOwner, 219
- cerrar, 2
- cerrar la aplicación, 54
- CheckBox, 27, 62, 229, 247
- Checked, 230
- CheckStateChanged, 234
- Children, 298
- ChildWindow, 651
- ciclo de vida de una aplicación, 50
- ciclo de vida de una ventana, 42
- cifrado, texto, 219
- cifrar, 801
- clase
 - Application, 14
 - Binding, 100, 320
 - Border, 27
 - Button, 27
 - CheckBox, 27, 229
 - Clipboard, 189
 - ColorDialog, 283
 - ComboBox, 27, 249
 - ContentElement, 59
 - Control, 27
 - Convert, 238
 - DataRow, 403
 - DataTable, 403
 - de entidad, 519
 - de un objeto, determinar, 296
 - Debug, 146
 - Dispatcher, 306
 - FrameworkContentElement, 59
 - FrameworkElement, 27, 59
 - GroupBox, 234
 - Label, 27
 - ListBox, 27, 240
 - Menu, 141
 - MenuItem, 141
 - Mouse, 82
 - NavigationWindow, 563
 - ObjectContext, 520
 - OpenFileDialog, 281
 - Page, 565
 - ProgressBar, 258
 - RadioButton, 27, 234
 - RoutedCommand, 148
 - SaveFileDialog, 282
 - ScrollBar, 27, 253
 - Separator, 141
 - Slider, 257
 - TextBlock, 27
 - TextBox, 27, 190
 - TextBoxBase, 190
 - TextRange, 301
 - TimeSpan, 310
 - UIElement, 59
 - Validation, 111
 - Window, 9
- clases, vista, 878
- Clear, 191, 245
- Click, 4, 78
- ClientAccessPolicy.xml, 745
- cliente Silverlight, 774
- cliente WCF, 736
- Clipboard, 189
- Close, 44, 146, 222, 408
- Closed, 43
- Closing, 43
- clr-namespace, 20, 104
- código C#, insertar en XAML, 13
- código subyacente, 11
- colección
 - añadir, 245
 - borrar todo, 245
 - borrar, 245
 - ConstraintCollection, 403
 - DataColumnCollection, 403
 - DataRelationCollection, 403
 - DataRowCollection, 403
 - DataTableCollection, 403
 - de elementos, 208, 882
 - insertar, 245
- colecciones, 321
- CollectionChanged, 358, 454

- CollectionView, 323, 336
 - CollectionViewSource, 323, 702
 - colocar el control, 886
 - color, 21, 283
 - Color, propiedad, 284
 - ColorAnimation, 663
 - ColorDialog, 283
 - Column, 34
 - columnas, 391
 - ColumnSpan, 34
 - ComboBox, 27, 62, 249
 - ComboBoxItem, 62, 249
 - Command, 150, 404, 407
 - CommandBinding, 153
 - CommandParameter, 165
 - commands, 147
 - CommandTarget, 151
 - CommitNew, 786
 - compilador de marcado de WPF, 11
 - Completed, 857
 - componentes de acceso a datos, 431
 - comunicación entre dominios, 745
 - concurrency, 543, 761, 844
 - ConcurrencyMode, 543
 - conexión abierta/cerrada, 520
 - conexión con la base de datos, 410
 - conexión, probar, 411
 - configuración de la aplicación, 47
 - configuración del cliente, 740
 - configuración servicio WCF, 757
 - ConfigurationManager, 830
 - confirmar, 216
 - conjunto de datos, 402
 - Connection, 404
 - ConstantExpression, 493
 - Constraint, 403
 - ConstraintCollection, 403
 - contenedor, 882
 - Content, 28, 564
 - ContentControl, 62, 634
 - ContentElement, 61
 - ContentLoader, 691
 - ContentRendered, 43
 - ContextMenu, 62, 178
 - contexto de datos, 101
 - contexto de dominio, 849
 - Load, 850
 - contexto de objetos, 519
 - contextos de corta duración, 533
 - contrato, 729
 - contrato de datos, 734
 - contrato de servicio, 730
 - Control, 27, 61
 - control adornado, 126
 - control de usuario, 605
 - control PasswordBox, 219
 - control, borrar, 884
 - control, mover, 884
 - controlador de eventos, 38, 74, 76
 - añadir, 77
 - controles, 3
 - controles de contenido, 634
 - controles de rango definido, 253
 - controles WPF, 228
 - ControlTemplate, 125
 - conversor, 106, 711
 - convertidores intrínsecos, 107
 - Convert, 79, 106, 238, 711
 - ConvertBack, 106, 711
 - ConverterCulture, 104
 - convertir un entero en una cadena de
 - caracteres en una base, 239
 - convertir una cadena de caracteres en una
 - base a un entero, 238
 - Copy, 191
 - Count, 488
 - crear una aplicación, 875
 - crear una base de datos, 392, 397, 399
 - crear una caja de diálogo, 219
 - crear una tabla, 392
 - CREATE DATABASE, 392
 - CREATE TABLE, 392
 - credentials, 800
 - CSDL, 506
 - cuadro combinado, 249
 - Current, 57
 - CurrentChanged, 708
 - CurrentItem, 337, 354, 708
 - CurrentSource, 564
 - Cut, 191
- D**
- Data Mapper, 511
 - Data Transfer Object, 432
 - DataAdapter, 404, 462
 - DataColumn, 403
 - DataColumnCollection, 403
 - DataContext, 101
 - DataContract, 734

