[2008]

Visual Basic 2008

By Everts Garay Gaitan

Si puede imaginar un programa informático, probablemente puede crearlo con Microsoft Visual Basic 2008 Express. Desde un programa sencillo que muestre un mensaje hasta una aplicación completa con acceso a una base de datos o a un servicio Web, Visual Basic proporciona las herramientas que necesita.Más que una simple herramienta de aprendizaje, Visual Basic proporciona un entorno de desarrollo totalmente funcional para programadores principiantes y aficionados que están interesados en generar aplicaciones de Windows Forms, aplicaciones de consola y bibliotecas de clases.

Granada, Nicaragua
evertsfnic@hotmail.com
[www.vacationinnicaragua.com]

¿Qué es Visual Basic Express?

Es tanto una herramienta para aprender a programar en Visual Basic, como una herramienta de desarrollo funcional para programadores que no necesitan la versión completa de Visual Basic. Pero Visual Basic Express es más que un subconjunto de Visual Basic: incluye muchas características que simplifican más que nunca la programación en Visual Basic.

La manera mejor de obtener información sobre lo que puede hacer con Visual Basic Express es examinar las lecciones incluidas en Paseo con guía por Visual Basic. Cuando termine, estará familiarizado con las herramientas y los conceptos de Visual Basic, y preparado para empezar a escribir sus propios programas.

¿A quién va dirigido Visual Basic Express?

Visual Basic Express es una herramienta eficaz capaz de crear aplicaciones y componentes plenamente funcionales que se pueden compartir con otros usuarios. No va dirigida, sin embargo, a desarrolladores profesionales ni a programadores que trabajan en un entorno de equipo. Otras versiones de Visual Basic proporcionan características que satisfacen las necesidades avanzadas de programación profesional y en equipo.

Si tiene que escribir aplicaciones que conecten con una base de datos en red, interactúen con Microsoft Office, sean compatibles con dispositivos móviles o sistemas operativos de 64 bits o requieran depuración remota, necesitará una versión más avanzada de Visual Basic.

En este tema se proporciona información general de Visual Basic, un programa para crear aplicaciones mediante el lenguaje Visual Basic. Igual que un programa como Microsoft Outlook proporciona diversas herramientas para trabajar con correo electrónico, Visual Basic Express es un kit de herramientas con el que realizar una amplia gama de tareas de programación.

Sugerencia

Si es principiante en programación, quizá desee finalizar el paseo guiado por Visual Basic, un conjunto de lecciones diseñado para enseñar los conceptos básicos, y después volver a este tema. Para iniciar el paseo, vea Crear el primer programa en Visual Basic.

El proceso de desarrollo

Visual Basic Express facilita el proceso de desarrollar aplicaciones; en la mayoría de los casos, el proceso consta de los pasos siguientes:

Cree un proyecto. Un proyecto contiene todos los archivos necesarios para la aplicación y almacena información sobre la aplicación. A veces, una aplicación contendrá más de un proyecto, por ejemplo, un proyecto de **aplicación para Windows** y uno o varios proyectos de **biblioteca de clases**. Tal aplicación se denomina *solución*, que es sólo otro nombre para un grupo de proyectos.

Diseñe la interfaz de usuario. Para ello, puede arrastrar distintos controles, como botones y cuadros de texto, a una superficie de diseño conocida como *formulario*. Puede establecer propiedades que definan el aspecto y comportamiento del formulario y de sus controles.

> **Nota**
>
> Para aplicaciones que no tienen ninguna interfaz de usuario, como bibliotecas de clases o aplicaciones de consola, este paso no es necesario.

Escriba el código. A continuación, tendrá que escribir el código de Visual Basic Expressque define cómo se comportará la aplicación y cómo interactuará con el usuario. Visual Basic Express facilita la escritura de código con características como Intellisense, finalización automática y miniprogramas.

Pruebe el código. Siempre deseará probar la aplicación para asegurarse de que se comporta del modo que esperaba; este proceso se conoce como *depuración.* Visual Basic Express dispone de herramientas de depuración que facilitan la búsqueda y corrección de errores en el código de forma interactiva.

Distribuya la aplicación. Una vez que la aplicación está completa, puede instalar el programa final en el equipo o distribuirlo y compartirlo con otros usuarios. Visual Basic Express utiliza una nueva tecnología conocida como publicación de ClickOnce, que permite implementar fácilmente una aplicación con un asistente y proporcionar versiones actualizadas de la aplicación si más adelante realiza cambios.

Familiarizarse

A primera vista, la interfaz de usuario de Visual Basic Express, también conocida como *entorno de desarrollo integrado* o *IDE*, puede resultar extraña, pero una vez que se familiarice, la encontrará fácil de utilizar. En las secciones siguientes se describen las partes del IDE que más utilizará.

Al iniciar

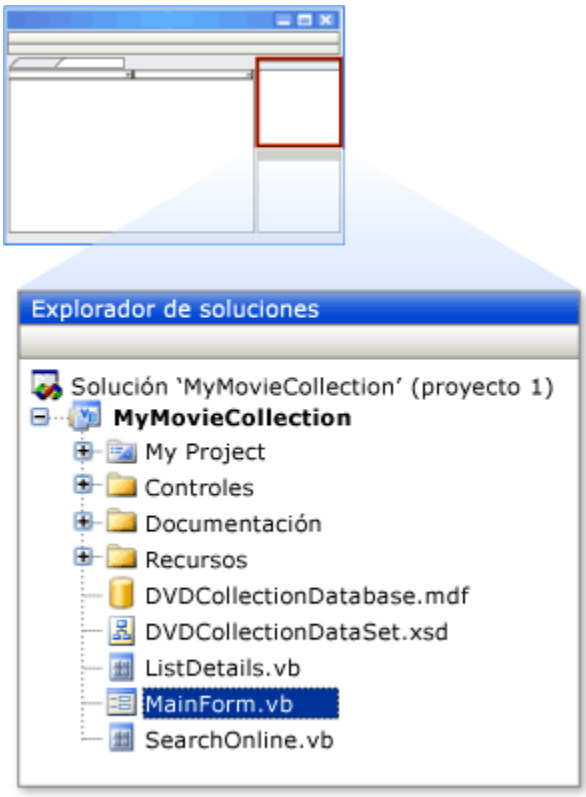
Cuando abre por primera vez Visual Basic Express, verá que la ventana **Página de inicio** ocupa la mayor parte. La **Página de inicio** contiene una lista de los proyectos recientes en los que se puede hacer clic, un área **Introducción** con

vínculos a temas de Ayuda importantes y una lista de vínculos a artículos en línea y otros recursos. Si se conecta a Internet, esta lista se actualizará regularmente.

Puede cambiar lo que aparece en la **Página de inicio** para que se ajuste a sus preferencias personales. Para obtener más información, vea Cómo: Personalizar la sección de noticias de la Página de inicio.

En el lado derecho del IDE, se muestra la ventana **Explorador de soluciones**. Inicialmente está en blanco, pero aquí es donde se mostrará la información sobre su proyecto o grupos de proyectos conocidos como soluciones. Para obtener más información, vea Utilizar el Explorador de soluciones.

Figura 1: Explorador de soluciones



En el lado izquierdo del IDE, se muestra una ficha vertical marcada **Cuadro de herramientas**. También está en blanco inicialmente, pero a medida que trabaja se rellenará con elementos que se pueden utilizar para la tarea en la que está trabajando. Para obtener más información, vea Usar el Cuadro de herramientas.

En la parte superior del IDE hay una barra de menús y una barra de herramientas. Los menús y los botones de la barra de herramientas cambian

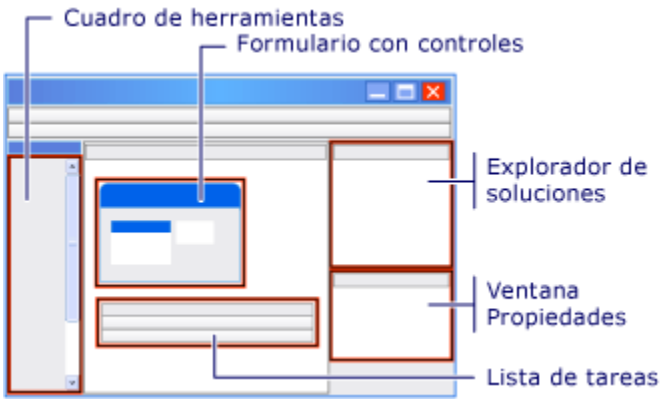
según la tarea del momento, tómese algo de tiempo para explorar y ver qué opciones están disponibles. También puede personalizar los menús y la barra de herramientas para que se ajusten a sus preferencias personales. Para obtener más información, vea Cómo: Personalizar las barras de herramientas (Visual Studio).

En la parte más inferior del IDE hay una barra de estado que muestra **Listo**. Cuando trabaja en el IDE, la barra de estado cambia y muestra mensajes relacionados con la tarea en curso, por ejemplo, la barra de estado muestra información sobre el progreso de un proyecto que está generando.

Modo de diseño

Cuando abre o crea un proyecto, el aspecto del IDE cambia al *modo de diseño*. Ésta es la parte visual de Visual Basic, donde se diseña el aspecto de la aplicación.

Figura 2: IDE en el modo de diseño



En el modo de diseño, la **Página de inicio** se cubre con otra ventana conocida como **Diseñador de Windows Forms**, que es básicamente un lienzo en blanco que representa la interfaz de usuario de la aplicación. Observe que la **Página de inicio** todavía está disponible haciendo clic en la ficha correspondiente en el **Diseñador de Windows Forms**.

Cuando está visible el **Diseñador de Windows Forms**, el **cuadro de herramientas** contiene varios controles (representaciones de botones, campos de texto, cuadrículas, etc.) que se pueden colocar en el formulario y organizar como se desee. Para obtener más información, vea Diseñador de Windows Forms.

También observará que aparece una nueva ventana, la ventana **Propiedades**, bajo la ventana **Explorador de soluciones**. Aquí es donde establecerá las

distintas propiedades que definen el aspecto y comportamiento del formulario y sus controles. Para obtener más información, vea Propiedades (Ventana).

De forma predeterminada, no se muestra la ventana Lista de tareas en la parte inferior del IDE, pero proporciona un lugar donde puede llevar un seguimiento de las tareas que es necesario realizar o anotar cuando programa. Para obtener más información, vea Lista de tareas (Visual Studio).

Si hace doble clic en un formulario o control, se abre una nueva ventana llamada Editor de código. Aquí es donde escribe el código real para la aplicación. El Editor de código es algo más que un simple editor de texto, ya que utiliza una tecnología conocida como IntelliSense que facilita la escritura del código al proporcionar información según escribe. Para obtener más información, vea Opciones de IntelliSense específicas de Visual Basic.

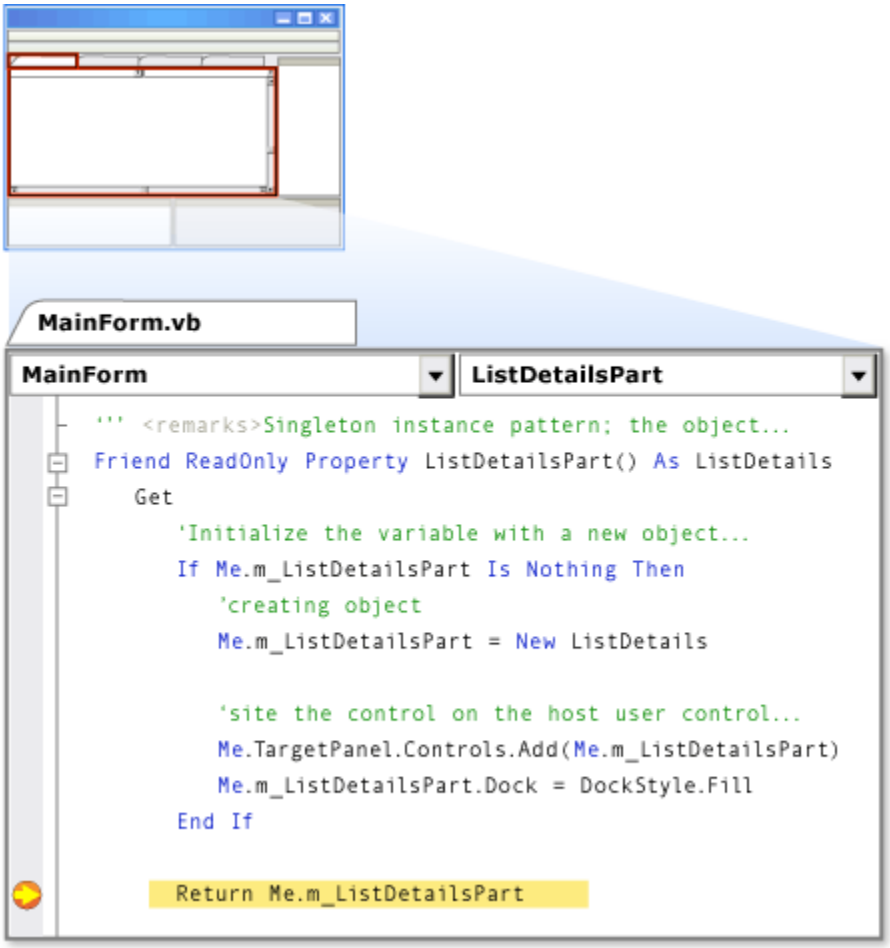
Nota

Para algunos tipos de proyectos, como los proyectos de **bibliotecas de clase** que no disponen de interfaz de usuario, se muestra el Editor de código en lugar del **Diseñador de Windows Forms**.

Modo de ejecución

Cuando ejecuta o depura la aplicación, el IDE cambia a *modo de ejecución*. Se inicia la aplicación y aparece una ventana adicional relacionada con la depuración. Cuando está en el modo de ejecución, no puede hacer cambios en el **Diseñador de Windows Forms**, la ventana **Propiedades** ni en el **Explorador de soluciones**, pero puede modificar el código en el Editor de código.

Figura 3: Formulario de Visual Basic Expressen el modo de interrupción



En el modo de ejecución, aparece una nueva ventana conocida como la ventana **Inmediato** en la parte inferior del IDE. Si coloca la aplicación en el modo de interrupción, puede consultar valores y probar el código en la ventana **Inmediato**. Para obtener más información, vea Inmediato (Ventana).

Durante la ejecución se pueden mostrar ventanas adicionales y observar los valores de variables, mostrar los resultados y otras tareas de depuración seleccionándolas en el menú **Depurar**.

Otras ventanas importantes

Hay numerosas ventanas adicionales en el IDE, cada una para una tarea de programación concreta. Algunas de las más comunes se muestran a continuación.

La ventana **Lista de errores** aparece en la parte inferior del IDE si se escribe código incorrecto o aparecen otros errores en tiempo de diseño. Para obtener más información, vea Lista de errores (Ventana).

La ventana del **Examinador de objetos** se utiliza para examinar las propiedades, métodos y eventos de los objetos que se pueden utilizar en la aplicación. Para obtener más información, vea Examinador de objetos.

El **Diseñador de proyectos** se utiliza para configurar las propiedades de la aplicación, incluidos los recursos, el comportamiento de depuración, la configuración de implementación y mucho más. Para obtener más información, vea Introducción al Diseñador de proyectos.

El **Explorador de base de datos** permite visualizar y utilizar bases de datos existentes o crear y diseñar otras nuevas. Para obtener más información, vea Explorador de servidores/Explorador de bases de datos.

Personalización

Visual Basic Express permite personalizar el IDE cambiando el diseño de ventana, seleccionando qué ventanas se van a mostrar, agregando o eliminando comandos de menú y botones de la barra de herramientas, etcétera. Para obtener más información, vea Personalizar el entorno de desarrollo.

Crear el primer programa en Visual Basic

La mejor manera de aprender a programar con Visual Basic Expresses crear realmente un programa. Los ejercicios siguientes le guían por el proceso de creación de un programa para ver páginas Web.

Si no entiende todo enseguida, no se preocupe; los conceptos presentados aquí se tratarán con más detalle en otras secciones del Paseo con guía por Visual Basic.

El primer paso para crear un programa de Visual Basic Expresses abrir Visual Studio y crear un *proyecto*. Esto se hará al crear cualquier programa de Visual Basic.

-Para crear un proyecto para el programa

En el menú **Inicio** de Windows, elija Microsoft Visual Basic Express.

Aparecerá la pantalla de bienvenida a Visual Basic Express. Ésta es la interfaz para Visual Basic Express, también conocida como *entorno de desarrollo integrado* o *IDE*.

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

Aparece el cuadro de diálogo **Nuevo proyecto**.

Seleccione **Aplicación para Windows** y haga clic en **Aceptar**.

Se muestra un nuevo *formulario* en el IDE y se agregan los archivos necesarios para el proyecto a la ventana **Explorador de soluciones**. Si éste es el primer proyecto de **Aplicación para Windows** que ha creado, se denomina "WindowsApplication1".

Información detallada

Acaba de crear un proyecto para el programa de exploración Web. Un proyecto en Visual Basic Expresses un lugar para almacenar partes del programa y mantenerlas organizadas.

Cuando crea un nuevo proyecto por primera vez, sólo existe en la memoria. Si cierra el entorno de desarrollo integrado (IDE) de Visual Basic, se le pide que guarde o descarte el proyecto. Cuando lo guarde, puede darle un nombre más significativo.

Al abrir el cuadro de diálogo **Nuevo proyecto**, había diversos tipos de proyectos entre los cuales elegir. El programa de exploración Web es una *Aplicación para Windows* normal; es decir, un programa que se puede ejecutar desde el menú **Inicio**.

Al crear el proyecto, aparecía un formulario (también conocido como *diseñador* de formularios) en el entorno de desarrollo integrado (IDE). Este formulario representa una ventana que se mostrará cuando se ejecuta el programa. Muchos programas muestran más de una ventana, por lo que un proyecto puede contener múltiples formularios.

Paso 2: Crear una interfaz de usuario

Es el momento de comenzar a crear un explorador Web. Se utilizará Microsoft Visual Basic Express para generar la *interfaz de usuario* (la parte visible con la cual interactúan los usuarios) agregando *controles* del **Cuadro de herramientas** al formulario.

El **Cuadro de herramientas** se encuentra en el lado izquierdo de Visual Studio y consta de varias fichas, como **Datos**, **Componentes** y **Todos los formularios Windows Forms**. Dentro de cada ficha hay un conjunto de entradas, que representan controles o componentes que se pueden agregar a la aplicación. Por ejemplo, la ficha **Todos los formularios Windows Forms** tiene entradas denominadas **Textbox**, **Button** y **Checkbox** que representan los controles que puede agregar a la aplicación arrastrándolos al formulario.

Para agregar controles a la aplicación

Haga clic en el panel **Cuadro de herramientas**.

Se abrirá el **Cuadro de herramientas**.

> Sugerencia
>
> El **Cuadro de herramientas** es más fácil de utilizar si mantiene la ventana abierta. Puede hacer esto haciendo clic en el icono **Ocultar automáticamente**, que parece una chincheta.

Haga clic en la ficha **Todos los formularios Windows Forms** del **Cuadro de herramientas**, seleccione el control **Panel** y arrastre un panel a la esquina superior izquierda del formulario.

> Sugerencia
>
> Si tiene dificultades para encontrar el control correcto, haga clic con el botón secundario del mouse (ratón) en el Cuadro de herramientas y seleccione Ordenar elementos alfabéticamente.

En la misma ficha, arrastre un control **Button** y colóquelo en la parte superior del control **Panel**.

> Sugerencia
>
> Puede cambiar la posición de los controles mediante una operación de arrastrar y colocar. También puede cambiar el tamaño de los controles haciendo clic y arrastrando el borde o la esquina del control.

Desde la misma ficha, arrastre un control **TextBox** y colóquelo en la parte superior del control **Panel**.

Finalmente, en la ficha **Todos los formularios Windows Forms**, seleccione un control **WebBrowser** y colóquelo debajo del control **Panel**.

> Sugerencia
>
> Si tiene abierta la ventana **Cuadro de herramientas**, es posible que desee cerrarla ahora para tener más espacio de trabajo. Puede hacerlo haciendo clic una vez más en el icono **Ocultar automáticamente**.

Información detallada

Acaba de agregar cuatro controles al formulario. Los controles contienen código que define el aspecto que tendrán y las tareas que pueden realizar.

Por ejemplo, considere el control **Button**: casi todos los programas tienen un botón "Aceptar" o un botón "Salir". Si bien podría escribir su propio código para dibujar un botón en la pantalla, cambiar su apariencia cuando se presione y

realizar alguna tarea cuando se haga clic en él, hacerlo para cada programa enseguida se convertirá en un asunto tedioso. El control **Button** ya contiene el código necesario para hacer estas tareas, lo que le ahorra mucho trabajo innecesario.

Como puede ver, el **Cuadro de herramientas** contiene numerosos controles y cada uno de ellos tiene un propósito único. Los controles **Panel** se pueden utilizar para contener otros controles, como los que acaba de agregar. Los controles **Button** se utilizan generalmente para realizar tareas cuando el usuario hace clic en ellos; por ejemplo, cerrar el programa. Los controles **TextBox** se utilizan para escribir texto en una pantalla a través del teclado. Un control **WebBrowser** proporciona funciones de exploración Web integradas similares a Internet Explorer: seguro que no desea escribir todo el código de esa función.

En próximas lecciones aprenderá a personalizar la apariencia de éstos y de otros muchos controles, así como también a escribir el código que define su comportamiento. Además de utilizar los controles del **Cuadro de herramientas**, también puede crear sus propios controles, los que se conocen como *controles de usuario*; también se hablará de ellos en una próxima lección.

Paso 3: personalizar aspecto y comportamiento

En la lección anterior, se creó una interfaz de usuario agregando controles a la aplicación. En este punto, sin embargo, ésta no parece ni funciona como una aplicación finalizada. En esta lección, establecerá las *propiedades* para controlar la apariencia de los controles, utilizando la ventana **Propiedades**.

Para establecer las propiedades de los controles

En el **Diseñador de Windows Forms**, seleccione el control Panel.

La ventana **Propiedades** ubicada en la esquina inferior derecha del IDE muestra todas las propiedades para el control **Panel** denominado **Panel1**.

En la ventana **Propiedades**, seleccione la propiedad Dock y, a continuación, haga clic en la flecha a la derecha. Se mostrará una ventana pequeña de selección de propiedades con varios cuadros.

> **Sugerencia**
>
> La propiedad **Dock** se encuentra bajo la categoría **Diseño**. Puede ordenar las propiedades alfabéticamente haciendo clic en el botón **AZ** de la ventana **Propiedades**.

Haga clic en el cuadro superior en la ventana de selección de propiedades para establecer la propiedad **Dock** en **Top**. El control **Panel** se expandirá para rellenar la parte superior del formulario.

En el **Diseñador de Windows Forms**, seleccione el control WebBrowser. En la ventana **Propiedades**, para establecer la propiedad **Dock** en **Fill** seleccione la propiedad **Dock**, haga clic en la flecha a la derecha y seleccione el cuadro del centro de la ventana de selección de propiedades.

En el **Diseñador de Windows Forms**, seleccione el control Button.

En la ventana **Propiedades**, seleccione la propiedad Text del control **Button**. En la columna de la derecha, elimine `Button1` y reemplácelo por `Go!`.

Cambie el tamaño o vuelva a ubicar cualquiera de los controles y cambie nuevamente el tamaño del formulario según sus preferencias.

> **Nota**
>
> Los controles TextBox y **Button** deben permanecer en la parte superior de **Panel** o no podrá verlos cuando se ejecuta la aplicación.

Información detallada

En esta lección, se establecen varias *propiedades* que cambian la apariencia de los controles para la aplicación. Una propiedad en Visual Basic Expressrepresenta un atributo de un objeto, en este caso, un control. Por ejemplo, un atributo de un control **Button** es el texto que muestra. En este caso, se establece la propiedad **Text** para mostrar "Ir". Para obtener más información sobre las propiedades, vea Información detallada: comprender propiedades, métodos y eventos.

Las propiedades pueden tomar muchos tipos diferentes de valores además del texto. Por ejemplo, la propiedad **Dock** utilizó una ventana de selección de propiedades para mostrar las opciones disponibles. Otros valores de la propiedad pueden ser números, una opción que se selecciona de una lista, o una opción true o false.

Si se cambió el tamaño de un control o se reubicó, también se establecieron las propiedades. Las propiedades Size y Location determinan el tamaño y la ubicación del control en el formulario. Para ver esto en acción, seleccione la propiedad **Size** en la ventana **Propiedades** y utilice el mouse para cambiar el tamaño del control. Cuando se suelta el botón del mouse, los nuevos valores **Size** se mostrarán en la ventana **Propiedades**.

Además de establecer las propiedades en la ventana **Propiedades**, la mayoría de éstas se pueden establecer escribiendo un código. En una próxima lección,

aprenderá más acerca de cómo escribir el código para establecer las propiedades.

Información detallada: comprender propiedades, métodos y eventos

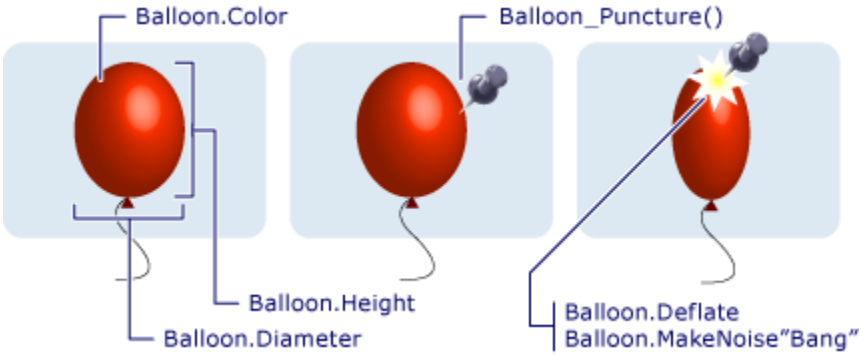
Todos los objetos en el lenguaje de Visual Basic, incluidos los formularios y controles, tienen sus propias propiedades, métodos y eventos. Las propiedades pueden considerarse como los atributos de un objeto, los métodos como sus acciones y los eventos como sus respuestas.

Un objeto corriente como un globo de helio también tiene propiedades, métodos y eventos. Las propiedades de un globo incluyen atributos visibles como su alto, diámetro y color. Otras propiedades describen su estado (inflado o desinflado) o atributos que no se ven, como su edad. Todos los globos tienen estas propiedades, aunque sus valores pueden diferir de un globo a otro.

Un globo también tiene métodos o acciones conocidas que puede realizar. Tiene un método para inflarse (llenarlo con helio), un método para desinflarse (expulsar su contenido) y un método para elevarse (soltarlo). Asimismo, todos los globos pueden tener estos métodos.

Los globos también tienen respuestas a ciertos eventos externos. Por ejemplo, un globo responde al evento de ser pinchado desinflándose o al evento de ser soltado elevándose.

Propiedades, métodos y eventos



Un globo tiene propiedades (Color, Alto y Diámetro), responde a eventos (Puncture) y puede ejecutar métodos (Deflate, MakeNoise).

Propiedades

Si pudiera programar un globo, el código de Visual Basic Expresspodría parecerse al siguiente "código" que establece las propiedades de un globo.

```
Balloon.Color = Red
```

```
Balloon.Diameter = 10

Balloon.Inflated = True
```

Observe el orden del código: el objeto (Globo) seguido por la propiedad (Color) seguida por la asignación del valor (= Rojo). Puede cambiar el color del globo sustituyendo un valor diferente.

Métodos

Los métodos de un globo se *denominan* de este modo.

```
Balloon.Inflate

Balloon.Deflate

Balloon.Rise(5)
```

El orden es parecido al de una propiedad: el objeto (un nombre), seguido por el método (un verbo). En el tercer método, hay un elemento adicional, llamado *argumento*, que especifica la distancia a que se elevará el globo. Algunos métodos tendrán uno o más argumentos para describir aún más la acción que se va a realizar.

Eventos

El globo podría responder a un evento de la siguiente manera.

Copiar código

```
Sub Balloon_Puncture()
    Balloon.MakeNoise("Bang")
    Balloon.Deflate
    Balloon.Inflated = False
End Sub
```

En este caso, el código describe el comportamiento del globo cuando se produce un evento Puncture: llama al método MakeNoise con un argumento "Bang", (el tipo de ruido a realizar), luego llama al método Deflate. Puesto que el globo ya no está inflado, la propiedad Inflated se establece en **False**.

Si bien, en realidad no puede programar un globo, puede programar un formulario o control de Visual Basic. Como programador, es el responsable. Decida las propiedades que se deben cambiar, los métodos que se deben invocar o los eventos que se deben responder para lograr la apariencia y el comportamiento deseados.

Paso 4: agregar código de Visual Basic

En la lección anterior, se utilizó la ventana **Propiedades** para configurar las propiedades de los controles en el formulario. En esta lección, se agregará el *código* que controlará las funciones del programa.

Para agregar el código y la funcionalidad al programa

En el **Diseñador de Windows Forms**, haga doble clic en el control Button .

Se abre una nueva ventana denominada Editor de código. Aquí es donde se agrega todo el código para el programa.

En el Editor de código, escriba lo siguiente.

Visual Basic Express — Copiar código

```
WebBrowser1.Navigate(Textbox1.Text)
```

Este código *se ejecutará* cuando los usuarios hagan clic en el botón.

Información detallada

Es posible que haya observado que cuando se abre el Editor de código, éste ya contiene algún código con el siguiente aspecto:

```
Private Sub Button1_Click(ByVal sender As
System.Object...

|

End Sub
```

Este código es un *controlador de eventos*, también denominado procedimiento *Sub*. Cualquier código dentro de este procedimiento (entre **Sub** y **End Sub**) se ejecuta cada vez que se haga clic en el botón. También puede haber observado que el cursor se encontraba dentro del procedimiento de evento, de manera que todo lo que tuvo que hacer fue escribir.

El código que escribió (`WebBrowser1.Navigate(TextBox1.Text)`) le indica al programa que utilice el *método* **Navigate** del control (denominado WebBrowser1) **WebBrowser** con un *argumento* de **TextBox1.Text** (el valor contenido en la propiedad **Text** del control **TextBox**). Para obtener más información sobre las propiedades, los métodos y los eventos vea Información detallada: comprender propiedades, métodos y eventos

Si no entiende el código, no se preocupe, aprenderá mucho más sobre la escritura de código en las siguientes lecciones

Paso 5: ejecutar y probar un programa

Ahora que el programa está terminado, es hora de ejecutarlo y probarlo. Para programas complejos, la prueba puede ser un proceso largo y difícil, que se analizará en detalle en una lección posterior. Afortunadamente, en este programa todo lo que debe hacer es ejecutarlo.

Para ejecutar el programa

Conecte su equipo a Internet.

En el menú **Depurar** del IDE de Visual Basic, haga clic en **Iniciar depuración**.

Este comando ejecuta el programa.

> Sugerencia
>
> El acceso directo para ejecutar el programa es F5.

En el cuadro de texto, escriba `http://www.microsoft.com/spanish` y haga clic en el botón **Ir**.

El control WebBrowser del programa va a la página principal de Microsoft. Desde allí, puede desplazarse por cualquier vínculo relacionado. Para visitar otra página Web, escriba la dirección en el cuadro de texto y haga clic en el botón **Ir**.

Para cerrar el programa, en el menú **Depuración**, haga clic en **Detener depuración**.

> Sugerencia
>
> También puede finalizar el programa haciendo clic en el botón Cerrar situado en la esquina superior derecha del formulario.

Información detallada

En esta lección se ejecutó el programa para ver si funcionaba. Para la mayoría de los programas de Visual Basic, se repetirá este proceso muchas veces. Generalmente, después de agregar nuevo código, ejecutará el programa para ver si el código lleva a cabo la acción que se espera; de lo contrario, deberá corregirlo. Este proceso se llama *depuración*; se analizará en detalle en una lección posterior.

Puede parecer increíble que el programa vaya a una página Web y que la muestre, como resultado de escribir una sola línea de código. Esto es lo bueno de Visual Basic, todo el código necesario se integra en el control **WebBrowser**, lo que le permite ahorrar tiempo y esfuerzo. Si tuviera que hacerlo todo personalmente, tomaría cientos o incluso miles de líneas de código.

Solución de problemas

Si el programa no se ejecuta ni muestra la página Web, hay algunas cosas que puede comprobar:

Asegúrese de estar conectado a Internet. Abra Internet Explorer e intente desplazarse a la página principal de Microsoft. Si funciona en Internet Explorer, también debe funcionar en el programa.

Asegúrese de que escribió la dirección (`http://www.microsoft.com/spanish`) correctamente.

Regrese y compruebe Paso 2: Crear una interfaz de usuario y asegúrese de que colocó los controles correctos en el formulario.

Regrese a Paso 4: agregar código de Visual Basic y asegúrese de que escribió el código correctamente.

Pasos siguientes

¡Enhorabuena! Ha completado su primer programa de Visual Basic. Ha mostrado cómo se pueden desarrollar programas eficaces en forma rápida y fácil utilizando Visual Basic. En las siguientes lecciones, se presentarán más características del lenguaje de programación de Visual Basic.

Introducción al lenguaje de programación Visual Basic

Microsoft Visual Basic Express es una manera rápida y sencilla de crear programas para Microsoft Windows. Aunque no tenga experiencia de programación en Windows, con Visual Basic Expressdispone de un completo conjunto de herramientas para simplificar las tareas de desarrollo.

¿Y qué es Visual Basic? "Visual" hace referencia al método utilizado para crear lo que ve el usuario, la *interfaz gráfica de usuario* o GUI. "Basic" hace referencia al lenguaje de programación BASIC, de Beginners All-Purpose Symbolic Instruction Code (Código de Instrucción Simbólico Todo Propósito para Principiantes), un lenguaje utilizado por más programadores que cualquier otro lenguaje en la historia de la informática. Puede crear programas útiles sólo con aprender algunas de sus características. Los vínculos siguientes le servirán para empezar a programar en Visual Basic; cada vínculo incluye ejemplos, así como acceso a información adicional.

Programar los conceptos

¿Qué es exactamente un lenguaje de programación? Los vínculos siguientes le darán cierta información general sobre lo que es un lenguaje y cómo almacena diferentes tipos de información.

Término	Definición
Conceptos básicos: funcionamiento de la programación	Cómo funciona un lenguaje de programación y terminología básica.
Representación de palabras, números y valores con variables	Cómo las variables almacenan valores y representan información, así como la manera de utilizarlas.
Palabras y texto: utilizar variables de cadena para organizar palabras	Cómo utilizar una variable String para representar palabras y texto.
Matrices: variables que representan más de un valor	Cómo utilizar una variable Array para representar varios valores del mismo tipo.
Aritmética: crear expresiones con variables y operadores	Cómo escribir código que realiza operaciones aritméticas.
Comparaciones: Utilizar expresiones para comparar valores	Cómo escribir código que compara valores numéricos.

Su primer programa

¿Preparado para un poco de programación real? Los vínculos siguientes le guiarán por el proceso de creación de un programa simple y le mostrarán cómo buscar los errores del programa.

Término	Definición
Hacer que el equipo haga algo: escribir el primer procedimiento	Cómo escribir código que indica a su programa que realice una acción determinada.
Hacer que un programa repita acciones: establecer bucles	Cómo escribir código que repite acciones en su programa y cuenta las veces que éstas se han

For...Next	realizado.
Hacer que un programa elija entre dos posibilidades: la instrucción If...Then	Cómo escribir código que hace cosas diferentes en respuesta a condiciones diferentes.
Qué hacer cuando algo sale mal: control de errores	Cómo escribir código que controla los errores de sus programas. También obtendrá información sobre los diferentes tipos de errores.

Más sobre Visual Basic

Los vínculos siguientes le ayudarán a aumentar su conocimiento de programación y de Visual Basic Express.

Término	Definición
Información detallada: comprender propiedades, métodos y eventos	Cómo funcionan las propiedades, los métodos y los eventos.
Información detallada: tipos de datos	Cómo se almacenan los datos utilizando los diferentes tipos de variables.
Información detallada: convertir un tipo de variable en otro	Cómo convertir datos de un tipo en otro, junto con algunos errores comunes de este proceso.
Información detallada: utilizar Do...While y Do...Until para repetir hasta obtener una condición	Cómo utilizar las instrucciones Do...While y Do...Until para repetir código basándose en ciertas condiciones.
Información detallada: utilizar Select Case para decidir entre varias opciones	Cómo ejecutar código basándose en varias condiciones donde hay muchas elecciones.
Paseo con guía por Visual Basic	Más cosas que puede hacer con el lenguaje de programación Visual Basic Express

Conceptos básicos: funcionamiento de la programación

Antes de comenzar el aprendizaje del *lenguaje de programación* Visual Basic, puede ser útil comprender lo que es un lenguaje de programación y cómo

funciona, incluso alguna terminología de programación. El mejor punto de partida es comenzar con los conceptos básicos.

Cómo funciona la programación

Por sí solo, un equipo no es muy inteligente.

Esencialmente, un equipo es sólo un gran grupo de pequeños modificadores electrónicos que están activados o desactivados. Al establecer diferentes combinaciones de estos modificadores, se logra que el equipo realice alguna acción, por ejemplo, que muestre algo en la pantalla o que emita un sonido. Eso es la programación en su concepto más básico: decirle a un equipo qué hacer.

Claro está que comprender qué combinación de modificadores logrará que el equipo haga lo que se desea será una gran tarea; aquí es donde los lenguajes de programación adquieren un papel importante.

¿Qué es un lenguaje de programación?

Las personas se expresan utilizando un lenguaje con muchas palabras. Los equipos utilizan un lenguaje simple que consta sólo de números 1 y 0, con un 1 que significa "activado" y un 0 que significa "desactivado". Tratar de hablar con un equipo en su propio lenguaje sería como tratar de hablar con los amigos utilizando el código Morse, se puede hacer, pero ¿para qué?

Un lenguaje de programación actúa como un traductor entre el usuario y el equipo. En lugar de aprender el lenguaje nativo del equipo (conocido como *lenguaje máquina*), se puede utilizar un lenguaje de programación para dar instrucciones al equipo de un modo que sea más fácil de aprender y entender.

Un programa especializado conocido como *compilador* toma las instrucciones escritas en el lenguaje de programación y las convierte en lenguaje máquina. Esto significa que, como desarrollador de Visual Basic, no precisa entender lo que el equipo hace o cómo lo hace, sólo es necesario entender cómo funciona el lenguaje de programación de Visual Basic.

Descripción general del lenguaje Visual Basic

En mucho sentidos, el lenguaje Visual Basic Expresses muy parecido al lenguaje cotidiano. Cuando se habla o escribe, se utilizan diferentes tipos de palabras, como nombres o verbos, que definen cómo se utilizan. Visual Basic Expresstambién tiene diferentes tipos de palabras, conocidas como *elementos de programación*, que definen cómo se utilizan para escribir programas.

Los elementos de programación de Visual Basic Expressincluyen *instrucciones*, *declaraciones*, *métodos*, *operadores* y *palabras clave*. A medida que avance en las siguientes lecciones, irá aprendiendo más sobre estos elementos y cómo utilizarlos.

El lenguaje escrito y hablado también tiene reglas, o *sintaxis*, que definen el orden de las palabras en una frase. Visual Basic Expresstambién tiene su sintaxis, al comienzo resulta extraña pero realmente es muy simple. Por ejemplo, para decir "La velocidad máxima de mi automóvil es 55", se escribiría:

```
Car.Speed.Maximum = 55
```

Más adelante aprenderá más sobre la sintaxis y las herramientas de Visual Basic, por ejemplo *IntelliSense*, que es una guía para utilizar la sintaxis correcta al escribir los programas.

El lenguaje hablado y escrito también posee su estructura: por ejemplo, un libro consta de capítulos con párrafos que contienen frases. Los programas escritos en Visual Basic Expresstambién tiene una estructura: los *módulos* son como los capítulos, los *procedimientos* como los párrafos y las *líneas de código* como las frases.

Representación de palabras, números y valores con variables

Las *variables* son un concepto importante en programación. Una variable es una letra o un nombre que puede almacenar un valor. Al crear programas, se pueden utilizar variables para almacenar números, por ejemplo, el alto de un edificio, o palabras, por ejemplo, el nombre de una persona. Resumiendo, se pueden utilizar variables para representar cualquier tipo de información que el programa necesite.

Puede surgir la pregunta, "¿Por qué utilizar una variable cuando en su lugar se puede utilizar simplemente la información?" Como su nombre indica, las variables pueden cambiar el valor que representan cuando el programa se está ejecutando. Por ejemplo, puede escribir un programa para realizar el seguimiento del número de caramelos que tiene en un frasco en su escritorio. Dado que los dulces se comen, es probable que la cantidad de caramelos del frasco cambie con el tiempo. En lugar de volver a escribir el programa cada vez que quiere comerse un caramelo, puede representar el número de caramelos con una variable que cambia con el tiempo.

Almacenar información en variables

Hay tres pasos para utilizar una variable:

Declarar la variable. Indicar al programa el nombre y el tipo de variable que se desea utilizar.

Asignar la variable. Proporcionar un valor a la variable.

Utilizar la variable. Recuperar el valor contenido en la variable y utilizarlo en el programa.

Declarar variables

Cuando declara una variable, tiene que decidir cómo llamarla y qué *tipo de datos* asignarle.

Se declara una variable utilizando las *palabras clave* **Dim** y **As**, como se muestra a continuación.

Visual Basic Express — Copiar código

```
Dim aNumber As Integer
```

Esta línea de código indica al programa que se desea utilizar una variable denominada `aNumber`, que almacene números enteros (el tipo de datos **Integer**).

Puesto que `aNumber` es un **Integer**, sólo puede almacenar números enteros. Por ejemplo, si desea almacenar 42,5 utilizará el tipo de datos **Double**. Y si desea almacenar una palabra, utilizará un tipo de datos **String**. Otro tipo de datos que vale la pena mencionar en este punto es **Boolean**, que puede almacenar un valor `True` o `False`.

Aquí hay más ejemplos de cómo declarar las variables.

Visual Basic Express — Copiar código

```
Dim aDouble As Double
Dim aName As String
Dim YesOrNo As Boolean
```

Asignar variables

Se asigna un valor a la variable con el signo =, que a veces se denomina *operador de asignación*, como se muestra en el ejemplo siguiente.

Visual Basic Express — Copiar código

```
aNumber = 42
```

Esta línea de código toma el valor `42` y lo almacena en la variable declarada previamente denominada `aNumber`.

Declarar y asignar variables con un valor predeterminado

Como se muestra anteriormente, puede declarar una variable en una línea de código y asignar más tarde el valor en otra línea. Esto puede producir un error si intenta utilizar la variable antes de asignarle un valor.

Por esa razón, es mejor declarar y asignar las variables en una línea única. Aunque no sepa aún el valor que contendrá la variable, puede asignar un valor predeterminado. El código para declarar y asignar las mismas variables mostradas anteriormente será similar al siguiente.

Visual Basic Express — Copiar código

```
Dim aDouble As Double = 0
Dim aName As String = "default string"
Dim YesOrNo As Boolean = True
```

Si declara las variables y asigna los valores predeterminados en una sola línea, puede evitar posibles errores. Todavía puede utilizar la asignación para dar después un valor diferente a la variable.

Inténtelo

En este ejercicio, escribirá un breve programa que crea cuatro variables, les asigna valores y a continuación muestra cada valor en una ventana llamada *cuadro de mensaje*. Comencemos creando el proyecto donde se almacenará el código.

Para crear el proyecto

Si aún no está abierto, abra Visual Basic Expressen el menú **Inicio** de Windows.

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Variables` y haga clic en **Aceptar**.

Visual Basic Expresscreará los archivos para el programa y abrirá el **Diseñador de Windows Forms**.

A continuación, creará las variables.

Para crear las variables y mostrar sus valores

Haga doble clic en el formulario.

El Editor de código abrirá una sección de código llamada `Form1_Load`. Esta sección de código, denominada *procedimiento*, contiene las instrucciones que se llevarán a cabo cuando se cargue por primera vez el formulario en la memoria.

En el procedimiento `Form1_Load`, escriba el siguiente código.

Visual Basic Express — Copiar código

```
Dim anInteger As Integer = 42
Dim aSingle As Single = 39.345677653
Dim aString As String = "I like candy"
Dim aBoolean As Boolean = True
```

Este código declara cuatro variables: **Integer**, **Single**, **String** y **Boolean—**, y asigna sus valores predeterminados.