DataErrorValidationRule, 110
 DataForm, 862
 DataGrid, 273, 432, 444, 701
 colección de objetos, 381
 columnas, 383
 detalles de las filas, 386
 eventos, 386
 filas, 384
 selección de celdas, 385
 validar, 457
 DataGridRow, 276, 384
 DataMember, 734
 DataPager, 703, 861
 DataReader, 404, 408
 DataRelationCollection, 403
 DataRow, 403, 469
 DataRowCollection, 403, 474
 DataSet, 461
 añadir al proyecto, 901
 métodos, 462
 DataSourceProvider, 337
 DataTable, 403, 469
 DataTableCollection, 403
 DataTemplate, 128, 266, 326
 DataTrigger, 124
 DataView, 471, 477
 DatePicker, 263
 datos jerárquicos, 332
 DbProviderFactory, 405
 Deactivated, 43, 50
 Debug, 146
 Decorator, 63
 deep link, 686
 deep zoom, 680
 degradados, 649
 delegado, 38, 709
 delegado Func, 487
 delegados, 38, 485
 DELETE, 395
 DeleteCommand, 409
 Deleted, 403
 DeleteObject, 540
 DependencyObject, 41, 60
 depuración, 146
 depurar una aplicación, 890
 descripción abreviada, 30
 DescriptionViewer, 639
 desencadenadores, 123, 329
 deshacer, 191, 205
 deslizador, 257

Detach, 762
 diálogo Acerca de, 225
 diálogo, introducir datos, 365
 diálogo, permanecer o cancelar, 378
 DialogResult, 222
 diccionario de recursos, 129, 133
 dirección, 729
 directorio virtual, 592
 diseño, 30
 diseño de tipo maestro-detalle, 343
 Dispatcher, 306
 DispatcherObject, 60, 306
 DispatcherTimer, 306, 310
 DispatcherUnhandledException, 51
 DisplayMemberPath, 426
 Dispose, 416
 DockPanel, 33, 63
 documento dinámico, 286
 en un fichero, 299
 modificar, 295
 navegar, 295
 documentos dinámicos, 285
 documentos fijos, 285
 DocumentViewer, 285
 DomainContext, 850
 DomainDataSource, 858
 DomainService, 847
 DoubleAnimation, 665
 DragMove, 45
 DROP TABLE, 395
 duración y diario de las páginas, 578
 DynamicResource, 23, 130

E

EDI, 873
 opciones, 894
 personalizar, 894
 EditingElementStyle, 458
 editor de textos, 186
 ejecutar, 17
 ejecutar aplicación, 880
 Elapsed, evento, 306, 307
 ElementName, 101, 105, 320
 elemento actual, 354
 elemento adornado, 126
 elemento de un menú, marcar, 205
 eliminar filas, 540
 Ellipse, 63
 EndEdit, 449