Sugerencia

Al escribir el código, quizá haya observado que después de escribir As, aparece una lista de palabras bajo el cursor. Esta característica se llama Intellisense. Le permite escribir las primeras letras de una palabra y seleccionarla en la lista. Una vez seleccionada, presione la tecla TAB para finalizar la palabra.

Nota

Siempre que representa texto real en un programa, éste debe aparecer entre comillas (""). Esto le dice al programa que interprete el texto como texto real en lugar de interpretarlo como un nombre de variable. Cuando asigna a una variable Boolean un valor True o False, no pone la palabra entre comillas, porque True y False son palabras clave de Visual Basic, con significados especiales propios.

Debajo del código que escribió en el paso anterior, escriba lo siguiente.

Visual Basic Express — Copiar código

```
MsgBox(anInteger)
MsgBox(aSingle)
```

```
MsgBox(aString)
MsgBox(aBoolean)
```

Visual Basic Express | Copiar código

```
End
```

Las primeras cuatro líneas del código le dicen al programa que muestre cada valor que asignó en el paso anterior en una nueva ventana, utilizando la *función* **MsgBox**. La línea final indica al programa que finalice después de ejecutar este procedimiento: se utiliza la *instrucción* **End**.

Presione F5 para ejecutar el programa.

Haga clic en el botón **Aceptar** en cada ventana según aparezcan. Observe que sucesivamente se muestra el valor de cada variable y, a continuación, el programa finaliza. Después de que el programa finaliza, puede regresar y cambiar los valores que se asignaron en el código y ejecutar la aplicación de nuevo: verá que se muestran los nuevos valores.

Información detallada: tipos de datos

Los tipos de datos en Visual Basic Expressdeterminan la clase de valores o datos que se puede almacenar en una variable, y cómo se almacenan esos datos. ¿Por qué hay tipos de datos diferentes? Piense en ello de esta manera: si tiene tres variables, dos de las cuales contienen números y la tercera contiene un nombre, puede realizar operaciones aritméticas con las dos primeras, pero no puede realizarlas con la que almacena el nombre. Asignar un tipo de datos a una variable facilita determinar cómo se puede, o no se puede, utilizar la variable.

Nota

Los tipos de datos también se utilizan en otros elementos de programación como constantes, propiedades y funciones. Obtendrá más información sobre los otros usos de los tipos de datos en una lección posterior.

Tipos de datos para números

La mayoría de los programas informáticos tratan con números de una forma u otra. Hay varias maneras diferentes de expresar números, Visual Basic Expresscuenta con varios tipos de datos numéricos que tratan eficazmente con números.

El tipo de dato numérico que más se utiliza es **Integer**, utilizado para representar un número entero (un número sin parte fraccionaria). Cuando se elige un tipo de datos para representar números enteros, hay que utilizar el tipo de datos **Long** si la variable va a almacenar números mayores que dos mil millones; de lo contrario, el tipo **Integer** es más útil.

No todos los números son números enteros; por ejemplo, cuando se divide dos números enteros, el resultado es a menudo un número entero más una fracción (9 dividido por 2 es igual a 4,5). El tipo de datos **Double** se utiliza para representar números que tienen una parte fraccionaria.

Nota

Hay tipos de datos numéricos adicionales como Decimal, Short, SByte y UInteger; éstos se suelen utilizar en programas muy grandes donde puede ser un problema utilización de la memoria o la velocidad. De momento, los tipos de datos numéricos básicos es todo lo que va a necesitar. Si desea obtener más información sobre los tipos de datos avanzados, vea Tipos de datos numéricos.

Tipos de datos para texto

La mayoría de los programas también tratan con texto, ya sea mostrando información al usuario o capturando texto escrito por el usuario. El texto normalmente se almacena en el tipo de datos **String**, que puede contener una serie de letras, números, espacios y otros caracteres. El tipo **String** puede tener cualquier longitud, desde una frase o un párrafo a sólo un carácter o nada en absoluto (*cadena nula*).

Por cada variable que representa un único carácter, también hay un tipo de datos **Char**. Si sólo necesita contener un carácter en una única variable, utilice el tipo de datos **Char** en lugar de **String**.

Otros tipos de datos

Además de texto y números, los programas a veces necesitan almacenar otros tipos de información, como un valor verdadero o falso, una fecha, o datos que tienen un significado especial para el programa.

Para aquellos valores que se pueden representar como verdadero/falso, sí/no o activado/desactivado, Visual Basic Expresscuenta con el tipo de datos **Boolean**. Una variable **Boolean** puede contener uno de dos valores posibles: **True** o **False**.

Aunque se pueden representar las fechas y horas como números, el tipo de datos **Date** facilita la tarea de calcular fechas u horas, como, por ejemplo, el número de días que quedan hasta su cumpleaños o el número de minutos que faltan hasta la hora de comer.

Si necesita almacenar más de un tipo de datos en una única variable, puede utilizar un tipo de datos *compuesto*. Los tipos de datos compuestos incluyen *matrices*, *estructuras* y *clases*. Más adelante obtendrá más información sobre estos tipos de datos.

Finalmente, hay casos en los que el tipo de datos que necesita almacenar debe ser distinto en momentos diferentes. El tipo de datos **Object** le permite declarar una variable y a continuación definir después su tipo de datos. También obtendrá más información sobre el tipo de datos **Object** en una lección posterior.

Palabras y texto: utilizar variables de cadena para organizar palabras

En esta lección, aprenderá a utilizar el tipo de datos **String** para representar palabras y texto.

En la lección anterior, aprendió a utilizar las variables para almacenar los datos en el programa y que cada variable debe ser del tipo adecuado para los datos que almacenará. En esta lección, aprenderá sobre el tipo de datos **String** utilizado para almacenar el texto.

¿Qué es una cadena?

Una *cadena* es cualquier serie de caracteres de texto, como letras, números, caracteres especiales y espacios. Las cadenas pueden ser frases y oraciones legibles, como "El rápido zorro de color café salta sobre el perro perezoso" o una combinación aparentemente ininteligible como "@#fTWRE^3 35Gert".

Las variables **String** se crean del mismo modo que otras variables: declarando primero la variable y asignándole un valor, como se muestra a continuación.

Visual Basic Express | Copiar código

```
Dim aString As String = "This is a string"
```

Al asignar un texto real (también denominado *literales de cadena*) a una variable **String**, el texto debe estar entre comillas (**""**). También puede utilizar el carácter = para asignar una variable **String** a otra variable **String**, como se muestra en este ejemplo.

Visual Basic Express | Copiar código

```
    Dim aString As String = "This is a string"
...
    Dim bString As String = ""
```

```
bString = aString
```

El código anterior establece el valor de `bString` en el mismo valor que `aString` (`This is a string`).

Puede utilizar el carácter `&` para combinar dos o más cadenas secuencialmente en una nueva cadena, como se muestra a continuación.

Visual Basic Express — Copiar código

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & bString
```

El ejemplo anterior declara tres variables de **String** y asigna respectivamente "Across the Wide" y "Missouri" a las dos primeras y luego asigna los valores combinados de las dos primeras a la tercera variable. ¿Cuál cree que es el valor de `cString`? Le puede sorprender saber que el valor es `Across the WideMissouri` porque no hay ningún espacio al final de `aString` o al principio de `bString`. Las dos cadenas están simplemente unidas. Si desea agregar espacios o cualquier otro símbolo entre dos cadenas, debe hacerlo mediante un *literal de cadena*, como `" "`, como se muestra a continuación.

Copiar código

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & " " & bString
```

El texto contenido ahora en `cString` dice `Across the Wide Missouri`.

Inténtelo

Para unir las cadenas

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**:

En el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba **Concatenación**.

Haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el procedimiento del evento **Form1.Load**, declare cuatro variables de cadena y asigne los valores de cadena, como se muestra a continuación:

Visual Basic Express — Copiar código

```
Dim aString As String = "Concatenating"
Dim bString As String = "Without"
Dim cString As String = "With"
Dim dString As String = "Spaces"
```

Agregue el siguiente código para concatenar las cadenas y mostrar los resultados:

Visual Basic Express — Copiar código

```
    MsgBox(aString & bString & dString)
    ' Displays "ConcatenatingWithoutSpaces"
...
    MsgBox(aString & " " & cString & " " & dString)
    ' Displays "Concatenating With Spaces"
```

El texto mostrado en el cuadro de mensaje es el resultado de la unión de las variables de cadena asignadas en un paso anterior. En el primer cuadro, las cadenas están unidas sin espacios. En el segundo, los espacios se insertan explícitamente entre cada cadena.

Matrices: variables que representan más de un valor

En esta lección, aprenderá a utilizar *matrices* para almacenar grupos de valores.

Como aprendió en las lecciones anteriores, las variables se utilizan para almacenar diferentes tipos de datos que el programa utiliza. Hay otro tipo de variable denominado *matriz* que proporciona una manera conveniente de almacenar diversos valores del mismo tipo.

Por ejemplo, suponga que está escribiendo un programa para un equipo de béisbol y desea almacenar los nombres de todos los jugadores que se encuentran en el campo de juego. Puede crear nueve variables de cadenas separadas, una para cada jugador o puede declarar una variable de matriz que se parezca al código que aparece a continuación.

Visual Basic Express Copiar código

```
Dim players() As String
```

Una variable de matriz se declara colocando paréntesis después del nombre de la variable. Si se sabe cuántos valores se necesita almacenar, también se puede especificar el tamaño de la matriz en la declaración de la siguiente manera.

Visual Basic Express Copiar código

```
Dim players(8) As String
```

Puede parecer extraño que el tamaño de la matriz sea `8` cuando un equipo del béisbol tiene `9` jugadores. Esto se debe a que la matriz está formada por una cantidad de valores o *elementos*, que comienzan con el elemento 0 y terminan con el número especificado en la declaración. En este caso, la matriz contiene los elementos 0 a 8, de un total de nueve.

Asignar valores a las matrices

Al igual que con otros tipos de valores, debe asignar valores a las matrices. Para ello, se hace referencia al número del elemento como parte de la asignación, como se muestra a continuación.

Visual Basic Express Copiar código

```
players(0) = "John"
players(3) = "Bart"
```

En el código anterior, el valor `John` se asigna al primer elemento de la matriz (elemento 0) y el valor `Brett` se asigna al cuarto elemento (elemento 3). Los elementos de la matriz no se tienen que asignar en orden y cualquier elemento sin asignar tendrá un valor predeterminado; en este caso, una cadena vacía.

Al igual que con otros tipos de valores, puede declarar y asignar los valores a una matriz en una línea única de la siguiente manera.

Visual Basic Express Copiar código

```
Dim players() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

En este caso, las llaves indican una lista de valores. Los valores se asignan a los elementos en el orden mostrado. Observe que no se especifica el tamaño de la matriz, lo determina el número de elementos que se muestran.

Recuperar valores de las matrices

Así como se utilizan números para especificar la posición de un elemento en una matriz, el número de elementos se utiliza para especificar qué valor desea recuperar.

Visual Basic Express — Copiar código

```
Dim AtBat As String
AtBat = players(3)
```

El código anterior recupera el cuarto elemento de la matriz y lo asigna a la variable de cadena `AtBat`.

Inténtelo

Para almacenar los valores en una matriz

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `MyFirstArray` y, a continuación, haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **Textbox** al formulario.

En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el procedimiento de evento **Button1_Click**, agregue el siguiente código:

Visual Basic Express — Copiar código

```
Dim players() As String = {"Dan", "Fred", "Bart", "Carlos", _
  "Ty", "Juan", "Jay", "Sam", "Pedro"}
```

```
Dim i As Integer = CInt(Textbox1.Text)
MsgBox(players(i) & " is on first base.")
```

Observe que el código anterior utiliza la función **CInt** para convertir el valor **String** (`TextBox1.Text`) en un **Integer** (`i`). Puede obtener más información sobre las conversiones en Información detallada: convertir un tipo de variable en otro.

Presione F5 para ejecutar el programa.

Escriba un número comprendido entre 0 y 8 en el cuadro de texto y haga clic en el botón. Se muestra el nombre que corresponde a ese elemento en un cuadro de mensaje

Aritmética: crear expresiones con variables y operadores

En esta lección, aprenderá a crear expresiones para realizar operaciones aritméticas y devolver valores.

Una *expresión* es un segmento de código que realiza operaciones aritméticas y, a continuación, devuelve un valor. En el siguiente ejemplo se muestra una expresión de suma simple.

5 + 4

Cuando se evalúa, la expresión `5 + 4` devuelve el valor `9` y se compone de dos partes: los *operandos* (`5` y `4`), que son los valores en los que se realiza la operación, y el *operador* (`+`), que especifica la operación que se va a realizar.

Utilizar valores devueltos por expresiones

Para que una expresión sea útil, se debe realizar una acción con el valor que se devuelve. Lo más común es asignar el valor a una variable, tal como se muestra a continuación.

Visual Basic Express — Copiar código

```
Dim anInteger As Integer = 5 + 4
```

Este ejemplo declara una nueva variable **Integer**, llamada `anInteger` y le asigna el valor devuelto por `5 + 4`.

Operadores aritméticos

Las expresiones habitualmente se utilizan para realizar operaciones aritméticas con variables: suma, resta, multiplicación o división. La tabla siguiente describe los operadores normalmente utilizados para operaciones aritméticas.

Operador	Descripción	Ejemplo
+ (suma)	Devuelve la suma de dos operandos	5 + 4
- (resta)	Devuelve la diferencia de dos operandos	5 - 4
* (multiplicación)	Devuelve el producto de dos operandos	5 * 4
/ (división)	Devuelve el cociente de dos operandos	5 / 4

El tipo de variable que se utiliza al realizar la operación aritmética puede afectar el resultado. La división de dos números a menudo da como resultado un valor que no es un número entero. Por ejemplo, cuando se divide 3 por 2, el resultado es 1,5. Si se asigna el valor devuelto de esa expresión a una variable **Integer**, se redondeará al número entero más cercano. Al realizar la división, se debe utilizar una variable **Double** para almacenar el valor devuelto.

Nota

También se puede convertir una variable de un tipo de datos en otra mediante las funciones de conversión de Visual Basic. Para obtener más información, vea Información detallada: convertir un tipo de variable en otro.

Inténtelo

Para sumar números

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Arithmetic` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre dos controles **Textbox** al formulario.

En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el procedimiento de evento **Button1_Click**, escriba el siguiente código.

Visual Basic Express | Copiar código

```
Dim A As Double = Textbox1.Text
Dim B As Double = Textbox2.Text


MsgBox(A + B)
MsgBox(A - B)
MsgBox(A * B)
MsgBox(A / B)
```

Las primeras dos líneas declaran las variables `A` y `B`, que contendrán los valores numéricos utilizados en este programa y asignarán los valores de los dos controles **TextBox** (su texto) a las variables `A` y `B`.

Las cuatro líneas finales crean expresiones con las dos variables y cada uno de los operadores aritméticos básicos y muestran los resultados de esas expresiones en un cuadro de mensaje.

Presione F5 para ejecutar la aplicación.

Escriba un número en cada uno de los cuadros de texto y haga clic en **Button1**.

Nota

Si escribe algún otro carácter en los cuadros de texto, se producirá un error.

Las expresiones se crean utilizando los dos números que se escriben y cada uno de los cuatro operadores aritméticos básicos (suma, resta, multiplicación y división). El resultado de cada expresión se muestra en un cuadro de mensaje

Información detallada: convertir un tipo de variable en otro

Como ha visto, hay variables de diferentes tipos. El tipo determina la clase de datos que puede contener una variable. Una variable de tipo **Integer** sólo puede contener datos numéricos sin separadores decimales. Una variable de tipo **String** sólo puede contener texto.

¿Qué pasa cuándo desea mostrar un valor **Integer** en un control TextBox que requiere una variable de tipo **String**? La respuesta es que los datos se deben convertir de un tipo a otro. En este tema, estudiará cómo convertir los datos de un tipo en otro y aprenderá algunas técnicas utilizadas para la conversión de datos, así como algunos de sus problemas habituales.

Convertir variables en texto

Cada variable de Visual Basic Expressse puede convertir en texto utilizando una función especial llamada **CStr** (que viene de abreviar **C**onvert to **Str**ing). Esta función, como el nombre implica, devuelve los datos representados por la variable como de tipo **String**. El procedimiento siguiente muestra un ejemplo sencillo de convertir un valor **Integer** en texto.

¡Inténtelo!

Para convertir una variable en texto

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Conversion` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador del evento **Form1_Load**, escriba el siguiente código.

Visual Basic Express — Copiar código

```
Dim anInteger As Integer = 54
MsgBox(CStr(anInteger))
```

Este código declara una variable de tipo entero llamada `anInteger`, le asigna un valor de 54 y, a continuación, convierte ese valor en texto y lo muestra en un cuadro de mensaje llamando a la función **CStr**.

Presione F5 para compilar y ejecutar la aplicación. Aparece un cuadro de mensaje que indica `54`.

Probemos algo sólo por diversión. En el Editor de código, cambie la línea que indica `MsgBox(CStr(anInteger))` para que indique `MsgBox(anInteger)` y presione F5 para ejecutarla. ¿Qué pasa? El programa se comporta exactamente como hizo antes. Visual Basic Expresses lo suficientemente inteligente como para saber que lo que desea realmente es

convertir la variable de tipo **Integer** en texto para que se muestre en el cuadro de mensaje. Sin embargo, no puede confiar en este comportamiento para todas las clases: existen muchos tipos de variables que no se pueden convertir automáticamente. Por consiguiente, es una buena práctica usar siempre la función **CStr**, aun cuando una variable se convertiría automáticamente en texto.

Además de la conversión de variables **Integer** en texto, se puede utilizar la función **CStr** en cualquier tipo de dato numérico, como **Double** o **Long**. También se puede utilizar para convertir la información de **Date** y tipos de datos **Boolean** en texto. Para obtener más información sobre los tipos de datos, vea Información detallada: tipos de datos.

Conversión entre tipos de datos numéricos

Como aprendió en la lección aritmética, a veces el resultado de una operación aritmética no se puede expresar como un valor de tipo **Integer**. Así como Visual Basic Expresstiene una función para convertir los números en texto, también tiene funciones para convertir las variables de un tipo de datos numéricos en otro. Por ejemplo, puede utilizar la función **CDbl** (de **C**onvert to **Dou**ble) en una operación aritmética para devolver un número fraccionario al trabajar con variables de tipo **Integer**. El procedimiento siguiente muestra cómo utilizar la función **CDbl** al dividir dos enteros.

¡Inténtelo!

Para convertir tipos de datos numéricos

En el Editor de código, elimine el código que escribió en el procedimiento anterior y escriba lo siguiente:

Visual Basic Express — Copiar código

```
Dim A As Integer = 1
Dim B As Integer = 2
MsgBox(CDbl(A / B))
```

Este código declara dos variables de tipo **Integer** (`A` y `B`), les asigna los valores `1` y `2` y, a continuación, convierte el resultado de la operación de división (`A / B`) utilizando la función **CDbl** y lo muestra en un cuadro de mensaje.

Presione F5 para compilar y ejecutar la aplicación. Aparece un cuadro de mensaje que indica `0.5`.

Visual Basic Express también tiene funciones para otros tipos de variables numéricas. Por ejemplo, si agrega dos variables de tipo **Double** y desea redondear el resultado al número entero más cercano, utilice la función **CInt**.

Otras funciones de conversión numéricas son **CByte**, **CDec**, **CLng** y **CShort**. Para obtener una lista de todas las funciones de conversión de Visual Basic, vea Funciones de conversión de tipos.

Comparaciones: Utilizar expresiones para comparar valores

En esta lección, obtendrá información sobre cómo utilizar los *operadores de comparación* para crear expresiones que comparan valores.

En la última lección, obtuvo información sobre cómo utilizar operadores aritméticos para crear expresiones numéricas y devolver valores numéricos. Se puede utilizar otro tipo de operador, los *operadores de comparación*, para comparar valores numéricos y devolver valores **Boolean** (**True** o **False**).

Los operadores de comparación se utilizan frecuentemente para comparar valores y tomar decisiones basadas en esa comparación. La toma de decisiones en el programa se tratará exhaustivamente en Hacer que un programa elija entre dos posibilidades: la instrucción If...Then.

La siguiente tabla resume los operadores de comparación:

Operador	Descripción	Ejemplos
= (igual)	Devuelve True si el número del lado izquierdo es igual al número del lado derecho.	5 = 4 (falso) 4 = 5 (falso) 4 = 4 (verdadero)
<> (distinto de)	Devuelve True si el número de la izquierda no es igual al número de la derecha.	5 <> 4 (True) 4 <> 5 (True) 4 <> 4 (False)
> (mayor que)	Devuelve True si el número de la izquierda es mayor que el número de la derecha.	5 > 4 (True) 4 > 5 (False) 4 > 4 (False)
< (menor que)	Devuelve True si el número de la izquierda es menor que el número de la derecha.	5 < 4 (False) 4 < 5 (True) 4 < 4 (False)

>= (mayor o igual que)	Devuelve True si el número de la izquierda es mayor o igual que el número de la derecha.	5 >= 4 (True) 4 >= 5 (False) 4 >= 4 (True)
<= (menor o igual que)	Devuelve True si el número de la izquierda es menor o igual que el número de la derecha.	5 <= 4 (False) 4 <= 5 (True) 4 <= 4 (True)

Inténtelo

Para comparar expresiones

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Comparison` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre dos controles **Textbox** al formulario.

En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, escriba el siguiente código:

Visual Basic Express — Copiar código

```
Dim A As Double = CDbl(Textbox1.Text)
Dim B As Double = CDbl(Textbox2.Text)
MsgBox(A > B)
MsgBox(A < B)
MsgBox(A = B)
```

Las dos primeras líneas declaran las variables `A` y `B`, que contendrán los valores numéricos usados en este programa; utilizan la instrucción `CDbl` para convertir el texto de `Textbox1` y `Textbox2` en valores numéricos. Finalmente, las últimas tres líneas crean expresiones que permiten comparar las dos variables mediante tres operadores de comparación básicos y muestran los resultados de esas expresiones en tres cuadros de mensaje.

Presione F5 para ejecutar la aplicación.

Escriba un número en cada uno de los cuadros de texto y haga clic en **Button1.**

El primer cuadro de mensaje mostrará **True** si A (el número que escribió en el primer cuadro de texto) es mayor que B (el número que escribió en el segundo cuadro de texto); de lo contrario, mostrará **False**. El segundo cuadro de mensaje mostrará **True** si A es menor que B, y el tercer cuadro de mensaje mostrará **True** si ambos números son iguales.

Pruebe a escribir diferentes números en los cuadros de texto para ver cómo cambian los resultados.

Hacer que el equipo haga algo: escribir el primer procedimiento

En esta lección, aprenderá a crear un *procedimiento*, un bloque de código independiente que se puede ejecutar desde otros bloques de código, y a crear *parámetros* para los procedimientos.

Un procedimiento es simplemente un fragmento de código que indica al programa que realice una acción. Aunque es posible que no lo haya notado, ya se han utilizado procedimientos en las lecciones anteriores. Por ejemplo, la función **MsgBox** tiene un procedimiento integrado que realiza la acción de mostrar un cuadro de diálogo.

Mientras Visual Basic Expresstiene muchos procedimientos integrados para realizar las acciones comunes, siempre habrá casos en que se desea que el programa realice una acción que un procedimiento integrado no puede controlar. Por ejemplo, la función **MsgBox** no puede mostrar un cuadro de diálogo con una imagen. Debe escribir un procedimiento para realizar esta tarea.

¿Qué es un procedimiento?

Un procedimiento es un bloque de código independiente que se puede ejecutar desde otros bloques de código. En general, cada procedimiento contiene el código necesario para realizar una tarea. Por ejemplo, puede tener un procedimiento llamado **PlaySound** que contiene el código necesario para reproducir un archivo de onda. Aunque puede escribir código que reproduzca un sonido cada vez que el programa deba realizar un ruido, tiene más sentido crear un procedimiento único al que se pueda llamar en cualquier parte del programa.

Un procedimiento se *ejecuta llamándolo* en el código. Por ejemplo, para ejecutar el procedimiento **PlaySound**, simplemente se agrega una línea de código al programa con el nombre del procedimiento, como se muestra a continuación.

```
PlaySound
```

Es todo lo que tiene que hacer. Cuando el programa llegue a esa línea, irá al procedimiento **PlaySound** y ejecutará el código contenido allí. A continuación, el programa regresa a la siguiente línea que viene después de la llamada a **PlaySound**.

Puede llamar a tantos procedimientos como desee. Los procedimientos se ejecutan en el orden de llamada. Por ejemplo, podría tener también un procedimiento llamado **DisplayResults**; para ejecutarlo después de ejecutar el procedimiento **PlaySounds**, llame a los procedimientos como se muestra a continuación.

```
PlaySounds

DisplayResults
```

Funciones y Subs

Existen dos tipos de procedimientos: *funciones* y *subrutinas* (llamadas a veces *sub*). Una función devuelve un valor al procedimiento que la llamó, mientras que una subrutina simplemente ejecuta código. Se llama a una subrutina cuando una línea de código, que contiene el nombre de ésta, se agrega al programa como en el siguiente ejemplo.

```
DisplayResults
```

Las funciones son diferentes, porque las funciones no sólo ejecutan códigos, también devuelven un valor. Por ejemplo, imagine una función llamada **GetDayOfWeek** que devuelve un **Integer** que indica el día de la semana. Se llama a esta función primero mediante la declaración de una variable para almacenar el valor devuelto y luego se asigna el valor devuelto a la variable para un uso posterior, tal como se muestra a continuación.

```
Dim Today As Integer

Today = GetDayOfWeek
```

En este ejemplo, el valor devuelto por la función se copia a la variable denominada `Today` y se almacena para un uso posterior.

Escribir procedimientos

Los procedimientos se escriben colocando primero una *declaración de procedimiento*. Una declaración de procedimiento realiza varias acciones: indica si el procedimiento es una función o una subrutina, denomina el procedimiento y detalla todos los parámetros que puede tener (los parámetros se analizarán en detalle más adelante en esta lección). A continuación, se ofrece un ejemplo de una declaración de procedimiento sencilla.

Visual Basic Express | Copiar código

```
Sub MyFirstSub()
End Sub
```

La palabra clave `Sub` indica al programa que este procedimiento es una subrutina y no devolverá un valor. El nombre de la subrutina (`MyFirstSub`) viene a continuación y el paréntesis vacío indica que no hay *parámetros* para este procedimiento. Finalmente, la palabra clave **End Sub** indica el fin de la subrutina. Todos los códigos que tiene que ejecutar esta subrutina van entre estas dos líneas.

Declarar funciones es similar, pero, además, se debe especificar el tipo de valor devuelto (como por ejemplo,**Integer**, **String**, etc.). Por ejemplo, una función que devolvió un valor **Integer** puede ser similar a la siguiente.

Visual Basic Express | Copiar código

```
Function MyFirstFunction() As Integer
End Function
```

Las palabras clave `As Integer` indican que la función devolverá un valor **Integer** . Para devolver un valor desde una función, utilice la palabra clave **Return**, como se muestra en el ejemplo siguiente.

Visual Basic Express | Copiar código

```
Function GetTheNumberOne() As Integer
    Return 1
End Function
```

Este procedimiento devolverá el número 1.

Inténtelo

Para crear procedimientos

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `MyFirstProcedure` y, a continuación, haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el Editor de código, busque la línea que contiene `End Class`. Éste es el fin de la sección de código que compone el formulario. Inmediatamente antes de esta línea, agregue el siguiente procedimiento:

Visual Basic Express — Copiar código

```
Function GetTime() As String
  Return CStr(Now)
End Function
```

Esta función utiliza el procedimiento **Now** integrado para obtener la hora actual, luego utiliza la función **CStr** para convertir el valor devuelto por **Now** en una **String** legible. Finalmente, ese valor **String** se devuelve como el resultado de la función.

Sobre la función que agregó en el paso anterior, agregue el siguiente **Sub**.

Visual Basic Express — Copiar código

```
Sub DisplayTime()
  MsgBox(GetTime)
End Sub
```

Esta subrutina llama a la función `GetTime` y muestra el resultado que devolvió en un cuadro de mensaje.

Finalmente, agregue una línea al controlador de eventos **Form1_Load** que llama a la subrutina **DisplayTime**, como se muestra en el ejemplo.

Visual Basic Express — Copiar código

```
DisplayTime()
```

Presione F5 para ejecutar el programa.

Cuando el programa se inicia, se ejecuta el procedimiento de evento `Form1_Load` . Este procedimiento llama a la subrutina **DisplayTime**, de manera que la ejecución del programa va al procedimiento de subrutina **DisplayTime**. Esa subrutina a su vez llama a la función **GetTime**, por lo que la ejecución del programa va a la función **GetTime**. Esta función devuelve una

String que representa el tiempo para el procedimiento de subrutina **DisplayTime**, el que muestra esa cadena en un cuadro de mensaje. Después de que la subrutina termina de ejecutarse, el programa continúa normalmente y muestra el formulario.

Parámetros en funciones y subrutinas

A veces se deberá proporcionar información adicional a los procedimientos. Por ejemplo, en el procedimiento **PlaySound**, se desea reproducir uno de varios sonidos diferentes. La información acerca de qué sonido reproducir se puede proporcionar utilizando los parámetros.

Los parámetros se parecen mucho a las variables. Tienen un tipo y un nombre y almacenan información al igual que las variables. Se pueden utilizar como variables en un procedimiento. Las dos diferencias principales entre los parámetros y las variables son:

Los parámetros se declaran en la declaración de procedimiento, no en líneas individuales de código.

Sólo se pueden utilizar los parámetros en el procedimiento en el que se declaran.

Los parámetros se declaran en la declaración de procedimiento, en los paréntesis que siguen al nombre del procedimiento. La palabra clave **As** se utiliza para declarar el tipo y la palabra clave **ByVal** precede generalmente a cada parámetro. Visual Basic Expressagregará automáticamente esta palabra clave si no se agrega, ésta tiene una función bastante avanzada que va más allá de los temas tratados en esta lección.

A continuación, se muestra un ejemplo de una subrutina con parámetros.

Visual Basic Express — Copiar código

```
Sub PlaySound(ByVal SoundFile As String, ByVal Volume As Integer)
  My.Computer.Audio.Play(SoundFile, Volume)
End Sub
```

Se llamará a la subrutina con los valores para los parámetros como se muestra a continuación.

Visual Basic Express — Copiar código

```
PlaySound("Startup.wav", 1)
```

También se pueden declarar los parámetros para las funciones exactamente de la misma forma que lo haría con las subrutinas.

Inténtelo

Para crear una función con parámetros

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `parameters` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre dos controles **Textbox** al formulario.

En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

Inmediatamente después de la línea **End Sub** del controlador de eventos `Button1_Click`, agregue el siguiente procedimiento:

Visual Basic Express — Copiar código

```
Function AddTwoNumbers(ByVal N1 As Integer, ByVal N2 As Integer) _
As Integer
  Return N1 + N2
End Function
```

En el procedimiento `Button1_Click`, agregue el código siguiente:

Visual Basic Express — Copiar código

```
Dim aNumber As Integer = CInt(Textbox1.Text)
Dim bNumber As Integer = CInt(Textbox2.Text)
MsgBox(AddTwoNumbers(aNumber, bNumber))
```

Este código declara dos enteros y convierte el texto de los dos cuadros de texto en valores enteros. Luego pasa dichos valores a la función **AddTwoNumbers** y muestra el valor devuelto en un cuadro de mensaje.

Presione F5 para ejecutar el programa.

Escriba un valor numérico en cada cuadro de texto y haga clic en el botón. Se sumarán los dos números y el resultado se mostrará en un cuadro de mensaje.

Hacer que un programa repita acciones: establecer bucles For...Next

En esta lección, aprenderá a utilizar la instrucción **For...Next** para repetir las acciones en el programa y para contar cuántas veces se han realizado estas acciones.

Cuando escribe un programa, debe repetir las acciones con frecuencia. Por ejemplo, suponga que está escribiendo un método que muestra una serie de números en pantalla. Deseará repetir la línea de código que muestra el número las veces que sea necesario.

El bucle **For...Next** le permite especificar un número y repetir un código contenido dentro de ese bucle para el número específico de veces. El siguiente ejemplo muestra cómo aparece un bucle **For...Next** en un código.

Visual Basic Express | Copiar código

```
Dim i As Integer = 0
For i = 1 To 10
  DisplayNumber(i)
Next
```

El bucle **For...Next** comienza con una *variable de contador*, `i`. Ésta es una variable que utiliza el bucle para contar la cantidad de veces que se ha ejecutado. La siguiente línea (`For i = 1 to 10`) le dice al programa cuántas veces se debe repetir el bucle y los valores `i` que va a tener.

Cuando el código entra en el bucle **For...Next**, se inicia con `i` que contiene el primer valor, en este caso 1. El programa ejecuta las líneas de código entre la línea `For` y la línea `Next`, en este caso llamando al método `DisplayNumber` con un parámetro de `i` (en este caso también 1).

Cuando se alcanza la línea `Next`, se agrega `1` a `i` y la ejecución de programa regresa nuevamente a la línea `For`. Esto se repite hasta que el valor de `i` es mayor que el segundo número en la línea `For`, en este caso 10. Cuando esto sucede, el programa continúa con cualquier código después de la línea `Next`.

Inténtelo

Para utilizar la instrucción For...Next

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `ForNext` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **TextBox** y un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, escriba el siguiente código:

Visual Basic Express — Copiar código

```
Dim i As Integer = 0
Dim NumberOfRepetitions As Integer = CInt(Textbox1.Text)
For i = 1 To NumberOfRepetitions
  MsgBox("This line has been repeated " & i & " times")
Next
```

Presione F5 para ejecutar el programa.

En el cuadro de texto, escriba un número y haga clic en el botón.

Aparece un Cuadro de mensaje las veces indicadas en el cuadro de texto.

Información detallada: utilizar Do...While y Do...Until para repetir hasta obtener una condición

En esta lección, aprenderá a utilizar las instrucciones **Do...While** y **Do...Until** para repetir el código basándose en algunas condiciones.

En la lección anterior, aprendió a utilizar la instrucción **For...Next** para recorrer un bloque de código un número específico de veces, pero ¿qué ocurre si el número de veces que el código se debe repetir es diferente para algunas condiciones? Las instrucciones **Do...While** y **Do...Until** permiten repetir un bloque de código mientras cierta condición sea **True** o hasta que cierta condición sea **True**.

Por ejemplo, si se disponía de un programa para agregar una serie de números, pero nunca deseó que la suma de los números fuera mayor que 100. Se podría

utilizar la instrucción **Do...While** para llevar a cabo la suma de la siguiente forma:

Visual Basic Express | Copiar código

```
Dim sum As Integer = 0
Do While sum < 100
  sum = sum + 10
Loop
```

En el código anterior, la línea `Do While` evalúa la variable `sum` para ver si es menor que 100; si lo es, se ejecuta la siguiente línea de código; si no lo es, se desplaza a la línea siguiente del código a continuación de `Loop`. La palabra clave `Loop` le dice al código que regrese a la línea `DoWhile` y evalúe el nuevo valor de `sum`.

Inténtelo

Para utilizar una instrucción Do...While

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `DoWhile` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **TextBox** y un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, escriba el siguiente código:

Visual Basic Express | Copiar código

```
Dim sum As Integer = 0
Dim counter As Integer = 0
Do While sum < 100
  sum = sum + CInt(Textbox1.Text)
  counter = counter + 1
Loop
```

```
MsgBox("The loop has run " & CStr(counter) & " times!")
```

Presione F5 para ejecutar el programa.

En el cuadro de texto, escriba un número y haga clic en el botón.

Aparece un cuadro de mensaje que muestra el número de veces que se agregó a sí mismo el número antes de llegar a 100.

En el menú **Depurar**, elija **Detener depuración** para finalizar el programa. Mantenga abierto este proyecto. Más adelante se agregarán elementos.

Instrucción Do...Until

La instrucción **Do...While** repite un bucle mientras una condición permanece **True**, pero a veces es posible que desee que el código se repita a sí mismo hasta que una condición se convierta **True**. Puede utilizar la instrucción **Do...Until** del siguiente modo.

Visual Basic Express — Copiar código

```
Dim sum As Integer = 0
Do Until sum >= 100
  sum = sum + 10
Loop
```

Este código es similar al código para la instrucción **Do...While**, sólo que esta vez, el código evalúa la variable `sum` para ver si es igual a o mayor que 100.

Inténtelo

Para utilizar una instrucción Do...Until

Agregue el siguiente código debajo de la línea `MsgBox`.

Visual Basic Express — Copiar código

```
Dim sum2 As Integer = 0
```

```
Dim counter2 As Integer = 0
Do Until sum2 >= 100
  sum2 = sum2 + CInt(Textbox1.Text)
  counter2 = counter2 + 1
Loop
MsgBox("The loop has run " & CStr(counter2) & " times!")
```

Presione F5 para ejecutar el programa.

En el cuadro de texto, escriba un número y haga clic en el botón.

Aparece un segundo cuadro de mensaje que muestra el número de veces que se agregó el número a sí mismo antes de igualar 100 o más.

Hacer que un programa elija entre dos posibilidades: la instrucción If...Then

En esta lección, aprenderá a utilizar la instrucción **If...Then** para ejecutar el código basado en condiciones.

Los programas deben realizar diferentes acciones en respuesta a distintas condiciones. Por ejemplo, quizá desee que el programa compruebe qué día de la semana es y haga algo diferente dependiendo del día. La instrucción **If...Then** permite evaluar una condición y ejecutar las diferentes secciones de código basándose en los resultados de esa condición.

El siguiente ejemplo muestra cómo funciona la instrucción **If...Then**.

Visual Basic Express Copiar código

```
If My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Monday Then
  MsgBox("Today is Monday!")
End If
```

Cuando se ejecuta este código, se evalúa la condición (la parte entre **If** y **Then**). Si la condición es true, se ejecuta la siguiente línea de código y se muestra un cuadro de mensaje; si es false, el código pasa a la línea **End If**. En otras palabras, el código estipula "Si hoy es lunes, muestre el mensaje".

Inténtelo

Para utilizar la instrucción If...Then

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `IfThen` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador de eventos **Form1_Load**, escriba el siguiente código.

Visual Basic Express Copiar código

```
If My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Saturday Or _
My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Sunday Then
  MsgBox("Happy Weekend!")
End If
```

Presione F5 para ejecutar el programa.

Si hoy es sábado o domingo, aparecerá un cuadro de mensaje indicándole `Happy Weekend!`. De lo contrario, no aparecerá ningún cuadro de mensaje.

En el menú **Depurar**, seleccione **Detener depuración** para finalizar el programa. Mantenga abierto este proyecto. Se utilizará en el siguiente procedimiento, "Para utilizar la cláusula **Else**".

Es posible que haya observado en el ejemplo anterior que la instrucción **If...Then** utilizó el operador **Or** para evaluar varias condiciones ("Si es sábado **Or** si es domingo"). Puede utilizar los operadores **Or** y **And** para evaluar tantas condiciones como desee en una instrucción **If...Then** única.

La cláusula Else

Ha visto cómo utilizar la instrucción **If...Then** para ejecutar el código si una condición es true, pero ¿qué pasa si desea ejecutar un código si una condición es true, pero otro si es false? En este caso, puede utilizar la cláusula **Else**. La cláusula **Else** le permite especificar un bloque de códigos que se ejecutará si la condición es false. El siguiente ejemplo muestra cómo funciona la cláusula **Else**.

Visual Basic Express Copiar código

```
If My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Friday Then
  MsgBox("Today is Friday!")
Else
  MsgBox("It isn't Friday yet!")
End If
```

En este ejemplo, se evalúa la expresión; si es true, se ejecuta la siguiente línea de código y se muestra el primer cuadro de mensaje. Si es false, el código se desplaza a la cláusula **Else** y se ejecuta la línea **Else** siguiente, que muestra el segundo cuadro de mensaje.

Inténtelo

Para utilizar la cláusula Else

Cambie el código en la instrucción **If...Then** de la siguiente forma.

Visual Basic Express — Copiar código

```
If My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Saturday Or _
My.Computer.Clock.LocalTime.DayOfWeek = DayOfWeek.Sunday Then
  MsgBox("Happy Weekend!")
Else
  MsgBox("Happy Weekday! Don't work too hard!")
End If
```

Presione F5 para ejecutar el programa. El programa mostrará ahora un cuadro de mensaje que indica si es un fin de semana o un día de la semana, con contenido adecuado.

Nota

Para cambiar el día de la semana haga doble clic en la hora en la barra de tareas de Windows, si desea probar la ejecución de los dos bloques de código. (La barra de tareas es la que contiene el botón Inicio de Windows; de manera predeterminada, se encuentra en la parte inferior del escritorio y la hora se muestra en la esquina derecha).

Información detallada: utilizar Select Case para decidir entre varias opciones

En esta lección, aprenderá a utilizar la instrucción **Select Case** para ejecutar código basado en múltiples condiciones.

En la lección anterior, aprendió a utilizar las instrucciones **If...Then** para ejecutar diferentes bloques de código según las condiciones. Aunque es posible evaluar más de dos condiciones en una instrucción **If...Then** mediante la palabra clave **ElseIf**, la instrucción **Select Case** proporciona una manera mucho mejor de evaluar varias condiciones.

La instrucción **Select Case** permite utilizar tantas condiciones (o casos) como sea necesario, y conviene escribir el código para situaciones en las que hay muchas opciones. Por ejemplo, suponga que el programa utilizó una variable **String** para almacenar una opción de color y se necesitaba obtener el valor de color. El código para la instrucción **Select Case** podría ser similar al siguiente:

Visual Basic Express Copiar código

```
Select Case Color
  Case "red"
     MsgBox("You selected red")
  Case "blue"
     MsgBox("You selected blue")
  Case "green"
     MsgBox("You selected green")
End Select
```

Cuando se ejecuta este código, la línea **Select Case** determina el valor (`Color`) de la expresión. Suponga que `Color` es una variable **String** y que esta variable es un parámetro para un método que contiene la instrucción **Select Case**. El valor de `Color` se compara con el valor para la primera instrucción **Case**. Si el valor coincide, se ejecuta la siguiente línea de código y el código pasa a la línea **End Select**; si el valor no coincide, se evalúa la siguiente línea **Case**.

La instrucción **Case** adopta muchas formas distintas; en el ejemplo anterior es **String**. Pero puede ser cualquier tipo de datos o expresión.

Puede evaluar un intervalo de números utilizando la palabra clave **To**, como sigue:

Visual Basic Express Copiar código

```
Case 1 To 10
```

En este ejemplo, cualquier número entre 1 y 10 será una coincidencia.

También puede evaluar varios valores en una sola instrucción **Case** separándolos con comas de la siguiente forma:

Visual Basic Express | Copiar código

```
Case "red", "white", "blue"
```

En este ejemplo, cualquiera de los tres valores producirá una coincidencia.

También puede utilizar operadores de comparación y la palabra clave **Is** para evaluar los valores de la siguiente manera.

Visual Basic Express | Copiar código

```
Case Is > 9
```

En este ejemplo, cualquier número mayor que 9 provocará una coincidencia.

Case Else

El ejemplo anterior funciona cuando conoce todas las condiciones posibles, pero ¿qué sucede si hay una condición con la que no contaba? Por ejemplo, si el valor de `Color` es `yellow`, el código simplemente evaluará los tres casos sin encontrar una coincidencia y no se mostrará ningún cuadro de mensaje.

La instrucción **Case Else** se puede utilizar para ejecutar el código cuando no se encuentra ninguna coincidencia, como en el siguiente ejemplo.

Visual Basic Express | Copiar código

```
Select Case Color
  Case "red"
     MsgBox("You selected red")
  Case "blue"
     MsgBox("You selected blue")
  Case "green"
     MsgBox("You selected green")
  Case Else
     MsgBox("Please choose red, blue, or green")
End Select
```

En el código anterior, si el valor de `Color` es `yellow` el código lo comparará con las primeras tres líneas **Case** sin encontrar una coincidencia. Cuando se llega a la línea **Case Else**, se ejecuta la siguiente línea de código antes de pasar a **End Select**.

Para utilizar la instrucción Select Case

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `SelectCase` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **TextBox** y un control **Button** al formulario.

Haga doble clic en el botón para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, escriba el siguiente código.

Visual Basic Express — Copiar código