- enfocar un componente, 79
 - enlace, 729
 - a colecciones, 321
 - a una colección, 349
 - con otros controles, 105
 - de datos de WPF, 91
 - de datos WPF, 99
 - de datos, 319
 - de datos, crear, 102
 - información, 117
 - profundo, 686
 - TwoWay, 353
 - Entities, 853
 - Entity Client, 508
 - Entity Framework, 505
 - Entity SQL, 507
 - EntityCollection, 521
 - EntityKey, 548
 - EntityQuery, 853
 - EntityReference, 522
 - EntitySet, 548
 - entorno de desarrollo integrado, 873
 - Entrar, Aceptar, 220
 - enumeración HorizontalAlignment, 29
 - error durante el diseño, 601
 - ErrorContent, 114
 - Errors, 111
 - ErrorTemplate, 111, 126
 - Esc, Cancelar, 220
 - escribir datos en una tabla, 394
 - espacios de nombres, 17, 889
 - declarar en XAML, 104
 - XAML, 19
 - eSQL, 507
 - esquema del documento, 884
 - esquema XSD, 904
 - esquemas XML, 904
 - estilo, 121
 - estilos, 571
 - etiqueta, 28, 881
 - evento, 4
 - adjunto, 248
 - Checked, 230
 - Click, 78
 - controlado, 72
 - de propagación, directo y de túnel, 72
 - Initialized, 201
 - Loaded, 79, 199, 201
 - LoadingRow, 276
 - NavigationFailed, 581
 - PropertyChanged, 96
 - RowEditEnding, 447
 - SelectionChanged, 446
 - TextChanged, 95
 - Unchecked, 230
 - Unloaded, 201
 - Validation.Error, 112
 - eventos, 37, 71, 881
 - adjuntos, 39
 - de entrada, 72
 - de la aplicación, 50
 - de WPF, 73
 - del teclado, 76
 - enrutados, 71
 - lista, 888
 - suscribirse a, 74
 - EventSetter, 123
 - excepciones, 363
 - ExceptionValidationRule, 110, 371
 - Execute, 176
 - Executed, 153
 - ExecuteNonQuery, 407
 - ExecuteReader, 407
 - Exit, 50
 - explorador de bases de datos, 899
 - explorador de soluciones, 876
 - expresión de consulta, 490, 493
 - compilación, 497
 - expresiones lambda, 485
 - Expression, 491, 493
 - extensiones de marcado, 22
 - extremo, 729, 813
- ## F
- fecha y hora, 882
 - fechas, 262
 - fichas, 882
 - ficheros y documentos dinámicos, 299
 - Figure, 291
 - filas, 391
 - FileInfo, 717
 - FileName, 282
 - FileStream, 717
 - Fill, 410
 - Filter, 282, 709
 - FilterDescriptors, 860
 - FilterIndex, 282
 - filtrar datos, 709, 860
 - filtrar los elementos de una colección, 356

filtro de nombres de fichero, 282
 filtros, 478
 finalizar la ejecución, 17, 880
 finally, 413
 FindLogicalNode, 42
 FindName, 42
 FindResource, 105, 131
 FixedDocument, 285
 Floater, 291
 FlowDocument, 287
 FlowDocumentPageViewer, 286
 FlowDocumentReader, 286
 FlowDocumentScrollViewer, 286
 foco, 63, 79, 220
 Focus, 79, 88, 238
 Focusable, 80
 FocusedElement, 88
 FocusManager, 80, 151
 Font..., 28
 FontFamily, 198
 FontSize, 199
 Format, 79
 FormatException, 79
 FormsAuthentication, 799, 805
 formularios, 3
 Frame, 561, 582, 683
 FrameworkContentElement, 61
 FrameworkElement, 27, 42, 61
 from, 488, 496
 FromByteArray, 714
 FromResource, 715
 fuentes, 198
 fuentes relativas, 120
 Func, 487
 función de un usuario autenticado, 824
 funciones de página, 583
 funciones de usuario, 839

G

GetCustomAttributes, 50
 GetDefaultView, 337
 GetExecutingAssembly, 50
 GetHasError, 378
 GetNavigationService, 565
 GetObject, 48
 GetRolesForCurrentUser, 822
 GetString, 48
 GetWindow, 45
 GoBack, 564

GoForward, 564
 GotFocus, 80
 GotKeyboardFocus, 80
 gráficos, 656
 Grid, 34, 63, 630
 Grid.Column, 68
 Grid.Row, 68
 GridSplitter, 36, 630
 GridView, 264
 group, 499
 group ... by, 488
 GroupBox, 234
 GroupDescriptors, 860
 GroupName, 234
 grupos de aplicaciones, 789
 guardar una aplicación, 890

H

habilitar o inhabilitar los elementos de un
 menú, 202
 Handled, 72, 76, 85
 HasError, 111, 378, 381
 HashPasswordForStoringInConfigFile, 801
 HeaderedItemsControl, 62, 332
 Height, 69
 herramientas, caja, 880
 Hide, 44
 HierarchicalDataTemplate, 267, 332
 historial de navegación, 683
 HorizontalAlignment, 29, 70
 HorizontalContentAlignment, 29, 70
 hospedar aplicaciones WPF, 559
 Hyperlink, 566, 580
 HyperlinkButton, 685, 690

I

ICommand, 148, 168
 ICommandSource, 151
 Icon, 45, 177, 207, 887
 icono de la aplicación, 887
 icono, añadir a la aplicación, 207
 IDataErrorInfo, 110, 115
 identidad utilizada por el grupo de
 aplicaciones, 789
 IDENTITY, 763
 IDisposable, 416
 IEditableObject, 449
 IEnumerable, 489, 503