```
Dim Number As Integer = CInt(Textbox1.Text)
Select Case Number
  Case 1
     MsgBox("Less than 2")
  Case 2 To 5
     MsgBox("Between 2 and 5")
  Case 6, 7, 8
     MsgBox("Between 6 and 8")
  Case 9 To 10
     MsgBox("Greater than 8")
  Case Else
     MsgBox("Not between 1 and 10")
End Select
```

Presione F5 para ejecutar el programa.

En el cuadro de texto, escriba un número y haga clic en el botón.

Aparecerá un cuadro de mensaje que muestra el mensaje de la instrucción **Case** que coincide con el número que ha especificado

Qué hacer cuando algo sale mal: control de errores

En esta lección, aprenderá a crear código de control de errores básico para los programas.

Incluso los programas mejor diseñados a veces encuentran errores. Algunos errores son defectos en el código que se pueden encontrar y corregir. Otros errores son una consecuencia natural del programa; por ejemplo, el programa puede intentar abrir un archivo que ya está en uso. En casos así, los errores se pueden predecir, pero no evitar. Como desarrollador, es su trabajo predecir estos errores y ayudar a que el programa los solucione.

Errores en tiempo de ejecución

Un error que se produce mientras un programa se está ejecutando se llama *error en tiempo de ejecución*. Los errores en tiempo de ejecución se producen cuando un programa trata de hacer algo para lo cual no fue diseñado. Por ejemplo, si el programa intenta realizar una operación no válida, como convertir una cadena no numérica en un valor numérico, se producirá un error en tiempo de ejecución.

Cuando se produce un error en tiempo de ejecución, el programa produce una *excepción*, que soluciona los errores buscando código dentro del programa para tratar el error. Si no se encuentra tal código, se detiene el programa y se tiene que reiniciar. Dado que esto puede conducir a la pérdida de datos, es prudente crear el código de control de errores dondequiera que se tenga previsto que se produzcan errores.

El bloque Try...Catch...Finally.

Se puede utilizar el bloque **Try...Catch...Finally** para controlar errores en tiempo de ejecución en el código. Puede utilizar **Try** para un segmento de código; si ese código produce una excepción, salta al bloque **Catch** y se ejecuta el código del bloque **Catch**. Después de que ese código ha finalizado, se ejecuta cualquier código en el bloque **Finally**. La instrucción **End Try** cierra el bloque **Try...Catch...Finally** completo. En el ejemplo siguiente se ilustra cómo se utiliza cada bloque.

Visual Basic Express | Copiar código

```
Try
  ' Code here attempts to do something.
```

```
Catch
  ' If an error occurs, code here will be run.
Finally
  ' Code in this block will always be run.
End Try
```

Primero, se ejecuta el código del bloque **Try**. Si se ejecuta sin error, el programa omite el bloque **Catch** y ejecuta el código del bloque **Finally**. Si se produce un error en el bloque **Try**, la ejecución salta inmediatamente al bloque **Catch** y se ejecuta el código que se encuentra allí; luego se ejecuta el código del bloque **Finally**.

Inténtelo

Para utilizar el bloque Try...Catch

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `TryCatch` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **TextBox** y un control **Button** al formulario.

Haga doble clic en **Button** para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, escriba el siguiente código:

Visual Basic Express — Copiar código

```
Try
  Dim aNumber As Double = CDbl(Textbox1.Text)
  MsgBox("You entered the number " & aNumber)
Catch
  MsgBox("Please enter a number.")
Finally
  MsgBox("Why not try it again?")
End Try
```

Presione F5 para ejecutar el programa.

En el cuadro de texto, escriba un valor numérico y haga clic en el botón. Aparece un cuadro de mensaje que muestra el número que ha escrito, seguido por una invitación para volver a intentarlo.

A continuación, escriba un valor no numérico en el cuadro de texto, como una palabra y haga clic en el botón. Esta vez, cuando el programa intente convertir el texto del cuadro de texto en un número, no podrá hacerlo y se producirá un error. En lugar de finalizar el código en el bloque **Try**, se ejecuta el bloque **Catch** y aparece un cuadro de mensaje solicitando que se escriba un número. Se ejecuta el bloque **Finally** y se le invita a intentarlo de nuevo.

Crear la apariencia visual de un programa: introducción a los formularios Windows Forms

La interfaz de usuario es la parte del programa que ven los usuarios cuando ejecutan el programa. Una interfaz de usuario suele estar formada por una ventana o formulario principal y varios controles, como botones, campos para la introducción de texto, etc. Los programas de Visual Basic Expressque se ejecutan en el equipo se denominan *Aplicaciones de Windows Forms* y la interfaz de usuario se crea mediante los controles de formularios Windows Forms.

Las lecciones de esta sección le mostrarán cómo crear una interfaz de usuario utilizando algunos de los controles de formularios Windows Forms más comunes.

Comunicarse con el usuario del programa: interfaz de usuario

En esta lección, aprenderá lo que es una *interfaz de usuario* (UI), qué son los controles y cómo agregar controles a una interfaz de usuario.

En los primeros días de los equipos personales, los usuarios interactuaban con programas principalmente a través de una línea de comandos. Se iniciaba un programa y después se hacía una pausa para recibir los datos proporcionados por el usuario. La mayoría de los programas utilizados hoy, sin embargo, se ejecutan en una o varias ventanas que permiten que el usuario se comunique, o *relacione*, con el programa escribiendo, haciendo clic en los botones, eligiendo elementos en los menús preestablecidos y así sucesivamente. En estas lecciones y las subsiguientes, aprenderá a generar interfaces de usuarios propias basadas en Windows.

-Utilizar formularios

Los formularios son las unidades de creación básicas para la interfaz de usuario. Cada formulario del programa representa una ventana que se aparece a los usuarios. Al trabajar en el IDE (entorno de desarrollo integrado) de Visual Basic, un formulario es el *diseñador* que se utiliza para diseñar la interfaz de usuario, lo que sería similar a utilizar Windows Paint para dibujar una imagen.

Los controles se utilizan en el diseñador para crear la apariencia de la interfaz de usuario. Un control es un objeto que tiene un aspecto y comportamiento predefinidos. Por ejemplo, un control Button tiene el aspecto y el comportamiento de un botón de comando: cuando un usuario hace clic en él, cambia para mostrarlo.

Cada control de Visual Basic Expresstiene una finalidad. Por ejemplo, los controles TextBox se utilizan para introducir texto, mientras que los controles PictureBox se utilizan para mostrar imágenes. Hay más de cincuenta controles diferentes incluidos en Visual Basic; también se pueden crear controles propios conocidos como *controles de usuario*. Obtendrá más información sobre cada tipo de control en lecciones posteriores.

Al diseñar la interfaz de usuario, se arrastran los controles desde el **Cuadro de herramientas**, se colocan en un formulario, luego se ubican y se cambian de tamaño para crear el aspecto deseado. Puede cambiar el aspecto aún más estableciendo propiedades de formularios y controles en la ventana **Propiedades**. Por ejemplo, los formularios y la mayoría de los controles tienen una propiedad **BackColor** que se utiliza para establecer su color de fondo.

Las propiedades también se utilizan para definir el comportamiento de un formulario o control. Por ejemplo, la propiedad ShowInTaskbar de un formulario determina si el formulario aparecerá en la **barra de tareas** de Windows cuando se esté ejecutando el programa. Mediante el uso de propiedades, puede personalizar la apariencia y el comportamiento de la interfaz de usuario.

Inténtelo

Para cambiar las propiedades de un formulario

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `FirstForm` y haga clic en **Aceptar**.

Se crea un nuevo proyecto de formularios Windows Forms. Aparece un nuevo formulario en la ventana principal y sus propiedades son visibles en la ventana **Propiedades**, en la esquina inferior derecha del IDE de Visual Basic.

Haga clic en el formulario una vez para seleccionarlo.

En la ventana **Propiedades**, cambie la propiedad Text para que se lea "Mi primer formulario" y presione ENTRAR.

El texto en la parte superior del formulario cambia después de especificar el nuevo valor.

En la ventana **Propiedades**, cambie la propiedad BackColor a un color diferente seleccionando un color desde la lista desplegable.

Observe que la propiedad **BackColor** se cambia a través de una interfaz especial. Esta interfaz le permite ver el color antes de seleccionarlo y le permite elegir entre los colores utilizados actualmente por el sistema, colores estándar Web o una selección de colores más personalizada. También puede escribir sólo el nombre del color (por ejemplo, `Red`) en el cuadro en la ventana **Propiedades**.

Experimente cambiando otras propiedades del formulario. Cuando esté listo, continúe con el siguiente procedimiento.

Agregar controles al formulario

En este procedimiento, agregará los controles al formulario seleccionando el control en la ventana **Cuadro de herramientas**, que se encuentra normalmente en el lado izquierdo del IDE de Visual Basic, y arrastrándolo al formulario. Se manipularán las propiedades de los controles.

Para agregar controles al formulario

Desde el **Cuadro de herramientas**, arrastre un control **Button**, un control **TextBox**, un control Label y finalmente un control CheckBox hasta el formulario.

Seleccione el control **Button** y arrástrelo alrededor del formulario para cambiar su ubicación.

Observe cómo aparecen las instrucciones cuando lo arrastra cerca de los otros controles. Estas instrucciones pueden ayudarle a colocar los controles en forma precisa.

Repita el proceso con los otros controles hasta que la interfaz de usuario tenga el aspecto que desea.

Seleccione el control **Button**, luego haga clic y arrastre el cuadrado blanco de la esquina inferior derecha para cambiar el tamaño.

Pruebe las propiedades de control durante algunos minutos. Haga clic en cada control en el formulario para seleccionarlo y cambie algunas de sus propiedades en la ventana **Propiedades**. Entre las propiedades que puede tratar de cambiar se encuentran: **Font**, **BackColor**, **ForeColor** y **Text**.

Presione F5 para ejecutar el programa. Aparecerá una ventana con los controles que acaba de agregar. Observe que puede hacer clic en el botón, activar y desactivar la casilla de verificación y escribir en el cuadro de texto

Interactuar con el usuario: utilizar botones

En esta lección, obtendrá información sobre cómo agregar un control Button a un formulario, cómo cambiar el aspecto del botón y cómo escribir código que se ejecute cuando se haga clic en él.

La manera más fácil para los usuarios de interactuar con el programa es mediante botones. Por ejemplo, muchos programas tienen botones **Salir**. Como se vio en la lección anterior, el control **Button** de Visual Basic Expressparece y se comporta como botón de comando. El control **Button** también tiene eventos predefinidos que se pueden utilizar para iniciar acciones tales como finalizar un programa.

Utilizar los botones

En general, los botones son controles rectangulares que tienen una apariencia elevada en el formulario. Sin embargo, hay muchas propiedades que se pueden establecer para cambiar su apariencia. La más obvia es la propiedad Text, que determina el texto mostrado y este texto se muestra en la *fuente* o el tipo de letra determinado por la propiedad Font. La propiedad BackColor determina el color del botón y la propiedad ForeColor determina el color del texto.

Cuando el usuario hace clic en un botón en tiempo de ejecución, el control **Button** provoca el evento Click. Cuando aparece un evento, los controles ejecutan el código como respuesta a esos eventos. Puede escribir código que se ejecute creando un *event handler*.

Un controlador de eventos es un método que se ejecuta cuando ocurre un evento. Cuando el usuario hace clic en un botón, el evento **Click** del botón tiene un controlador de eventos. Es más fácil de lo que parece y, en el ejemplo siguiente, obtendrá información sobre cómo escribir un controlador de eventos. Los eventos y los controladores de eventos se tratarán con más detalle en Hacer que el programa reaccione ante el usuario: crear un controlador de eventos.

¡Inténtelo!

Para utilizar los botones

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `ButtonExample` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **Button** hasta el formulario.

En la ventana **Propiedades**, cambie la propiedad **Text** para que se lea: `What time is it?` y, a continuación, presione **ENTRAR**.

Observe que el texto no se ajusta en el botón.

En la ventana **Propiedades**, seleccione la propiedad AutoSize y establezca su valor en `True`.

El botón cambia de tamaño para ajustar el texto.

En el formulario, haga doble clic en el botón para abrir el Editor de código.

Se abre el Editor de código en medio de un método denominado **Button1_Click**. Éste es el controlador de eventos **Button1.Click**. El código que escribe aquí se ejecutará cuando se haga clic en el botón.

En el controlador del evento **Button1_Click**, escriba la siguiente línea de código.

Visual Basic Express Copiar código

```
MsgBox("The current time is " & Now.ToShortTimeString)
```

Presione F5 para ejecutar el programa.

El programa comienza y aparece el formulario. Cuando hace clic en **Button**, aparece un cuadro de mensaje que muestra la hora actual

Mostrar y recibir texto: utilizar etiquetas y cuadros de texto

En este tema, aprenderá a utilizar los controles Label y TextBox para mostrar texto y aceptar la entrada de texto del usuario.

Una de las maneras más fáciles de transmitir y recibir la información de los usuarios es a través de texto. Puede mostrar texto sobre la funcionalidad de un programa y también recibir datos como texto del usuario y utilizarlos en el programa. Visual Basic Expressproporciona dos controles diseñados para mostrar y recibir el texto. Son los controles **Label** y **TextBox**.

Mostrar texto con el control Label

El control **Label** es el control primario para mostrar texto. Éste aparece en el formulario como texto delimitado por un área de forma rectangular. Generalmente, el color de esta área es igual que el color del formulario, por lo que aparece como si fuera texto del formulario.

Dado que el control **Label** tiene como objetivo principal mostrar texto, las propiedades más importantes para un control **Label** son las propiedades que controlan su aspecto. La propiedad Text contiene el texto que se muestra en el control **Label**. La propiedad Font determina la fuente con la que se mostrará el texto en la propiedad **Text**. La propiedad ForeColor determina el color del texto en sí y la propiedad BackColor determina el color del área que rodea el texto.

Recibir texto con el control TextBox

Cuando se necesita mostrar y recibir texto, se diseña el control **TextBox** para controlar el trabajo. Además de mostrar el texto, el control **TextBox** permite a los usuarios escribir texto en el control **TextBox** en tiempo de ejecución, y el programa puede recuperar ese texto.

Al igual que con el control **Label**, las propiedades que son más importantes para el control **TextBox** son aquéllas relacionadas con la apariencia. Una propiedad importante es la propiedad **Text**, que representa el texto del control **TextBox**. Cuando un usuario escribe en el control **TextBox**, la propiedad **Text** se actualiza para reflejar los cambios. De este modo, el texto que se muestra en el control **TextBox** siempre refleja el valor de la propiedad **Text**.

También hay propiedades que afectan al comportamiento del control **TextBox**. La propiedad Multiline determina si el control **TextBox** permite varias líneas. Si esta propiedad se establece en **False**, el control **TextBox** siempre tendrá exactamente una línea de alto y no se podrá ampliar verticalmente. Si se establece en **True**, el control **TextBox** permite varias líneas y puede tener el alto deseado.

Inténtelo

Para crear una interfaz de usuario con los controles Label y Textbox

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `TextBoxExample` y, a continuación, haga clic en Aceptar.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **TextBox**, **Label** y Button hasta el formulario.

Seleccione el control **Label** y arrástrelo sobre el control **TextBox**.

En la ventana **Propiedades**, cambie la propiedad Text del control **Label** al siguiente código.

```
Enter your name and click the button.
```

Ahora que ha creado una interfaz de usuario básica, deberá agregar algo de código al programa y estará listo para probarlo.

Para agregar el código y probar el programa

Haga doble clic en el control **Button** para abrir el Editor de código.

El Editor de código se abre en el controlador de eventos **Button1_Click**.

Agregue la siguiente línea de código al controlador de eventos **Button1_Click**.

Visual Basic Express | Copiar código

```
MsgBox("Your Name is " & Textbox1.Text)
```

Presione F5 para ejecutar el programa.

Cuando aparezca el formulario, escriba su nombre en el control **TextBox** y haga clic en el botón. Aparece un cuadro de mensaje que muestra el texto del control **TextBox**. Cambie el texto y haga clic en el botón nuevamente. Cada vez que haga clic en el botón, se mostrará el texto actualizado.

Hacer que el programa reaccione ante el usuario: crear un controlador de eventos

En esta lección, aprenderá a crear un controlador de eventos.

Como se ha visto en lecciones anteriores, los controles tienen propiedades, métodos y eventos y se utilizan para crear la interfaz de usuario. Los eventos son situaciones especiales que le pueden suceder a un control. Por ejemplo, se puede hacer clic en un control, se puede escribir texto en él, el puntero del mouse se puede mover sobre el control y así sucesivamente.

Cuando se produce algo interesante, el control *provoca un evento*; es decir, envía una señal al programa para hacerle saber que ha sucedido algo. El programa comprueba si tiene algún método para controlar dicho evento. Tales métodos se denominan *controladores de eventos*. Un ejemplo es un método que se ejecuta cuando se hace clic en un botón, como el método que se creó en Interactuar con el usuario: utilizar botones.

Puede crear controladores de eventos para una variedad de eventos de control. En esta lección, creará controladores de eventos para controlar los eventos MouseEnter y MouseLeave de un botón: los eventos que se provocan cuando se mueve un mouse sobre un control.

¡Inténtelo!

Para controlar el evento MouseEnter

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `EventHandler` y, a continuación, haga clic en Aceptar.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control Button hasta el formulario.

En las ventanas **Propiedades**, establezca la propiedad AutoSize en **True**.

En el menú **Ver**, elija **Código** para abrir el Editor de código.

Justo sobre el Editor de código, observe los dos cuadros desplegables. El cuadro de la izquierda contiene una lista de todos los controles que aparecen en el formulario, al igual que **Form1**, (**General**) y (**Eventos de Form1**). El cuadro de la derecha muestra cada uno de los eventos disponibles para el elemento que se muestra en el cuadro de la izquierda.

En el cuadro de la izquierda, elija **Button1**.

En el cuadro de la derecha, elija **MouseEnter**.

Aparece un nuevo controlador de eventos denominado **Button1_MouseEnter** en el Editor de código.

En el controlador de eventos **Button1_MouseEnter**, escriba el siguiente código.

Visual Basic Express — Copiar código

```
Button1.Text = "The Mouse has entered"
```

Presione F5 para ejecutar la aplicación. Pase el puntero del mouse sobre el botón. Observe que cuando el puntero del mouse pasa sobre **Button1**, el texto del botón cambia.

Agregar otro controlador de eventos

Quizá haya notado en el ejemplo anterior que aunque el texto de **Button1** cambia cuando el puntero del mouse pasa sobre él, cuando éste se quita, el texto no vuelve a cambiar. Si desea que el texto cambie cuando el mouse ya no está sobre el botón, debe controlar el evento **MouseLeave** además del evento **MouseEnter**.

Para controlar el evento MouseLeave

En el Editor de código, asegúrese de que está seleccionado **Button1** en la lista desplegable de la izquierda y seleccione **MouseLeave** del cuadro desplegable de la derecha.

Aparece un nuevo controlador de eventos denominado **Button1_MouseLeave** en el Editor de código.

En el controlador de eventos **Button1_MouseLeave** , escriba el siguiente código.

Visual Basic Express | Copiar código

```
Button1.Text = "The mouse has left"
```

Presione F5 para ejecutar la aplicación.

Ahora cuando el puntero del mouse pasa sobre el botón, el texto cambia a `The mouse has entered`, pero cuando el mouse ya no está sobre el botón, el texto cambia a `The mouse has left`.

Información detallada: compartir un controlador de eventos

En esta lección, aprenderá a crear un controlador de eventos compartido que controla eventos para más de un control.

En la lección anterior, Hacer que el programa reaccione ante el usuario: crear un controlador de eventos, aprendió a escribir un código en respuesta a los eventos MouseEnter y MouseLeave para un control Button. Sin embargo, ¿qué pasa si tiene dos o más controles **Button** y desea mostrar el mismo mensaje para todos ellos? Se puede escribir el código en los controladores de eventos para cada control, pero afortunadamente, hay una manera más fácil.

Si examina atentamente los métodos controladores de eventos para el evento **MouseEnter**, observará que la declaración **Method** (`Private Sub Button1_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.MouseEnter`) contiene una cláusula **Handles** (`Handles Button1.MouseEnter`). Como es lógico, la

palabra clave **Handles** le dice al controlador de eventos cuáles eventos debe controlar.

Para compartir un controlador de eventos entre varios controles, simplemente debe agregar los nombres de los controles adicionales y el nombre del evento que desea controlar. Luego, el controlador de eventos recibe una notificación cuando se produce el evento para cualquiera de dichos controles. Por ejemplo, si tiene dos controles **Button** y desea utilizar el mismo controlador de eventos para ambos, la cláusula **Handles** puede tener el siguiente aspecto.

```
Handles Button1.MouseEnter, Button2.MouseEnter.
```

Ahora tiene un método único que controla el evento **MouseEnter** para ambos controles, pero ¿cómo sabe el controlador de eventos cuál control provocó el evento? Si examina nuevamente la declaración **Method**, observará la cláusula `ByVal sender As Object`; la palabra clave **Sender** le dice al controlador de eventos cuál objeto (en este caso cuál control) provocó el evento.

Inténtelo

Para compartir un controlador de eventos

Abra el proyecto EventHandler que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Hacer que el programa reaccione ante el usuario: crear un controlador de eventos y finalizar los procedimientos de dicha lección.

En el **Explorador de soluciones**, seleccione **Form1.vb** y, a continuación, en el menú **Ver** elija **Diseñador**.

En el **Cuadro de herramientas**, arrastre otro control **Button** hasta el formulario.

En la ventana **Propiedades**, establezca la propiedad AutoSize en **True**.

En el menú **Ver**, elija **Código** para abrir el Editor de código.

En la declaración de método **Button1_MouseEnter** (`Private Sub Button1_MouseEnter(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.MouseEnter`), cambie la cláusula **Handles** para que se lea `Handles Button1.MouseEnter, Button2.MouseEnter`.

En el cuerpo de la declaración de evento, reemplace el código con lo siguiente.

Visual Basic Express — Copiar código

```
If sender.Equals(Button1) Then
```

```
  Button1.Text = "The mouse has entered Button1"
Else
  Button2.Text = "The mouse has entered Button2"
End If
```

Este código comprueba si el remitente era **Button1** , si es así, se actualiza la propiedad **Text** de **Button1**, si no lo es, se actualiza la propiedad **Text** de **Button2**.

En la declaración de método **Button1_MouseLeave**, cambie la cláusula **Handles** para que se lea de la siguiente manera.

```
Handles Button1.MouseLeave, Button2.MouseLeave.
```

En el cuerpo de la declaración de evento, reemplace el código con lo siguiente.

Visual Basic Express Copiar código

```
sender.Text = "The mouse has left"
```

En este caso, el código establece la propiedad **Text** del remitente (**Button1** o **Button2**) en la misma cadena.

Presione F5 para ejecutar la aplicación.

Ahora, cuando el puntero del mouse (ratón) pasa sobre el botón, el texto cambia a `The mouse has entered` junto con el nombre del botón, y cuando el mouse ya no está sobre el botón, el texto vuelve a ser `The mouse has left`.

Intente agregar más controles al formulario y modificar las cláusulas **Handles** para incluirlos, ni siquiera deben ser del mismo tipo.

Obtener opciones seleccionadas por el usuario: utilizar casillas de verificación y botones de opción

En esta lección, aprenderá a utilizar casillas de verificación y botones de opción para presentar y recuperar las elecciones del usuario.

Cuando se crea la interfaz de usuario para el programa, a menudo se necesita un modo de presentar las elecciones. Por ejemplo, suponga que escribió una aplicación para tomar las órdenes para una pizzería; deseará que los usuarios puedan seleccionar cualquiera o todas las variedades de ingredientes para

cubrir la pizza. El control CheckBox proporciona una representación visual que hace que esta opción sea fácil de crear.

El control **CheckBox** se compone de una etiqueta de texto y un cuadro que el usuario puede seleccionar. Cuando el usuario hace clic en el cuadro, aparece una marca de verificación en él. Si se vuelve a hacer clic en el cuadro, la marca de verificación desaparece. El estado de la casilla de verificación se puede recuperar utilizando la propiedad **CheckBox.Checked**. Si el cuadro muestra una marca de verificación, la propiedad devuelve **True**. Si no se muestra ninguna comprobación, la propiedad devuelve **False**.

Inténtelo

Para utilizar casillas de verificación

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `UserChoices` y, a continuación, haga clic en Aceptar.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control Button y tres controles **CheckBox** hasta el formulario.

En la ventana **Propiedades**, cambie la propiedad Text de **CheckBox1**, **CheckBox2** y **CheckBox3** para que diga `Pepperoni`, `Sausage` y `Mushrooms` respectivamente.

En la ventana **Propiedades**, cambie la propiedad **Text** de **Button1** para que diga `Order Pizza`.

En el formulario, haga doble clic en el botón; se abrirá el controlador de eventos **Button1_Click** en el Editor de código.

Agregue el código siguiente al controlador de eventos **Button1_Click**:

Visual Basic Express — Copiar código

```
Dim toppings As String = ""
If CheckBox1.Checked = True Then
    toppings &= "Pepperoni "
End If
```

```
If CheckBox2.Checked = True Then
    toppings &= "Sausage "
End If
If CheckBox3.Checked = True Then
    toppings &= "Mushrooms"
End If
If toppings <> "" Then
    MsgBox("Your pizza has the following toppings: " & toppings)
End If
```

Presione F5 para ejecutar el programa. Cuando aparezca el formulario, seleccione algunos ingredientes y haga clic en el botón. Se muestra un cuadro de mensaje que indica su elección de ingredientes para la pizza.

Utilizar botones de opción para realizar elecciones exclusivas

Acaba de aprender a permitir que un usuario elija alguna o todas las diversas opciones. Pero ¿qué pasa si desea que el usuario elija sólo una de varias opciones? En este caso, puede utilizar el control RadioButton.

A diferencia de las casillas de verificación, los botones de opción siempre funcionan como parte de un grupo. Al seleccionar un botón de opción inmediatamente se borran todos los otros botones de opción en el grupo. Al definir un grupo de botones de opción, se indica al usuario que "tiene este conjunto de opciones entre las que puede elegir una y solamente una".

Puede utilizar grupos de controles **RadioButton** para permitir a los usuarios elegir entre las opciones exclusivas. Por ejemplo, puede permitir que un usuario elija salsa normal o salsa picante en la pizza, pero no ambas. Como un control **CheckBox**, puede recibir información sobre el estado del control **RadioButton** de la propiedad **RadioButton.Checked**.

Para utilizar botones de opción

En el **Cuadro de herramientas**, arrastre dos controles **RadioButton** al formulario.

En la ventana **Propiedades**, establezca la propiedad **Text** para **RadioButton1** en `Sauce Regular`.

Establezca la propiedad Checked para **RadioButton1** en **True**.

Sugerencia

Al definir un grupo de elecciones, siempre debe establecer una elección para que sea el valor predeterminado.

En la ventana **Propiedades**, establezca la propiedad **Text** para **RadioButton2** en `Spicy Sauce`.

En el formulario, haga doble clic en el botón para abrir el controlador de eventos **Button1_Click** en el Editor de código.

En el controlador de eventos **Button1_Click**, agregue el siguiente código:

Visual Basic Express — Copiar código

```
If RadioButton1.Checked = True Then
    MsgBox("You chose regular sauce")
Else
    MsgBox("You chose spicy sauce")
End If
```

Presione F5 para ejecutar el programa. Elija uno de los botones de opción y, a continuación, haga clic en el botón **Order Pizza**. Se mostrará un cuadro de mensaje que tiene en cuenta su elección.

Intente seleccionar ambos botones de opción al mismo tiempo. Observe que los botones de opción son excluyentes. Después de hacer clic en uno, el otro se borra automáticamente.

Información detallada: utilizar varios grupos de botones de opción

En esta lección, aprenderá a crear varios grupos de botones de opción mutuamente exclusivos en un formulario único.

En la lección anterior aprendió a crear un grupo de botones de opción con el fin de presentar un conjunto de opciones mutuamente excluyentes. ¿Qué sucede, sin embargo, si necesita presentar dos o más conjuntos diferentes de opciones? Verá que todos los controles RadioButton de un formulario se tratan como un grupo único, lo que permite seleccionar un solo botón de opción.

Afortunadamente, Visual Basic Expresstiene varios controles conocidos como *controles contenedores* que pueden contener otros controles. Colocando un control contenedor en el formulario y colocando después controles **RadioButton** dentro del control contenedor, puede tener varios grupos de botones de opción en el mismo formulario.

Los controles contenedores más comunes son el control GroupBox y el control Panel. La diferencia principal entre ambos es que el control **GroupBox** tiene un borde visible a su alrededor y el control **Panel** no lo tiene. Cuando se utiliza un control contenedor para agrupar botones de opción, el control **GroupBox** es la mejor elección porque el borde proporciona una indicación visual de que las opciones del grupo están relacionadas.

¡Inténtelo!

Para utilizar un control GroupBox como un contenedor

Abra el proyecto UserChoices que creó en la lección anterior. Si no lo guardó, necesitará regresar primero a la lección anterior, Obtener opciones seleccionadas por el usuario: utilizar casillas de verificación y botones de opción, y finalizar los procedimientos.

En el **Explorador de soluciones**, seleccione **Form1.vb** y, a continuación, en el menú **Ver** elija **Diseñador**.

En el **Cuadro de herramientas**, arrastre un control **GroupBox** hasta el formulario.

En la ventana **Propiedades**, cambie la propiedad Text del control **GroupBox** para que se lea `Select a crust`.

Con el control **GroupBox** seleccionado, arrastre dos controles **RadioButton** del **Cuadro de herramientas** y colóquelos sobre el control **GroupBox**.

En la ventana **Propiedades**, cambie las propiedades Text de **RadioButton3** y **RadioButton4** a `Thin crust` y `Thick crust`, respectivamente.

En el formulario, haga doble clic en el botón **Pedir pizza** para abrir el controlador del evento **Button1_Click** del Editor de código.

En el controlador del evento **Button1_Click**, agregue el siguiente código:

Visual Basic Express — Copiar código

```
If RadioButton3.Checked = True Then
    MsgBox("You chose a thin crust")
Else
    MsgBox("You chose a thick crust")
End If
```

Presione F5 para ejecutar el programa. Elija uno de los botones de opción y, a continuación, haga clic en el botón **Pedir pizza**. Se muestra un cuadro de

mensaje que tiene en cuenta su elección. Observe que se conserva su selección de salsa.

Cómo ilustrar: mostrar imagines

En esta lección aprenderá a utilizar un control PictureBox para mostrar imágenes y a mostrar una imagen como imagen de fondo en un formulario.

Se dice que una imagen vale más que mil palabras y, de hecho, muchos programas las utilizan para transmitir información. Hay varias maneras de mostrar imágenes en Visual Basic: la más común es utilizando un control **PictureBox**.

Los controles **PictureBox** actúan como un contenedor para las imágenes; se elige la imagen que se va a mostrar estableciendo la propiedad Image. La propiedad **Image** se puede establecer en la ventana **Propiedades** o se puede escribir el código para decirle al programa cuál imagen se va mostrar.

Otras propiedades útiles para el control **PictureBox** son la propiedad AutoSize, que determina si **PictureBox** se expandirá para ajustar la imagen, y la propiedad SizeMode, que se puede utilizar para expandir, centrar o ampliar la imagen dentro del control **PictureBox**.

Antes de agregar una imagen a un control **PictureBox**, generalmente se agregará el archivo de imagen al proyecto como un *recurso*. Una vez que se agrega un recurso al proyecto, puede volver a utilizarlo cuantas veces lo desee: por ejemplo, se puede mostrar la misma imagen en varios lugares.

Inténtelo

Para agregar una imagen como un recurso

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Pictures` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En la ventana **Explorador de soluciones**, haga doble clic en el nodo **My Project** para abrir el **Diseñador de proyectos**.

En el **Diseñador de proyectos**, haga clic en la ficha **Recursos**.

Haga clic en **Agregar recurso** y, a continuación, elija **Agregar archivo existente** de la lista desplegable.

Se abrirá el cuadro de diálogo **Agregar archivo existente a los recursos**. Si no ve ningún archivo de imagen, vaya a una carpeta que sí contiene imágenes.

Seleccione un archivo de imagen (con una extensión de archivo .bmp, .gif o .jpg) y haga clic en **Abrir**. Para este ejemplo, es mejor elegir una imagen pequeña.

La imagen se agregará al proyecto y aparecerá en la ventana **Administrador de recursos**.

Repita los dos pasos anteriores para agregar una segunda imagen al proyecto.

En el menú **Archivo**, elija **Cerrar**. Si se le solicita guardar los cambios, elija **Sí**.

Para mostrar imágenes utilizando un control PictureBox

En el Explorador de soluciones, seleccione Form1.vb y en el menú Ver elija Diseñador.

En el **Cuadro de herramientas**, arrastre un control **PictureBox** hasta el formulario.

En la ventana **Propiedades**, haga clic en el botón **...** ubicado junto a la propiedad **Image** para abrir el cuadro de diálogo **Seleccionar recurso**.

En la lista **Entrada**, elija una de las imágenes que agregó y haga clic en **Aceptar**.

Seleccione la propiedad **SizeMode** y establézcala en **AutoSize**.

Observe cómo el control **PictureBox** cambia automáticamente de tamaño para ajustar la imagen.

En el formulario, haga doble clic en el control **PictureBox** para abrir el controlador de eventos **PictureBox1_Click** en el Editor de código.

Agregue el código siguiente al controlador de eventos **PictureBox1_Click**.

Nota

Deberá reemplazar "MyPictureName2" con el nombre real de la segunda imagen que agregó anteriormente.

Visual Basic Express Copiar código

```
PictureBox1.Image = My.Resources.MyPictureName2
```

Presione F5 para ejecutar el programa. Cuando aparece el formulario, haga clic en la imagen para que aparezca la segunda imagen.

Mostrar una imagen de fondo en un formulario

Además de mostrar una imagen en un control **PictureBox**, también puede mostrar una imagen como el fondo para el formulario. La propiedad BackgroundImage de un formulario se utiliza para mostrar una imagen que aparecerá detrás de cualquier otro control en el formulario, casi igual que un papel tapiz en el escritorio de Windows.

Así como Windows permite elegir si el papel tapiz está centrado, en mosaico o expandido para rellenar la pantalla, se puede utilizar la propiedad BackgroundImageLayout para hacer lo mismo para un formulario.

Sugerencia

Muchos de los otros controles, como Panel, GroupBox e incluso el control Button también tienen una propiedad **BackgroundImage**. Pruébelos.

Inténtelo

Para mostrar una imagen de fondo en un formulario

En el Explorador de soluciones, seleccione Form1.vb y en el menú Ver elija Diseñador.

Seleccione el formulario haciendo clic en él fuera del control **PictureBox**.

En la ventana **Propiedades**, haga clic en el botón **...** ubicado junto a la propiedad **BackgroundImage** para abrir el cuadro de diálogo **Seleccionar recurso**.

En la lista **Entrada**, elija una de las imágenes que agregó antes y haga clic en **Aceptar**.

Observe que la imagen se muestra en el formulario detrás de **PictureBox** y se ordena en mosaico de manera predeterminada.

Nota

Si la imagen en el control PictureBox es demasiado grande, es posible que no se pueda ver la imagen de fondo. En este caso, seleccione el control PictureBox y arrástrelo hacia la parte inferior del formulario.

Seleccione la propiedad **BackgroundImageLayout** y establézcala en Stretch.

Observe cómo la imagen se expande para rellenar todo el formulario.

Haga doble clic en el formulario para abrir el Editor de código.

Asegúrese de que **Eventos de Form1** esté seleccionado en el cuadro desplegable del lado izquierdo y elija **Hacer clic** desde el cuadro desplegable del lado derecho.

Agregue el código siguiente al controlador de eventos **Form1_Click**

Visual Basic Express Copiar código

```
If Me.BackgroundImageLayout = ImageLayout.Stretch Then
  Me.BackgroundImageLayout = ImageLayout.Center
Else
  Me.BackgroundImageLayout = ImageLayout.Stretch
End If
```

Presione F5 para ejecutar el programa. Cuando aparece el formulario, haga clic en él para cambiar el diseño.

Proporcionar opciones al usuario: crear menús en tiempo de diseño

En esta lección, aprenderá a crear menús y a escribir un código que se ejecuta cuando se seleccionan los elementos de menú.

Los menús proporcionan a los usuarios una manera fácil y familiar de realizar elecciones relacionadas con el programa. Los usos comunes para los menús incluyen: exponer las opciones del programa, agregar accesos directos para tareas comunes como cortar y pegar o cargar y guardar los archivos.

Visual Basic Expressfacilita la implementación de los menús. Puede utilizar el control MenuStrip para crear menús gráficamente. Cuando se arrastra hasta un formulario, el control **MenuStrip** aparece como un cuadro con las palabras "escriba aquí" situadas en la parte superior del formulario. Puede hacer clic en el cuadro y escribir en él para crear los títulos de menú.

Cuando se establece el título para un elemento de menú, se pueden crear elementos de menú adicionales abajo y a la derecha del primero, lo que le permite ampliar el menú con tantos elementos o subelementos adicionales como desee. Cuando la apariencia del menú se ha completado, puede crear controladores de eventos para controlar los eventos Click para cada elemento.

Inténtelo

Para agregar un menú

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Menus` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **MenuStrip** hasta el formulario.

Independientemente de dónde lo coloca, el control **MenuStrip** se asocia a la parte más alta del formulario.

Es posible que haya observado que hay un icono **MenuStrip1** agregado en un área gris debajo del formulario, esta área se denomina bandeja de componentes. Si hace clic fuera del control **MenuStrip**, desaparecerá, puede volverlo a ver haciendo clic en el icono **MenuStrip1**.

En el formulario, haga clic en el control **MenuStrip**, escriba `File` y, a continuación, presione **Entrar**.

Aparecen nuevos cuadros para las entradas adicionales del menú abajo y a la derecha del primer elemento de menú. Éstos constituyen espacios para los elementos adicionales del menú. Puede continuar agregando elementos de menú en cualquier dirección hasta que se complete el menú.

En el cuadro situado debajo del primer cuadro, escriba `Exit` y, a continuación, presione **Entrar**.

Haga doble clic en el menú **Salir** para abrir el Editor de código.

En el controlador de eventos **ExitToolStripMenuItem_Click**, escriba el siguiente código.

Visual Basic Express | Copiar código

```
Application.Exit()
```

Presione F5 para ejecutar el programa. Con el mouse (ratón), seleccione el menú **Archivo** y, a continuación, elija **Salir**. Se cierra la aplicación.

En esta lección, aprendió a utilizar el control **MenuStrip** para diseñar los menús. Ahora, puede continuar con la siguiente lección sobre los temporizadores o puede explorar maneras más avanzadas de utilizar los menús en Información detallada: más información acerca de los menús y luego seguir con la lección de los temporizadores.

Información detallada: más información acerca de los menús

En esta lección, aprenderá a habilitar o deshabilitar menús en tiempo de ejecución, así como a crear menús emergentes.

En la lección anterior, aprendió a utilizar el control MenuStrip para crear menús que permitan a los usuarios elegir opciones relacionadas con el programa. Sin embargo, en ciertos casos, es posible que algunas opciones sólo estén disponibles en determinados momentos. Por ejemplo, un comando de menú **Copiar** sólo estará disponible si hay algo que se pueda copiar.

La mayoría de los programas deshabilitan, en lugar de ocultar, los comandos de menú cuando no están disponibles. Cuando un elemento de menú se deshabilita, el texto del menú pasa a estar atenuado y, al hacer clic en el elemento de menú, no se realiza ninguna acción. Al utilizar un control **MenuStrip**, puede deshabilitar y habilitar elementos de menú mediante la propiedad Enabled de MenuItem.

¡Inténtelo!

Para deshabilitar o habilitar elementos de menú

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Menus2` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **MenuStrip** y un control TextBox hasta el formulario.

En el formulario, haga clic en el control **MenuStrip** y escriba `Edit,` a continuación, presione `ENTRAR`.

En el cuadro situado debajo del primer cuadro, escriba `Copy,`, a continuación, presione ENTRAR.

En la ventana **Propiedades**, establezca la propiedad **Enabled** de **CopyToolStripMenuItem** en **False**.

Haga doble clic en el control **TextBox** para abrir el Editor de código.

En el controlador del evento **TextBox1_TextChanged**, escriba el siguiente código.

Visual Basic Express | Copiar código

```
If Textbox1.Text <> "" Then
  CopyToolStripMenuItem.Enabled = True
Else
  CopyToolStripMenuItem.Enabled = False
End If
```

Presione F5 para ejecutar el programa. Haga clic en el menú **Edición**; el elemento de menú **Copiar** estará deshabilitado. Escriba algún texto en el control **TextBox** y, a continuación, vuelva a hacer clic en el menú **Edición**; el elemento de menú **Copiar** estará ahora habilitado.

Crear menús emergentes

Muchos programas utilizan menús emergentes, también conocidos como *menús contextuales*, para facilitar el acceso a los comandos que se utilizan normalmente. El acceso a un menú contextual se obtiene haciendo clic con el botón secundario del mouse en un formulario o en un control en tiempo de ejecución. Puede crear sus propios menús contextuales en Visual Basic Expressutilizando un control ContextMenuStrip.

Al igual que sucede con el control **MenuStrip**, cuando arrastra un control **ContextMenuStrip** hasta un formulario, el control **ContextMenuStrip** aparece como un cuadro en la parte superior del formulario con el texto "Escriba aquí" en su interior, y se agrega un icono a la bandeja de componentes. A diferencia de **MenuStrip**, sólo pueden agregarse elementos adicionales debajo del primer elemento de menú, creándose un menú vertical.

Además, es necesario que **ContextMenuStrip** esté asociado al formulario o al control donde desee que aparezca. Esto se realiza estableciendo la propiedad **ContextMenuStrip** del formulario o del control en el nombre del control **ContextMenuStrip**. Puede asociar un solo control **ContextMenuStrip** a tantos controles como desee.

Inténtelo

Para crear un menú contextual

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `ContextMenus` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control **ContextMenuStrip** hasta el formulario.

En la ventana **Propiedades**, seleccione la propiedad **ContextMenuStrip** del formulario y elija **ContextMenuStrip1** en la lista desplegable.

En el formulario, haga clic en el control **ContextMenuStrip** y escriba `Option1,` a continuación, presione ENTRAR.

En el cuadro situado debajo del primer cuadro, escriba `Option2,`, a continuación, presione ENTRAR.

Haga doble clic en el elemento de menú **Option1** para abrir el Editor de código.

En el controlador del evento **Option1ToolStripMenuItem_Click**, escriba el siguiente código.

Visual Basic Express — Copiar código

```
MsgBox("You chose Option 1")
```

En el **Editor de código**, seleccione **Option2ToolStripMenuItem** en el cuadro desplegable izquierdo y, a continuación, elija **Hacer clic en** del cuadro desplegable derecho.

Aparecerá un nuevo controlador de eventos denominado **Option2ToolStripMenuItem _Click** en el editor de código.

En el controlador del evento **Option2ToolStripMenuItem _Click**, escriba el código siguiente:

Visual Basic Express — Copiar código

```
MsgBox("You chose Option 2")
```

Presione F5 para ejecutar el programa. Haga clic con el botón secundario del mouse en el formulario y después en uno de los elementos del menú contextual: aparecerá un cuadro de mensaje que notifica la opción elegida.

A tiempo: utilizar controles Timer para realizar acciones regulares

En esta lección, obtendrá información sobre cómo utilizar el componente Timer para realizar acciones no solicitadas por datos proporcionados por el usuario.

A veces, encontrará de utilidad realizar acciones repetidamente en los programas, por ejemplo, guardar un archivo cada pocos minutos o actualizar la

interfaz de usuario. El componente **Timer** permite realizar las acciones fijas regularmente sin ninguna entrada por parte del usuario.

El componente **Timer** se diferencia de los controles que ha utilizado hasta ahora en que no tiene una representación visual en tiempo de ejecución. Los controles que no tienen ninguna representación visual se conocen como *componentes*. Dado que el usuario no puede de ninguna manera interactuar directamente con el componente **Timer**, se ejecuta en segundo plano.

El componente **Timer** tiene dos propiedades y un evento que son los más utilizados. La propiedad Enabled determina si el componente **Timer** funciona. Si la propiedad **Enabled** se establece en **True**, el componente **Timer** está activo. Si la propiedad **Enabled** se establece en **False**, el componente **Timer** no está activo.

La propiedad Interval determina el número de milisegundos entre los pasos del componente **Timer**. Por ejemplo, si la propiedad **Interval** se establece en `1000`, el componente **Timer** provocará el evento Tick cada 1.000 milisegundos o cada segundo.

El componente **Timer** provoca el evento **Tick** a intervalos regulares que dependen del valor de la propiedad **Interval**. Puede agregar código a un controlador de eventos **Timer.Tick** y este código se ejecutará cuando el evento **Tick** se active.

Al establecer las propiedades **Enabled** y **Interval** y al agregar el código al controlador de eventos **Tick**, puede crear código que se ejecute a intervalos regulares sin necesidad de la acción del usuario.

Inténtelo

Para utilizar un componente Timer

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Timer` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre un control Label y un control **Timer** hasta el formulario.

El componente **Timer** no aparece en el propio formulario, sino en la bandeja de componentes bajo el formulario. Esto es porque el componente Timer no tiene una representación visual.

Seleccione el componente **Timer** y, a continuación, en la ventana **Propiedades**, establezca la propiedad **Enabled** en **True** y la propiedad **Interval** en `1000`.

Haga doble clic en el componente **Timer** para abrir el Editor de código.

En el controlador del evento **Timer1_Tick**, escriba el siguiente código.

Visual Basic Express — Copiar código