- Image, 177, 181, 194
- ImageBrush, 128
- imagen, añadir a un botón, 194
- imagen, cargar, 716
- imágenes, 633, 710
- ImageSource, 711
- imprimir, 284
- imprimir un documento dinámico, 303
- INavigationContentLoader, 691
- información sobre un enlace, 117
- iniciadores de colecciones, 484
- iniciadores de objetos, 483
- inicio de sesión, 837
- InitialDirectory, 282
- Initialized, 201
- Inline, 291
- InlineUIContainer, 292
- INotifyCollectionChanged, 358
- INotifyPropertyChanged, 96, 321, 353, 694
- InputBinding, 158, 160
- InputGesture, 158, 160
- InputGestureText, 147, 157
- Insert, 245
- INSERT, 394
- insertar filas, 537
- insertar o borrar elementos en una vista, 358
- InsertCommand, 409
- interceptar la tecla pulsada, 86
- interfaces de usuario dinámicas, 42
- interfaces gráficas, 5
- interfaz, 3
 - ICommand, 148, 168
 - IEditableObject, 449
- into, 501
- introducir datos, 364
- Invoke, 306
- inyección de código SQL, 417
- inyectar código XAML, 39
- IQueryable, 489, 503
- IsActive, 43, 44
- IsCancel, 220
- IsCheckable, 177
- IsChecked, 177, 230
- IsCurrentUserInRole, 822
- IsDefault, 69, 220
- IsEditable, 249
- IsEnabled, 179, 203
- IsFocusScope, 152
- IsKeyDown, 88
- IsKeyToggled, 88

- IsKeyUp, 88
- isla de datos XML, 335
- IsLoggedIn, 819
- IsMouseDown, 82
- IsMouseOver, 82
- IsMuted, 670
- IsReadOnly, 249
- IsThreeState, 230
- IsVisible, 80
- it, 510
- Items, 208
 - propiedad, 242
- ItemsControl, 62
- ItemsSource, 242, 274, 325
- IValueConverter, 106

J

- join, 488, 500
- Journal, 579
- JournalOwnership, 687

K

- KeepAlive, 565, 579, 616
- Key, 87, 158
- KeyBinding, 157
- Keyboard, 80, 88
- KeyDown, 72, 76
- KeyGesture, 158, 160
- KeyUp, 76

L

- Label, 27, 62, 63, 639
- LargeChange, 253
- LayoutMode, 886
- Lazy Loading, 511
- lector de datos, 407
- Left, 45
- let, 502
- Line, 63
- LinearGradientBrush, 649
- líneas de ayuda, 886
- LineBreak, 226, 292
- LINQ, 481
- LINQ to Entities, 507
- List, 288
- lista, 240, 882
 - acceder a sus elementos, 244

- borrar elemento, 246
- de elementos CheckBox, 247
- de los eventos, 888
- desplegable, 249, 882
 - añadir elemento, 252
 - borrar elemento, 252
 - diseñar, 250
- diseño, 243
- elemento seleccionado, 242
- selección múltiple, 245
- ListBox, 27, 62, 240, 324
- ListBoxItem, 62, 240
- ListCollectionView, 324
- ListView, 264
- Load documento dinámico, 302
- LoadComponent, 12
- Loaded, 43, 79, 199, 201
- LoadingRow, 276
- LoadOperation, 850
- LoadSize, 861
- LocationChanged, 45
- LogicalTreeHelper, 42
- Login, 819
- Logout, 819
- LostFocus, 80
- LostKeyboardFocus, 80

M

- maestro-detalle, 343, 358
- mage.exe, 595
- Main, 14
- MainWindow, 42, 57
- manejador de eventos, 38
- marcas de tiempo, 762
- marco, 881
- marco de la ventana, 2
- Margin, 29, 70
- máscaras, 133
- Matrix, 658
- MatrixTransform, 661
- Max, 488
- MaxDropDownHeight, 249
- MaxHeight, 69
- maximizar, 2
- Maximum, 253
- MaxWidth, 69
- MediaElement, 669
- MediaEnded, 670
- MediaFailed, 670

- MediaOpened, 670
- MemberExpression, 493
- membership, 799, 814
- mensaje, mostrar, 146
- Menu, 62, 141
- menú, 140
 - añadir imagen a un elemento, 177
 - contextual, 178
 - controlador de un elemento, 144
 - de control, 2
 - de desbordamiento, 182
 - dinámico, 207
 - emergente, 178
 - señalar elemento, 204
- MenuBase, 62
- MenuItem, 62, 141
- MenuItem.Icon, 177
- menús, 139, 881
 - detalles, 177
 - diseño, 141
 - líneas de separación, 144
- MessageBox, 146, 214
- MessageBox.Show, 94
- MessageBoxResult, valor retornado..., 215
- metadatos, 578
- MethodCallExpression, 493
- método abreviado, 157
- método LoadComponent, 12
- método Main, 14
- método Run, 15
- método Shutdown, 15
- Métodos del generador de consultas, 507
- métodos extensores, 484
- Microsoft Access, 406
- Microsoft.ACE.OLEDB, 407
- Microsoft.Jet.OLEDB, 407
- Microsoft.Win32, 280
- Min, 488
- MinHeight, 69
- minimizar, 2
- Minimum, 253
- MinWidth, 69
- modal o no modal, 214
- Mode, 100, 694
- modelo asíncrono, 740
- modelo de datos, 347
- modelo de entidades, 506
- modelos de contenido, 64
- Model-View-ViewModel, 169
- modificadores de tamaño, 884