```
Label1.Text = My.Computer.Clock.LocalTime.ToLongTimeString
```

Presione F5 para ejecutar la aplicación.

El texto de la etiqueta se actualiza cada segundo con la hora correcta.

¿Qué salió mal? Encontrar y corregir errores mediante depuración

Al escribir un programa, se pueden producir y se producirán errores. Es posible que se cometa un error tipográfico, el programa se puede comportar no como lo esperaba o no se puede ejecutar en lo absoluto. Cuando hay un error en el programa, debe encontrarlo y corregirlo, el proceso de encontrar y corregir los errores se denomina *depuración*.

En las siguientes lecciones, aprenderá sobre varias técnicas para depurar un programa Visual Basic.

Encontrar errores: introducción a la depuración en Visual Basic

En esta lección, obtendrá información sobre cómo corregir errores del programa mediante la depuración.

No importa lo minuciosamente que se diseñe un programa o se escriba el código, siempre pueden aparecer errores. En ocasiones los errores impedirán que se inicie el programa, unas veces harán que el programa deje de ejecutarse o se *bloquee* y otras se ejecutará pero no ofrecerá los resultados esperados.

Y, por supuesto, cuando los errores aparecen, querrá encontrarlos y corregirlos. Los errores de un programa se conocen normalmente como *errores*, y el proceso de encontrarlos y corregirlos se denomina *depurar*.

El proceso de depuración es *iterativo*; es decir, se repetirá una y otra vez. Por lo general, escribe código, ejecuta el programa hasta que aparece un error, encuentra el error, lo corrige y, a continuación, ejecuta el programa de nuevo.

En la mayoría de los casos, no necesita detener el programa para corregirlo. Puede corregir el código donde apareció el error y seguir ejecutando el programa desde allí; este proceso se llama *Editar y continuar*.

La depuración se realiza en el IDE (entorno de desarrollo integrado) de Visual Basic, que contiene varios comandos y ventanas especiales para ayudar a encontrar los errores. Obtendrá más información en las lecciones siguientes.

¡Inténtelo!

Nota

En este ejemplo hay una excepción. Las excepciones son objetos que se crean (y producen) cuando el programa detecta un error. Se crean distintos tipos de excepciones, dependiendo del tipo de error generado. Con los valores predeterminados del usuario, si se produce una excepción cuando se ejecuta el programa de Visual Basic, aparecerá un cuadro de diálogo que describe el error y ayuda a corregirlo.

Para utilizar el proceso de editar y continuar

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Edit` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador del evento **Form_Load**, agregue el siguiente código:

Visual Basic Express — Copiar código

```
Dim number As Integer = 1
Dim numbers As String = ""
MsgBox(numbers + 1)
```

Presione F5 para ejecutar el programa. Se detendrá el programa y se mostrará un cuadro de diálogo de excepciones con el mensaje "No se controló InvalidCastException".

La excepción ha ocurrido porque hay un error tipográfico en el código. Se utilizó la variable equivocada: debería ser `number`, **Integer**, no `numbers`, que es una variable **String**.

Observe que el programa todavía está en ejecución; está en el modo de *interrupción* de depuración. Con Editar y continuar, puede corregir el error sin tener que detener el programa (ni volver a ejecutarlo para comprobarlo).

En el Editor de código, cambie `numbers + 1` por `number + 1`.

Presione F5 para continuar. Debe aparecer un cuadro de mensaje con el número 2.

Conozca sus errores: tres tipos de errores de programación

En esta lección, conocerá los diferentes tipos de errores que pueden aparecer al escribir un programa.

Incluso los programadores más experimentados cometen errores; y conocer cómo depurar una aplicación y encontrar esos errores es una parte importante de la programación. No obstante, antes de obtener información sobre el proceso de depuración, conviene conocer los tipos de errores que deberá buscar y corregir.

Los errores de programación pertenecen a tres categorías: *errores de compilación*, *errores en tiempo de ejecución* y *errores lógicos*. Las técnicas para depurar cada uno de ellos se tratarán en las tres lecciones siguientes.

Errores de compilación

Los errores de compilación, también conocidos como *errores del compilador*, son errores que impiden que su programa se ejecute. Cuando se presiona F5 para ejecutar un programa, Visual Basic Expresscompila el código en un lenguaje binario que entiende el equipo. Si el compilador de Visual Basic Expressse encuentra con código que no entiende, emite un error de compilador.

La mayoría de los errores del compilador se deben a errores cometidos al escribir el código. Por ejemplo, puede escribir mal una palabra clave, omitir alguna puntuación necesaria o intentar utilizar una instrucción **End If** sin antes utilizar una instrucción **If**.

Afortunadamente el Editor de código de Visual Basic Expressfue diseñado para identificar estos errores antes de que se intente ejecutar el programa. Aprenderá a encontrar y corregir los errores de compilación en la lección siguiente, Error ortográfico: encontrar y eliminar errores del compilador.

Errores en tiempo de ejecución

Los errores en tiempo de ejecución son errores que aparecen mientras se ejecuta su programa. Estos errores aparecen normalmente cuando su programa intenta una operación que es imposible que se lleve a cabo.

Un ejemplo de esto es la división por cero. Suponga que tiene la instrucción siguiente:

```
Speed = Miles / Hours
```

Si la variable `Hours` tiene un valor de 0, se produce un error en tiempo de ejecución en la operación de división. El programa se debe ejecutar para que se pueda detectar este error y si `Hours` contiene un valor válido, no se producirá el error.

Cuando aparece un error en tiempo de ejecución, puede utilizar las herramientas de depuración de Visual Basic Expresspara determinar la causa. Aprenderá a encontrar y corregir los errores en tiempo de ejecución en la lección ¡Uff! A mi programa no le ha gustado Encontrar y eliminar errores en tiempo de ejecución.

Errores lógicos

Los errores lógicos son errores que impiden que su programa haga lo que estaba previsto. Su código puede compilarse y ejecutarse sin errores, pero el resultado de una operación puede generar un resultado no esperado.

Por ejemplo, puede tener una variable llamada `FirstName` y establecida inicialmente en una cadena vacía. Después en el programa, puede concatenar `FirstName` con otra variable denominada `LastName` para mostrar un nombre completo. Si olvida asignar un valor a `FirstName`, sólo se mostrará el apellido, no el nombre completo como pretendía.

Los errores lógicos son los más difíciles de detectar y corregir, pero Visual Basic Expresstambién dispone de herramientas de depuración que facilitan el trabajo. Aprenderá a encontrar y corregir los errores lógicos en ¿Qué? Esto no debiera haber ocurrido. Detectar errores lógicos.

Error ortográfico: encontrar y eliminar errores del compilador

En esta lección, aprenderá a encontrar y corregir los errores del compilador.

Como vimos en la lección anterior, los errores del compilador aparecen cuando el compilador de Visual Basic Expressse encuentra con código irreconocible, generalmente porque se cometió algún error al escribir. Dado que los errores del compilador impiden que se ejecute un programa, deberá encontrarlos y corregirlos, o depurarlos, antes de ejecutar el programa.

Encontrar y corregir errores del compilador

Encontrar los errores del compilador es bastante fácil, ya que el programa no se ejecuta hasta que se han corregido. Cuando presiona F5, si hay algún error

del compilador, aparecerá un cuadro de diálogo que indica "**Errores al generar. ¿Desea continuar?**". Si selecciona **Sí**, se ejecutará la última versión sin errores del programa; si selecciona **No**, el programa se detendrá y aparecerá la ventana **Lista de errores**.

La ventana **Lista de errores** muestra toda la información sobre el error, incluida su descripción y ubicación en el código. Si hace doble clic en el error en la **Lista de errores**, se resaltará la línea incorrecta del código en el Editor de código. También puede presionar F1 para mostrar Ayuda y obtener más información sobre el error y cómo corregirlo.

El Editor de código de Visual Basic Expresstambién puede ayudar a encontrar y corregir los errores del compilador antes incluso de que se intente ejecutar el programa. Mediante una característica llamada *IntelliSense*, Visual Basic Expressobserva el código a medida que se escribe y si encuentra código que producirá un error del compilador, lo subraya con una línea ondulada de color azul. Si mantiene presionado el mouse sobre esa línea, se muestra un mensaje que describe el error. Si la ventana **Lista de errores** está visible, también mostrará los mensajes de error.

Inténtelo

Para encontrar y corregir errores del compilador

En el menú **Archivo**, seleccione **Nuevo Proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `CompilerErrors` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador de eventos **Form_Load**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
End If
```

Presione ENTRAR. Verá una línea ondulada de color azul debajo de `End If`.

Si mantiene presionado el mouse sobre la línea, verá el mensaje "'End If' debe ir precedida por la instrucción 'If' " correspondiente.

Cambie el código para que tenga la siguiente apariencia.

Visual Basic Express | Copiar código

```
If 1 < 2 Then
End If
```

Observe que ha desaparecido la línea ondulada de color azul.

Agregue la nueva línea de código siguiente después de la instrucción If... Then.

Visual Basic Express | Copiar código

```
MgBox("Hello")
```

Presione F5 para ejecutar el programa. Aparecerá un cuadro de diálogo con el mensaje "**Errores al generar. ¿Desea continuar y ejecutar la última versión generada correctamente**?"

Haga clic en **No**. Se mostrará la ventana **Lista de errores** con el mensaje de error "No se ha declarado el 'nombre MgBox'".

Haga doble clic en el mensaje de error de la **Lista de errores** y cambie el código por `MsgBox("Hello")`.

Presione F5 de nuevo. Ahora el programa debería ejecutarse y causar la aparición de un cuadro de mensaje.

¡Uff! A mi programa no le ha gustado Encontrar y eliminar errores en tiempo de ejecución

En esta lección, aprenderá a depurar un programa y a corregir errores en tiempo de ejecución.

Como aprendió en su momento, los errores en tiempo de ejecución se producen cuando el programa intenta realizar una operación que es imposible finalizar. Cuando se produce un error en tiempo de ejecución, el programa se detiene y aparece un mensaje de error; debe depurar el error y corregirlo para que el programa pueda continuar.

Encontrar y corregir errores en tiempo de ejecución

La mayoría de los errores en tiempo de ejecución se producen porque se cometió un error en el código; por ejemplo, olvidó asignar un valor a una variable antes de utilizarla. Cuando se ejecute el programa y se descubra el

error, el programa se detendrá y el cuadro de diálogo **Ayudante de excepciones** se mostrará en la ventana Editor de código. Cuando esto sucede, el programa está en modo de *interrupción*, que es el modo en que se realiza la depuración.

El cuadro de diálogo **Ayudante de excepciones** contiene una descripción del error, así como sugerencias para la solución de problemas que indican la causa. Puede hacer clic en las sugerencias sobre solución de problemas para mostrar los temas de Ayuda y obtener más detalles.

Es necesario corregir el error para que pueda continuar con el programa; para ello, debe inspeccionar el código para encontrar la causa del error. Por ejemplo, si sospecha que se produjo un error porque una variable contiene el valor equivocado, estando todavía en el modo de interrupción, puede utilizar IntelliSense para ver el valor de la variable. Cuando se coloca el mouse sobre la variable en el Editor de código, la información sobre herramientas muestra el valor de la variable. Si el valor no es lo que esperaba, compruebe en el código anterior dónde se estableció el valor y después arregle el código y continúe.

Inténtelo

Para revisar el valor de una variable

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `RunTimeErrors` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador de eventos **Form_Load**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
Dim miles As Integer = 0
Dim hours As Integer = 0
Dim speed As Integer = 0
```

Visual Basic Express — Copiar código

```
miles = 55
speed = miles / hours
```

```
MsgBox(CStr(speed) & " miles per hour")
```

Presione F5 para ejecutar el programa. Aparece un cuadro de diálogo **Ayudante de excepciones** con el mensaje "No se controló OverflowException".

Una línea de puntos que va del cuadro de diálogo a su archivo de código señala la línea de código que produjo el error.

Observe que la primera sugerencia sobre solución de problemas del **Ayudante de excepciones** indica que debe asegurarse de no estar dividiendo por cero.

Mueva el mouse sobre la variable `miles` y manténgalo ahí durante unos segundos. La información sobre herramientas que verá dice "miles 55".

Ahora mueva el mouse sobre la variable `hours`; la información sobre herramientas debe decir "hours 0".

Debido a que no se puede dividir por cero y el valor de `hours` es cero, ya ha encontrado la causa del error: no haber actualizado el valor de `hours`.

Agregue la siguiente línea de código sobre la línea `miles = 55`.

Visual Basic Express — Copiar código

```
hours = 2
```

Haga clic en la flecha amarilla situada en el margen izquierdo del código y arrástrela hasta la línea `hours = 2`.

Esto permite que el programa continúe desde esa línea en lugar de continuar desde la línea que contiene el error. Para que se reconozca la solución del error es necesario ejecutar la nueva línea de código recién agregada.

Presione F5 para que el programa continúe. Aparece un cuadro de diálogo que muestra "28 miles per hour".

Información detallada: qué ocurriría si... Comprobar código en la ventana Inmediato

En esta lección, aprenderá a evaluar y ejecutar un código utilizando la ventana **Inmediato**.

En la lección anterior, aprendió cómo corregir errores en tiempo de ejecución utilizando el **Ayudante de excepciones**. Sin embargo, a veces es posible que no esté claro cómo corregir un error y se desee probar una posible corrección

sin cambiar el código. Una ventana de depuración especial, la ventana **Inmediato**, permite hacer eso y más.

La ventana Inmediato

Cuando el programa está en modo de interrupción, se puede utilizar la ventana **Inmediato** para ejecutar fragmentos de código o evaluar variables y expresiones. Por ejemplo, si aparece un error en tiempo de ejecución debido a una variable vacía, puede comprobar el valor de la variable. Puede utilizar también la ventana **Inmediato** para asignar un valor a esa variable y comprobar cómo se ejecuta el resto del programa.

Sugerencia

Cuando ejecuta el programa en modo de depuración, puede poner el programa en modo de interrupción en cualquier momento, seleccionando Interrumpir del menú Depurar.

Para ejecutar el código en la **Ventana Inmediato** escríbalo como lo haría en el Editor de código y presione ENTRAR. Para evaluar una variable o expresión, escriba un signo de interrogación seguido por la variable o expresión que desea evaluar y presione ENTRAR, el resultado se mostrará en la siguiente línea.

¡Inténtelo!

Para probar el código en la ventana Inmediato

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Immediate` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el **Cuadro de herramientas**, arrastre dos controles TextBox y un control **Button** al formulario.

Haga doble clic en el botón para abrir el Editor de código.

En el controlador de eventos **Button_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
Dim miles As Integer = 0
Dim hours As Integer = 0
Dim speed As Integer = 0
```

Visual Basic Express | Copiar código

```
miles = CInt(Textbox1.Text)
hours = CInt(Textbox2.Text)
speed = miles / hours
MsgBox(CStr(speed) & " miles per hour")
```

Presione F5 para ejecutar el programa. Escriba `100` en el primer cuadro de texto y, a continuación, escriba `0` en el segundo cuadro de texto.

Haga clic en **Button1**. El programa se detendrá y aparecerá el cuadro de diálogo **Ayudante de excepciones** con el mensaje "No se controló OverflowException".

En la ventana **Inmediato** en la parte inferior del IDE, escriba `?miles` y presione ENTRAR.

El valor `100` debe aparecer en la línea siguiente.

Sugerencia

Puede abrir en cualquier momento la ventana Inmediato eligiendo Ventanas, Inmediato en el menú Depurar.

Escriba `?hours` y presione ENTRAR.

El valor 0 debe aparecer en la línea siguiente.

Escriba `hours = 4` y presione ENTRAR. Escriba `?hours` y presione ENTRAR.

Observe que el valor de `hours` es ahora `4`, el valor que especificó en la línea anterior. Puede cambiar el valor de `hours` en la ventana **Inmediato** sin cambiar el código del programa.

Presione F5 para continuar. Se mostrará un cuadro de mensaje con el resultado.

Sugerencia

Para evitar que se produzca este error en tiempo de ejecución, agregue un controlador de errores que compruebe que hay un número válido en el bloque Try y muestre un mensaje al usuario en el bloque Catch. Para obtener más información sobre controladores de errores, vea Qué hacer cuando algo sale mal: control de errores.

¿Qué? Esto no debiera haber ocurrido. Detectar errores lógicos

En esta lección, aprenderá a encontrar errores lógicos en el programa.

En lecciones anteriores, aprendió a encontrar y corregir errores del compilador y errores en tiempo de ejecución. El tercer tipo de error de programación, los errores lógicos, puede ser el más difícil de descubrir. Con los errores lógicos no se obtiene ninguna advertencia, se ejecutará el programa pero proporcionará resultados incorrectos. Es necesario investigar el código y determinar la razón del problema.

Afortunadamente, las herramientas de depuración de Visual Basic Expresspueden ayudar. Dos técnicas de depuración, que establecen *puntos de interrupción* e *instrucciones paso a paso* a través del código, permiten inspeccionar el código línea por línea mientras se ejecuta para encontrar el error.

Se puede establecer un punto de interrupción en el **Editor de código** para cualquier línea ejecutable de código. Cuando se ejecuta el programa, los puntos de interrupción fuerzan que se detenga y el programa entra en el modo de interrupción cuando llega a esa línea de código. Puede obtener la información que desee sobre el estado del programa en ese momento. Puede verificar el valor de cualquier variable, comprobar expresiones en la ventana **Inmediato** o realizar cambios en el código con Editar y continuar.

Cuando está en modo de interrupción, puede recorrer el código, ejecutando línea por línea para ver cómo funciona. Al presionar la tecla F8, se ejecutará la línea de código actual y se detendrá en la línea siguiente. Puede inspeccionar los valores de variables para ver cómo cambian de una línea a la siguiente.

Si la línea de código actual llama a una función o procedimiento **Sub** en otra parte del código, cuando presiona F8, la ejecución se desplazará a ese procedimiento. Una vez que se haya ejecutado ese procedimiento, el programa volverá a la línea siguiente a la que llamó al procedimiento. Si no desea recorrer un procedimiento, puede presionar MAYÚS+F8 para saltarlo.

Inténtelo

Para observar un error lógico

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `LogicErrors` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Desde el **Cuadro de herramientas**, arrastre dos controles TextBox y un control Button hacia el formulario.

Haga doble clic en **Button1** para abrir el Editor de código.

En el controlador de eventos **Button1_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
Dim minutes As Integer = CInt(Textbox1.Text)
Dim miles As Double = CDbl(Textbox2.Text)
Dim hours As Double = 0
hours = minutes / 60
MsgBox("Average speed " & GetMPH(hours, miles))
```

Debajo de la línea `End Sub`, agregue la siguiente función.

Visual Basic Express — Copiar código

```
Function GetMPH(ByVal miles As Double, ByVal hours As Double) _
As String
    GetMPH = CStr(miles / hours)
End Function
```

Presione F5 para ejecutar el programa. En el primer cuadro de texto, escriba `10` (para representar 10 minutos) y en el segundo cuadro de texto, escriba `5` (para representar las millas) y, a continuación, haga clic en **Button1**.

Aparecerá un cuadro con el mensaje "Average speed 0.03333334" (velocidad media 0,03333334) ; no obstante, si recorre 5 millas en diez minutos, la respuesta correcta serían 30 mph.

Mantenga abierto el proyecto: en el siguiente procedimiento aprenderá cómo encontrar el error lógico.

Encontrar errores lógicos

En el último ejemplo, algo está obviamente mal con la lógica del programa. Según el resultado, viaja menos de una milla por hora, no treinta millas por hora como espera, pero ¿dónde está el error?

En el siguiente procedimiento se establecerá un punto de interrupción y se examinará el código para encontrar el error.

Inténtelo

Para establecer un punto de interrupción y recorrer el código

En el Editor de código, busque la línea `hours = minutes / 60` y haga clic en el margen izquierdo.

Aparecerá un punto rojo en el margen y el código resaltado en rojo, lo que representa un punto de interrupción.

Presione F5 para ejecutar el programa nuevamente. En el primer cuadro de texto, escriba `10` y en el segundo cuadro de texto, escriba `5`. Haga clic en **Button1**.

El programa se detendrá cuando llegue al punto de interrupción. La línea `hours = minutes / 60` aparecerá resaltada en amarillo.

Inspeccione los valores de las variables manteniendo el mouse sobre ellos; el valor de `hours` debe ser `0` y el valor de `minutes` debe ser `10`.

Presione F8 para ejecutar la línea `hours = minutes / 60` y pasar a la siguiente línea.

Inspeccione los valores de las variables de la línea `MsgBox("Average speed " & GetMPH(hours, miles))`, el valor de `hours` debe ser ahora `0.166666672` y el valor de `miles` debe ser `5.0`.

Presione F8 de nuevo para ejecutar la línea actual.

Observe que la ejecución baja a la línea `Function GetMPH`.

Inspeccione los valores de las variables en esta línea; observará que el valor de `miles` es ahora `0.166666672` y el de `hours` es `5.0`, lo contrario de lo que eran en la línea anterior. Ha encontrado el error.

Mantenga abierto el proyecto: en el siguiente procedimiento aprenderá a corregir el error lógico.

Corregir errores lógicos

En el último procedimiento, los valores para las variables `miles` y `hours` cambiaron de lugar. ¿Puede identificar la causa?

Si examina la línea `MsgBox("Average speed " & GetMPH(hours, miles))`, verá que a la función `GetMPH` se pasan dos argumentos, `hours` y `miles`, en ese orden. Si examina la declaración de función `Function`

`GetMPH(ByVal miles As Double, ByVal hours As Double)...`, observará que los argumentos se muestran como `miles` primero y como `hours` después.

Se produjo un error en la lógica porque los argumentos se pasaron en el orden equivocado, produciendo un cálculo incorrecto. Si los argumentos hubieran sido de tipos diferentes, habría visto un error en tiempo de ejecución, pero como los argumentos eran del mismo tipo, no se produjo el error. Fue un error simple, pero el error resultante fue difícil de encontrar.

En el siguiente procedimiento se establecerá un punto de interrupción y se recorrerá el código para encontrar el error.

Inténtelo

Para corregir el error lógico

En el Editor de código, cambie la línea `MsgBox("Average speed " & GetMPH(hours, miles))` para que se lea de la siguiente manera:

Visual Basic Express — Copiar código