- modificar datos, 533
- modificar datos en una tabla, 394
- Modified, 403
- ModifierKeys, 158
- Modifiers, 88, 158
- mostrar datos, 781
- mostrar un mensaje, 146
- Mouse, 82
- MouseBinding, 160
- MouseDown, 82
- MouseEnter, 82
- MouseLeave, 82
- MouseLeftButtonDown, 677
- MouseLeftButtonUp, 677
- MouseMove, 82
- MouseUp, 82
- MoveCurrentToFirst, 707
- MoveCurrentToNext, 339, 355
- mover el control, 884
- MSL, 506
- MultiTrigger, 124
- Mutex, 53

N

- navegación
 - de fragmento, 582
 - de Silverlight, 681
 - de tipo Web, 560
 - estructurada, 583
 - externa, 690
 - falla, 581
 - historial, 583
 - personalizada de Silverlight, 680
- navegar por los elementos de una vista, 355
- Navigate, 564, 579, 684
- NavigateUri, 580, 685
- Navigating, 686, 826, 841
- Navigation, 684, 687
- NavigationFailed, 687
- NavigationService, 561, 565
- NavigationUIVisibility, 583
- NavigationWindow, 561
- nemónico, 69, 146
- N-Layer architecture, 761, 793
- NotifyOnValidationError, 111
- NT AUTHORITY\Servicio de Red, 763
- N-Tier architecture, 761, 793
- Nullable<bool>, 222

O

- object, 531
- Object, 60
- ObjectCollection, 245, 251
- ObjectContext, 513
- ObjectDataProvider, 362
- ObjectQuery, 507
- ObjectSet< >, 520
- ObjectStateEntry, 534, 548
- ObjectStateManager, 534, 548
- objeto aplicación, referencia, 57
- objetos como orígenes de datos, 347
- objetos de negocio, 435
- ObservableCollection, eventos, 451
- ObservableCollection, InsertItem, 451
- OdbcConnection, 405
- OdbcDataAdapter, 409
- OleDbConnection, 405
- OleDbDataAdapter, 409
- OleDbDataReader, 408
- OnReturn, 588
- OnStartup, 54
- OpenFile, 282
- OpenFileDialog, 281, 717
- OpenRead, 717
- operadores de consulta, 488
- OperationContract, 730
- OptimisticConcurrencyException, 543
- OracleConnection, 405
- OracleDataAdapter, 409
- orden, 409
- orden enrutada, 147
 - información adicional, 161
 - parámetros, 165
- orden parametrizada, 421
- orden Tab, 63
- ordenación de los datos, 860
- ordenar una vista de elementos, 356
- órdenes, 147
 - enrutadas comunes, 148
- orderby, 488, 496
- Orientation, 257
- origen de datos implícito, 103, 105
- OriginalSource, 77
- ORM, 505
- out, 54
- OwnedWindows, 45, 228
- Owner, 45, 228

P

- Padding, 71
- Page, 565
 - duración, 578
- PagedCollectionView, 702, 781
- PageFunction, 583
- PageSize, 861
- página de Silverlight, 683
- paginación, 703, 860
- páginas, pasar datos entre, 577
- Panel, 30, 62
- paneles de diseño, 30
- pantalla de presentación, 55
- Paragraph, 288, 293
- Parameter, órdenes enrutadas, 165
- ParameterExpression, 493
- Parameters, 421
- parámetros de consulta, 860
- parámetros en órdenes SQL, 421
- partial, 11
- pasos, 316
- Password, 219
- PasswordBox, 61, 219, 639
- PasswordChar, 219
- Paste, 191
- Path, 100, 103, 320
- patrones de diseño, 511
- perfiles, 827, 840
- plantilla, 125
- plantillas de datos, 324
- pool de conexiones, 413
- Popup, 650
- portapapeles, 189
- posición del ratón, 82
- posición inicial de la ventana, 43
- Position, 671
- PostgreSql, 406
- predicado, 709
- PresentationHost, 559
- PreviewGotKeyboardFocus, 80
- PreviewKeyDown, 72, 76, 86
- PreviewKeyUp, 76
- PreviewLostKeyboardFocus, 80
- PreviewMouseDown, 82
- PreviewMouseMove, 82
- PreviewMouseUp, 82
- PreviewTextInput, 76
- PRIMARY KEY, 393
- Print, 303
- PrintDialog, 280, 303
- PrintDocument, 285
- PrintVisual, 303
- procedimiento almacenado, 422, 434
- producto cartesiano, 500
- ProfileService, 809, 813, 827
- ProgressBar, 62, 258
- Property, 26
- PropertyChanged, 96, 100, 320
- propiedad
 - asociada, 24
 - auto-implementada, 483
 - cambió, notificar, 96
 - como atributo, 20
 - como elemento, 21
 - Content, 28
 - de contenido, 21
 - de dependencia, 25, 167
 - Font..., 28
 - HorizontalAlignment, 29
 - Icon, 887
 - Items, 208
 - Margin, 29
 - Mode, 100
 - PropertyChanged, 100, 320
 - Resources, 129
 - StartupUri, 15
 - ToolTip, 30
 - UpdateSourceTrigger, 100
 - Visibility, 210
 - xmlns, 9
- propiedades
 - asociadas, 111
 - de navegación, 520
 - de un objeto, 884
 - del proyecto, 892
 - HorizontalAlignment y VerticalAlignment, 29
- proveedor de datos, 404
- proveedor de datos de objetos, 362
- proveedor de entidades, 508
- proveedor XmlDataProvider, 331
- proveedores de LINQ, 503
- publicar un servicio WCF, 787
- publicar una aplicación Silverlight, 747
- publicar XBAP, 592
- puntero, 880
- punto de inserción, 79
- punto final, 729, 813

Q

QueryParameters, 860

R

RadialGradientBrush, 649

RadioButton, 27, 62, 234

RangeBase, 62

Read, 408

Rectangle, 63

recursos, 23, 129, 845

creados mediante código, 131

de una aplicación, 48

del sistema, 131

Redo, 191

referencias, 877

reflexión, 50, 531, 632

Refresh, 386, 546

registrar usuario, 838

registros, 391

reglas de validación, 110

rehacer, 191

rejilla de ayuda, 886

RelativeSource, 24, 121, 461

reloj, 305

despertador, 312, 314

digital, 307

Remove, 245

RemoveAt, 245

colección, 211

RemoveBackEntry, 564

RemovedItems, 446

RequiresAuthenticationAttribute, 839

RequiresRoleAttribute, 839

SizeMode, 45, 219

ResourceDictionary, 129

ResourceManager, 48

Resources, 129

RestoreBounds, 46

ReturnEventArgs, 588

RIA Services, 832, 843

RichTextBox, 61, 199, 285, 300

roles, 839

RoleService, 809, 813, 822

RootVisual, 625

RotateTransform, 660

RoutedCommand, 148

RoutedEvent, 71

RoutedEventArgs, 72

RoutedEventHandler, 38, 74

RoutedUICommand, 150

Row, 34, 447, 455

RowChanged, 476

RowDeleted, 476

RowEditEnding, 447

RowFilter, 478

RowSpan, 34

RowState, 474

RowValidationRules, 460

Run, 4, 15, 226, 292, 293

S

Save documento dinámico, 301

SaveChanges, 520

SaveFileDialog, 282

ScaleTransform, 660

Scroll, 256

ScrollBar, 27, 62, 253

ScrollIntoView, 323, 339, 356

ScrollViewer, 36, 62, 260

Section, 289

SecurePassword, 219

SecureString, 219

seguimiento de los cambios, 547

seguridad en XBAP, 595

seleccionar datos de una tabla, 395

seleccionar el contenido de una caja de texto,
80

seleccionar un objeto, 885

select, 488, 496

Select, 82, 191

SELECT, 395

SelectAll, 81, 191

SelectCommand, 409

SelectedIndex, 242

SelectedIndexChanged, 529

SelectedItem, 242, 251

SelectedItems, 245

SelectedText, 82, 190

SelectedValue, 426

SelectedValuePath, 426

SelectionChanged, 273, 446

SelectionLength, 81, 191

SelectionStart, 81, 190

Selector, 62

sentencia using, 416

señalar un elemento de un menú, 204

separadores, 144

- Separator, 61, 141
- seriación/deseriación, 12, 13
- ServiceContract, 730
- servicio, 727
 - de autenticación, 819
 - de conexiones, 413
 - de dominio, 834, 847, 849
 - de navegación, 565
 - de red, 764
 - de suscripciones, 814
 - WCF, 728
 - de funciones, 822
 - habilitado para Silverlight, 751
 - para acceso a datos, 765
- servicios de aplicación, 796, 808
 - configurar, 812
- Servicios de objetos, 508
- servicios Web y LINQ, 760
- SessionEnding, 50
- SetResourceReference, 132
- Setter, 122, 330
- Shape, 63
- Show, 44, 214, 217, 228
- ShowActivated, 46
- ShowDialog, 43, 217, 228, 281, 717
- ShowGridLines, 34
- ShowInTaskbar, 43, 46, 219
- ShowsNavigationUI, 564
- Shutdown, 15
- ShutdownMode, 54
- SignOut, 805
- Silverlight, 619, 724
 - <object>, 627
 - acceso a datos, 692
 - administrador de identificadores de recursos, 687
 - aplicación fuera del explorador, 691
 - arquitectura de una aplicación, 623
 - arquitectura, 620
 - autenticación y autorización, 691
 - compilación de la aplicación, 627
 - comunicación entre dominios, 745
 - controles, 629
 - crear aplicación, 622
 - diccionario de recursos, 682
 - diseño de una página, 629
 - Frame, 681
 - llamadas asíncronas, 741
 - navegación, 691
 - Page, 681
 - página de entrada, 627
 - SDK, 638, 644
 - vista de una colección, 702
- simultaneidad, 543, 761
- simultaneidad optimista, 844
- sincronizar con CurrentItem, 337
- sistema de diseño, 30
- sistema de navegación Silverlight, extender, 691
- sitio Web, 592
 - administración, 816
- SizeToContent, 46
- SkewTransform, 661
- Slider, 62, 257
- SmallChange, 253
- SOAP, mensaje, 728
- SolidColorBrush, 664
- soluciones y proyectos, 893
- Sort, 478
- SortDescriptors, 860
- Source, 77, 101, 320, 564
- Span, 293
- SplashScreen, 55
- SQL, 392
- SQL Server, 406, 414, 763, 894
- SQL Server Management Studio Express, 897
- SQLCMD, 895
- SqlConnection, 405
- SqlDataAdapter, 409
- SqlDataReader, 408
- SSDL, 506
- StackPanel, 31, 63, 180
- Startup, 50, 53
- StartupUri, 15, 52
- StateChanged, 46
- StaticResource, 23, 130
- StatusBar, 62, 184
- StatusBarItem, 62
- Storyboard, 330, 663, 665
- Stretch, 670
- Style, 60, 121, 273, 330
- subprocesos, 306
- Sum, 489
- suscribirse a un evento, 37, 74
- System.Data.Linq, 481
- System.Linq, 481
- System.Linq.Expressions, 492
- System.Windows, 29
- System.Windows.Markup, 12
- SystemColors, 131

SystemFonts, 131
SystemParameters, 131

T

Tab, 63
TabControl, 261
TabIndex, 63
tabla, 391, 882
Table, 289
TakeWhile, 489
tamaño de los controles, 884
tamaño de una fuente, 199
tamaño de una ventana, 66
TargetNullValue, 455
TargetType, 121
tecla de acceso, 29, 63, 69
tecla Entrar, 64
tecla pulsada, interceptar, 86
temas, 133
TemplateBinding, 24
temporizador, 306
 resolución, 309
Text, 77, 79, 249
TextBlock, 27, 62, 226, 632
TextBox, 27, 61, 64, 190, 638
TextBoxBase, 61, 190
TextChanged, 78, 95, 237
TextInput, 76
texto seleccionado, 202
TextProperty, 101
TextRange, 301
TextWrapping, 187
Tick, evento, 307
TickFrequency, 257
TickPlacement, 257
ticks, 316
Timer, 306
TimeSpan, 310, 676
tip, 30
tipo anónimo, 482, 531
tipo implícito, 482
tipos SQL, 393
ToByteArray, 714
ToDouble, 79
ToggleButton, 62
ToInt32, 238
ToList, 521
ToolBar, 62, 180
ToolBarTray, 183

Toolkit.dll, 862
ToolTip, 30, 62
Top, 45
Topmost, 46
ToString, 239
ToTraceString, 544
ToUpper, 239
transacciones, 423
 explícitas, 428
TransactionScope, 424
transformaciones, 658
TransformGroup, 661
TranslateTransform, 659
TreeView, 266
TreeViewItem, 266
Trigger, 123
Triggers, 329
TryFindResource, 131
T-SQL, 544

U

UIElement, 60, 100
Unchanged, 403
Unchecked, 230
Undo, 191, 205
UNIQUE, 393
Unloaded, 201
Update, 410, 476
UPDATE, 394
UpdateCommand, 409
UpdateSource, 117
UpdateSourceTrigger, 100, 104
URI, 582, 688
UriMapper, 687
UriMapping, 688
User, 839
UserControl, 62, 605, 625
using, 416
 directrices, 16

V

validación, 370
 de los datos, 109, 640
 personalizada, 118, 375
 proceso, 114, 376
 visualizar errores, 372
validar un campo de texto, 89
validar un grupo de enlaces, 379

Validate, 118, 375
 ValidateUser, 819
 Validation, 111, 374
 Validation.Error, 112
 Validation.GetHasError, 112
 ValidationAttribute, 635
 ValidationRule, 110, 118, 370
 ValidationSummary, 639
 Value, 253, 259
 Value Object, 432, 511
 ValueChanged, 253
 var, 482
 varbinary, 763
 variable, declaración implícita, 482
 ventana, 1
 activa, 43
 modal, 43
 no modal, 42
 posición, 43
 ventanas, 3
 colección, 42
 VerticalAlignment, 70
 VerticalContentAlignment, 29, 70
 VerticalScrollBarVisibility, 187
 vídeo, 669, 678
 View, 323
 Viewbox, 36
 vinculación de datos, 91
 virtualizar, 363
 VirtualizingStackPanel, 363
 Visibility, 71, 210
 vista de colección, 322, 351, 337
 vista de un DataTable, 477
 vista PagedCollectionView, 781
 vista predeterminada, 354
 vista
 agrupar sus elementos, 341
 elemento actual, 337
 filtrar sus elementos, 340, 607
 navegar por sus elementos, 338
 obtener, 337
 ordenar sus elementos, 339
 Visual, 60
 Volume, 670

W

WCF, 727
 WCF RIA Services, 832, 843

WebBrowser, 617
 webcam, 680
 WebContext, 836, 839
 where, 488, 496
 Width, 69
 Window, 9, 62
 Windows, 42
 Windows Forms, 5
 WindowStartupLocation, 43, 46
 WindowState, 46, 57
 WindowStyle, 47
 WindowTitle, 565
 WPF, 5, 59
 WPF, XBAP y Silverlight, 559
 WrapPanel, 32, 63
 WriteableBitmap, 713
 WriteLine, 146

X

x:Array, 24
 x:Name, 606
 x:Null, 23
 x:Static, 24, 132
 x:Type, 24
 x:XData, 335
 XAML, 8, 10
 extensiones de marcado, 22
 información general, 18
 XamlReader, 12, 41
 XamlWriter, 12
 XBAP, 6, 561, 590
 publicar, 592
 seguridad, 595
 y bases de datos, 597
 XCOPY, 895
 XML, simular BBDD, 904
 XmlDataProvider, 331
 XmlElement, 338
 xmlns, 9, 20
 XPath, 320, 334
 XPath=@..., 271, 334
 XSD, 904

Y

yield return, 498

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** ISBN: 978-84-86381-43-1
Guía del programador 444 págs.
 - Manual para **Quick C 2** ISBN: 978-84-86381-65-3
Guía del programador 540 págs.
 - Manual para **Quick BASIC 4.5** ISBN: 978-84-86381-74-5
Guía del programador 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) ISBN: 978-84-7897-180-0
Aplicaciones para Windows 846 págs. + 2 disquetes
 - Microsoft **Visual C ++** ISBN: 978-84-7897-561-7
Aplicaciones para Win32 (2ª edición) 792 págs. + disquete
 - Microsoft **Visual C ++** ISBN: 978-84-7897-344-6
Programación avanzada en Win32 888 págs. + CD-ROM
 - **Visual Basic 6** ISBN: 978-84-7897-357-6
Curso de programación (2ª edición) 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** ISBN: 978-84-7897-745-1
Lenguaje y aplicaciones 392 págs. + CD-ROM
 - Programación orientada a objetos
con C ++ (4ª edición) ISBN: 978-84-7897-761-1
648 págs. + CD-ROM
 - **C/C++** ISBN: 978-84-7897-762-8
Curso de programación (3ª edición) 708 págs. + CD-ROM
 - **Microsoft C#** ISBN: 978-84-7897-813-7
Lenguaje y aplicaciones (2ª edición) 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** ISBN: 978-84-9964-020-4
Lenguaje y aplicaciones (3ª edición) 520 págs. + CD-ROM
 - **Java 2** ISBN: 978-84-9964-032-7
Curso de programación (4ª edición) 820 págs. + CD-ROM
 - **Microsoft C#** ISBN: 978-84-9964-068-6
Curso de programación (2ª edición) 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 instálos.

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

Todo el contenido de este CD, excepto los ejemplos del libro, es propiedad de las firmas que los representan (Microsoft, etc.). La inclusión en este libro se debe a su gentileza y es totalmente gratuita y con la finalidad de apoyar el aprendizaje del software correspondiente. Para obtener más información y actualizaciones visite las direcciones indicadas en dicho software:

<http://www.microsoft.com/es/es/default.aspx>

Al realizar el proceso de instalación, haga el favor de consultar el acuerdo de licencia para cada uno de los productos.

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.