```
MsgBox("Average speed " & GetMPH(miles, hours))
```

Haga clic en el punto rojo en el margen izquierdo para borrar el punto de interrupción.

Presione F5 para ejecutar el programa. En el primer cuadro de texto, escriba `10` y en el segundo cuadro de texto, escriba `5.` A continuación, haga clic en **Button1**.

Esta vez el cuadro de mensaje debe mostrar el resultado correcto, "Average speed 30" (velocidad media 30).

Puede parecer que se corrigió el programa, pero hay otro error lógico aun más difícil de encontrar. Si desea probar y encontrarlo, mantenga el proyecto abierto, lo utilizará de nuevo en la lección Crédito extra: todavía hay algo erróneo.

Crédito extra: todavía hay algo erróneo

En esta lección, aprenderá a rastrear un error lógico que sólo se produce en situaciones determinadas.

En la lección anterior, ¿Qué? Esto no debiera haber ocurrido. Detectar errores lógicos, aprendió a encontrar y corregir un error de lógica. En el código de

ejemplo de esa lección, aún existe un error grave oculto: uno que es más difícil de encontrar porque sólo se produce en situaciones determinadas.

Probar un programa

Como desarrollador, se encuentra en desventaja cuando debe probar el programa para ver si se comporta según lo deseado. Sabe cómo debe funcionar, de modo que es improbable que se cometa un error que pueda revelar un error lógico. Sin embargo, un usuario que no esté familiarizado con el programa puede y hará cosas en las que no ha pensado.

Por ejemplo, en un programa que calcula millas por hora dividiendo el número de millas recorridas por el número de horas que demoró el viaje, ¿qué pasa si el usuario escribe cero para las horas o las millas? Probémoslo y vea.

Inténtelo

Para probar el programa

Abra el proyecto `LogicErrors` que se creó en la lección anterior, ¿Qué? Esto no debiera haber ocurrido. Detectar errores lógicos.

> **Nota**
>
> Si no finalizó o no guardó el proyecto anterior, deberá regresar y finalizarlo antes de poder continuar.

Presione F5 para ejecutar el programa. En el primer cuadro de texto, escriba `0` (para representar minutos) y en el segundo escriba `5` (para representar millas) y, a continuación, haga clic en **Button1**.

Se muestra un cuadro de mensaje con el mensaje "Velocidad media infinito."

Mantenga abierto el proyecto: en el siguiente procedimiento aprenderá a encontrar el error lógico.

5 dividido por 0 = ¿Infinito?

En el procedimiento anterior, es posible que "Infinito" no sea lo que se esperaba, pero es matemáticamente correcto: 0 cabe en 5 un número infinito de veces. Sin embargo, éste no es el resultado que se desea que los usuarios del programa vean. ¿Puede pensar en una forma de evitar esto?

Podría pensar en agregar un controlador de errores, un procedimiento descrito en la lección Qué hacer cuando algo sale mal: control de errores. Sin embargo, en este caso no funcionaría porque el resultado "Infinito" no es un error, únicamente no es lo que desea.

Puesto que no es útil mostrar una velocidad de cero, una manera de corregir el problema es probar un valor de cero y advertir al usuario que debe escribir un número mayor. Mientras se realiza esto, también se puede evitar que el usuario escriba números negativos, puesto que los números negativos también pueden generar un resultado falso.

En el siguiente procedimiento, se modificará el código en el controlador de eventos **Button1_Click** para llamar sólo a la función `GetMPH` si los valores son mayores que cero.

Inténtelo

Para corregir el error

En el Editor de código, cambie el código en el controlador de eventos **Button1_Click** de la siguiente manera:

Visual Basic Express — Copiar código

```
Dim minutes As Integer = CInt(Textbox1.Text)
Dim miles As Double = CDbl(Textbox2.Text)
Dim hours As Double = 0
If minutes <= 0 Or miles <= 0 Then
  MsgBox("Please enter a number greater than zero")
Else
  hours = minutes / 60
  MsgBox("Average speed " & GetMPH(hours, miles))
End If
```

Presione F5 para ejecutar el programa nuevamente. En el primer cuadro de texto, escriba `0,` y en el segundo, escriba `5`. A continuación, haga clic en **Button1**.

Aparecerá el cuadro de mensaje indicándole que especifique un número mayor que 0. Inténtelo probando el programa con otras combinaciones de números hasta que esté seguro de que se ha corregido el error.

Pasos siguientes

En esta lección, aprendió a encontrar y corregir un error lógico que produjo un comportamiento inesperado. En la siguiente lección, aprenderá a utilizar los comentarios en el código.

Agregar notas a programas: utilizar comentarios

En esta lección, obtendrá información sobre cómo crear comentarios en el código de sus programas.

El código que forma un programa puede ser difícil de leer y entender, sobre todo si el usuario no es la persona que lo escribió originalmente. Al utilizar *comentarios*, puede crear notas para sí mismo o para otros usuarios del código.

Los comentarios son entradas de texto del Editor de código y que omite el compilador de Visual Basic Expresscuando se ejecuta el programa. Por tanto, puede escribir una nota que explique lo que hace una sección determinada del programa, o bien un aviso para finalizar las tareas de programación incompletas.

El comentario se crea iniciando una línea con el carácter **'**. El ejemplo siguiente muestra cómo crear un comentario.

Visual Basic Express Copiar código

```
' This is a comment. WOW!
```

También puede agregar comentarios al final de las líneas, así como usar el carácter **'**. Este procedimiento suele realizar para proporcionar comentarios sobre líneas individuales de código, como se ve en el ejemplo siguiente.

Visual Basic Express Copiar código

```
MsgBox("Hello World!") ' This line causes a message box to appear.
```

Al igual que con los comentarios de una única línea, el programa omite todo lo que vaya después del carácter **'** de esa línea.

Utilizar comentarios para depuración

Otro uso común de los comentarios es evitar temporalmente que una línea de código se ejecute mientras depura su programa. Por ejemplo, suponga que tenía una línea que mostraba un cuadro de mensaje.

Visual Basic Express Copiar código

```
MsgBox("Hello World!")
```

Si quiere ejecutar el programa sin mostrar esa línea, pero no desea eliminarla permanentemente, utilice el carácter del comentario (') para ocultarla temporalmente de su programa, tal y como se muestra a continuación.

Visual Basic Express Copiar código

```
' MsgBox("Hello World!")
```

Como todo lo que va después del carácter `'` se omite, el programa se ejecutará sin ejecutar esa línea. Puede quitar el carácter `'` después y se mostrará el cuadro de mensaje.

¡Inténtelo!

Para insertar comentarios

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Comments` y, a continuación, haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el controlador del evento **Form1_Load**, escriba el siguiente código.

Visual Basic Express Copiar código

```
' This code will cause two message boxes to appear
MsgBox("This is Message Box 1") ' Display Message Box 1
MsgBox("This is Message Box 2") ' Display Message Box 2
```

Presione F5 para ejecutar el programa.

El programa se inicia y se muestran los dos cuadros de mensaje de uno en uno.

En el menú **Depurar**, elija **Detener depuración** para finalizar el programa.

En el Editor de código, agregue un carácter de comentario (`'`) a la primera línea del cuadro de mensaje, para que se lea lo siguiente.

Visual Basic Express Copiar código

```
' MsgBox("This is MessageBox 1") ' Ignore Message Box 1
```

Presione F5 para ejecutar el programa.

Observe que esta vez el programa omite la primera línea del cuadro de mensaje y sólo se muestra el segundo cuadro de mensaje.

Administrar registros: utilizar datos en un programa

La mayoría de los programas utiliza los datos de una forma u otra. Por ejemplo, en una lección anterior, especificó datos en forma de números; esos datos se utilizaron en un cálculo con el resultado devuelto en un cuadro de mensajes.

En programas muy simples, los datos se representan como campos dentro del programa. Sin embargo, para programas más complejos, los datos se almacenan en una estructura separada del programa, denominada *base de datos*.

En este conjunto de lecciones, aprenderá a crear una base de datos y a utilizarla para mostrar y actualizar datos de los programas.

Almacenar y obtener acceso a datos

En esta lección, aprenderá a utilizar una base de datos para almacenar datos y tener acceso a ellos.

Los *datos* son un concepto central en programación. La mayoría de los programas utilizan datos de una manera u otra. Por ejemplo, en una lección anterior, escribió datos en forma de números; después se utilizaron esos datos en un cálculo y se devolvieron en un cuadro de mensaje.

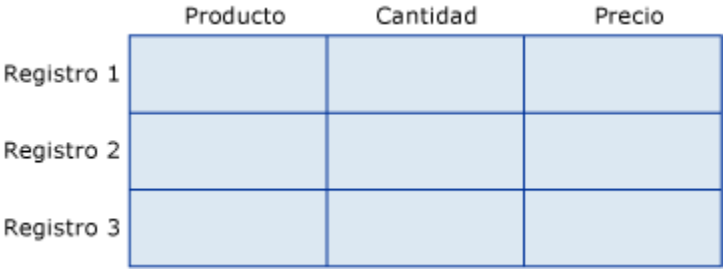
En programas muy simples, los datos se representan como campos dentro del programa. Sin embargo, para programas más complejos, los datos se almacenan en una estructura separada del programa, denominada *base de datos*.

¿Qué es una base de datos?

Una base de datos es una colección de datos, almacenados en un archivo independiente del programa. Los datos almacenados en una base de datos pueden ser de muchos tipos distintos: texto, números, imágenes y otros. Pueden conectarse distintos programas a la misma base de datos para ver y actualizar los datos que contiene.

Una base de datos se divide generalmente en una o más *tablas*. Una tabla es una colección de registros relacionados. Por ejemplo, si utilizó una base de datos que contenía los datos de un negocio pequeño, debería tener una tabla

que representara los productos, otra tabla para los pedidos y otra para los clientes.



Cada tabla se organiza en una cuadrícula de columnas y filas. Las columnas representan las categorías de los datos de un registro y las filas representan los registros individuales. Por ejemplo, en la ilustración anterior, la tabla **Orders** contiene una fila o registro separado que representa cada pedido y columnas que representan el producto pedido, junto con la cantidad y el precio.

Introducción a los datos

Para tener acceso a los datos de una base de datos del programa, debe tener primero una base de datos. Con Visual Basic, podrá crear fácilmente su propia base de datos o utilizar una base de datos creada por otra persona.

Con Visual Basic Express puede tener acceso a dos tipos diferentes de bases de datos: bases de datos de Microsoft SQL Server o de Microsoft Access. Para el propósito de estas lecciones, se utilizará una base de datos de SQL Server.

Cuando tenga una base de datos, puede conectarla al programa utilizando un objeto llamado *DataSet* y, a continuación, conectar los campos o controles de un formulario a los datos de la base de datos utilizando una técnica llamada *enlace de datos*.

Cuando un campo del programa, como un control **TextBox** , se *enlaza* a una columna de una tabla de base de datos, se pueden mostrar los datos de esa columna en el cuadro de texto, modificarlos en dicho cuadro de texto y guardarlos en la base de datos, o bien escribirlos en un nuevo registro y agregarlos a la base de datos.

Aunque esto puede sonar complicado, en realidad no es difícil. Las herramientas de base de datos de Visual Basic Expressfacilitan el trabajo con los datos, como verá en las siguientes lecciones.

Crear la primera base de datos

En esta lección, aprenderá a crear una base de datos que utilizará en lecciones posteriores para crear un programa de libreta de direcciones.

En la lección anterior, aprendió que una base de datos se puede utilizar para almacenar y recuperar datos para los programas de Visual Basic. Primero, debe tener una base de datos a la cual tener acceso. Si bien se puede utilizar una base de datos existente, para estas lecciones aprenderá a crear una nueva base de datos mediante Visual Database Tools, que se incluye en Visual Basic.

Requisitos previos

Para crear y tener acceso a una base de datos de SQL Server con Visual Basic Express, también debe instalar SQL Server . Éste se instala de manera predeterminada durante la instalación de Visual Basic Express, sin embargo, si decidió no instalarlo, deberá hacerlo antes de continuar.

Inténtelo

Para crear una base de datos

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `FirstDatabase` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

En el menú Proyecto, elija Agregar nuevo elemento.

En el cuadro de diálogo Agregar nuevo elemento, seleccione Base de datos SQL.

En el cuadro **Nombre**, escriba `FirstDatabase` y haga clic en **Agregar**.

Se iniciará el Asistente para la configuración de orígenes de datos.

En el Asistente para la configuración de orígenes de datos, haga clic en Cancelar.

Se agregará una nueva base de datos, `FirstDatabase.mdf`, al proyecto y aparecerá en el **Explorador de soluciones**.

Agregar una tabla

Como puede ver en el procedimiento anterior, crear una base de datos es fácil. En este punto, la base de datos no es útil, puesto que no contiene datos. En el siguiente procedimiento, se agregará una tabla a la base de datos, en este caso, una tabla para almacenar información de direcciones.

Inténtelo

Para agregar una tabla a la base de datos

En el menú Ver, seleccione Explorador de base de datos.

En el **Explorador de base de datos**, expanda el nodo (haga clic en el signo más) **FirstDatabase.mdf** y seleccione el nodo **Tables**.

En el menú Datos, elija Agregar nueva Tabla.

Se abrirá una ventana del **Diseñador de tablas**.

En la ventana **Propiedades**, seleccione **Nombre** y escriba `Addresses`.

En la ventana del **Diseñador de tablas**, seleccione el campo **Nombre de columna** y escriba `FirstName`.

Seleccione el campo **Tipo de datos** y seleccione **nvarchar (50)** de la lista desplegable; se activará automáticamente la columna **AllowNulls**.

Ahora se ha definido la primera columna en la nueva tabla.

Repita los dos pasos anteriores para agregar cuatro columnas más con los siguientes valores:

Nombre de columna: `LastName`, Tipo de datos: nvarchar(50)

Nombre de columna: `StreetAddress`, Tipo de datos: nvarchar(50)

Nombre de columna: `City`, Tipo de datos: nvarchar(50)

Nombre de columna: `Phone`, Tipo de datos: nvarchar(50)

En el menú Archivo, elija Guardar direcciones.

Agregar una clave

Ahora tiene una tabla en la base de datos que puede utilizar para almacenar datos de nombres, direcciones y teléfonos para la libreta de direcciones. El siguiente paso es: agregar una *clave* para evitar registros duplicados.

Una columna clave, también conocida como una *clave principal*, designa una columna o columnas en la tabla como un valor único. Sólo puede haber una fila en la tabla que contenga este valor; si intenta escribir una segunda fila con el mismo valor recibirá un error.

En el caso de la tabla `Addresses`, designe las columnas `FirstName` y `LastName` como clave principal, si bien puede conocer varias personas con el mismo nombre o apellido, es improbable que conozca dos personas con ambos.

Inténtelo

Para agregar una clave a la tabla

En el **Diseñador de tablas**, desactive la casilla de verificación **Permitir valores nulos** para las filas **Nombre** y **Apellido**

Seleccione las filas **Nombre** y **Apellido**.

> Sugerencia
>
> Puede hacer clic en el cuadrado gris a la izquierda del campo Nombre, presionar la tecla CTRL y hacer clic en la fila LastName para seleccionar ambos.

En el menú Diseñador de tablas, elija Establecer clave principal.

Aparecerá un pequeño símbolo de llave a la izquierda de cada fila.

En el menú Archivo, elija Guardar direcciones.

Agregar datos

Ahora tiene una base de datos que contiene una tabla única, `Addresses`. Por supuesto, una base de datos no es muy útil a menos que contenga datos. En el siguiente procedimiento, se agregarán algunos datos a la tabla `Addresses`. Si desea, puede sustituir los nombres y las direcciones de las personas que conoce por aquellos proporcionados en el ejemplo.

Inténtelo

Para agregar datos a la tabla

En el **Explorador de base de datos**, expanda el nodo **Tables**, seleccione el nodo **Direcciones** y, a continuación en el menú **Datos**, elija **Mostrar datos de tabla**.

Se abrirá una ventana de tabla de datos.

En la ventana de tabla de datos, seleccione el campo **Nombre** y escriba `Samantha`.

> Nota
>
> Observe que cuando selecciona el campo por primera vez, aparece el valor NULL en cada campo; null es un término de la base de datos que significa que el campo está vacío.

Seleccione el campo **LastName** y escriba `Smith`.

Seleccione el campo **Dirección** y escriba `123 45th Ave. E`.

Seleccione el campo **City** y escriba `Seattle`.

Seleccione el campo **Phone**, escriba `2065550100` y presione la tecla TAB.

Ahora ha definido el primer registro en la tabla Addresses.

Repita los cinco pasos anteriores para agregar dos registros más con los siguientes valores:

Nombre: Michael, **Apellido**: Alexander, **Dirección**: 789 W. Capital Way, **Ciudad**: Tacoma, **Teléfono**: 2065550101.

Nombre: Andrea, **Apellido**: Dunker, **Dirección**: 722 Moss Bay Blvd, **Ciudad**: Kirkland, **Teléfono**: 2065550102.

En el menú Archivo, seleccione Guardar todo para guardar el proyecto y la base de datos.

Según ha escrito los datos, quizá haya observado un pequeño icono de lápiz junto a los datos, que desaparece al utilizar la tecla TAB para moverse a la fila siguiente. El icono de lápiz significa que los datos no se han guardado en la base de datos. Cuando se desplaza fuera de la fila en la que está escribiendo los datos, los datos de la fila completa se guardan automáticamente en la base de datos.

Obtener la información necesaria: conectarse a una base de datos existente

En esta lección, obtendrá información sobre cómo conectar su programa a una base de datos existente.

La conexión a una base de datos existente es un procedimiento muy sencillo. Puede utilizar las herramientas visuales de Visual Basic Express para explorar la base de datos y agregar una copia local al proyecto. En esta lección, creará un nuevo proyecto y lo conectará a la base de datos Addresses que creó en la lección anterior.

¡Inténtelo!

Para conectar a una base de datos existente

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba Addresses y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el Explorador de soluciones, haga clic en la ficha Orígenes de datos .

En la ventana Orígenes de datos, seleccione Agregar nuevo origen de datos.

Se abrirá el Asistente para la configuración de orígenes de datos.

Seleccione **Base de datos** y, a continuación, haga clic en **Siguiente**.

Haga clic en el botón **Nueva conexión**.

Se abrirá el cuadro de diálogo **Agregar conexión**.

En el cuadro de diálogo Agregar conexión, si Origen de datos no es Archivo de base de datos de Microsoft SQL Server (cliente SQL), haga clic en el botón Cambiar y seleccione Archivo de base de datos de Microsoft SQL, en el cuadro de diálogo Cambiar origen de datos. Haga clic en Aceptar.

Haga clic en el botón **Examinar**, desplácese a la ubicación donde guardó la base de datos `FirstDatabase.mdf` y haga clic en **Abrir**.

Haga clic en **Aceptar** para cerrar el cuadro de diálogo y, a continuación, en el **Asistente para la configuración de orígenes de datos**, haga clic en **Siguiente**. Si se le pide copiar el archivo de datos en su proyecto, haga clic en **Sí**.

En la página siguiente del asistente, estará seleccionada la casilla de verificación **Sí, guardar la conexión como**. Haga clic en **Siguiente** para continuar.

En la página **Elija los objetos de base de datos**, expanda el nodo **Tablas** y, a continuación, active la casilla de verificación de la tabla **Addresses**.

Haga clic en **Finalizar** para finalizar.

Se ha agregado un archivo de base de datos local a su proyecto. Observe que se ha agregado un objeto `FirstDatabaseDataSet` a la ventana **Orígenes de datos**.

En el menú **Archivo**, elija **Guardar todo** para guardar el proyecto.

Mostrar información al usuario: mostrar datos en la interfaz de usuario

En esta lección, obtendrá información sobre cómo crear una interfaz de usuario básica para ver los datos en su base de datos local.

Ahora que ha creado una conexión a un archivo de base de datos local, el paso siguiente es crear una interfaz de usuario para mostrar los datos. La programación subyacente a la recuperación de los datos de una base de datos y mostrarlos en una interfaz de usuario es muy complicada. Afortunadamente, Visual Basic Expresscrea y configura automáticamente los objetos de datos

necesarios para usted, por lo que todo lo que necesita hacer es seleccionar y colocar los objetos. En esta lección, obtendrá información sobre cómo crear un sencillo formulario de visualización de datos.

¡Inténtelo!

Para crear un formulario de acceso a datos

Abra el proyecto `Addresses` de la lección anterior. Si no lo ha finalizado todavía, vaya a Obtener la información necesaria: conectarse a una base de datos existente y finalice la lección antes de continuar.

En el **Explorador de soluciones**, seleccione **Form1.vb** y, a continuación, en el menú **Ver** elija **Diseñador**.

En el Explorador de soluciones, haga clic en la ficha Orígenes de datos .

En la ventana **Orígenes de datos**, explore sin prisas los nodos **FirstDatabaseDataSet** y **Addresses**. Puede expandir el nodo **Addresses** para ver todos los campos individuales de la tabla.

Arrastre el nodo **Addresses** desde la ventana **Orígenes de datos** al formulario.

> **Nota**
>
> Algunos controles se agregan automáticamente al formulario, además se crean y se agregan varios componentes a la bandeja de componentes debajo del formulario. Hay un control DataGridView que mostrará las filas y columnas de la tabla y un control para la exploración (AddressesBindingNavigator). Asimismo, Visual Basic Expresscrea componentes que se conectan a la base de datos, administran la recuperación y actualización de datos y almacenan los datos en un DataSet local (AddressesBindingSource, AddressesTableAdapter y FirstDatabaseDataSet, respectivamente).

Seleccione el control **AddressesDataGridView** y en la ventana **Propiedades**, establezca la propiedad **Dock** en **Fill** (haga clic en el botón central).

De esta forma, se expandirá la cuadrícula para rellenar el formulario.

Presione F5 para ejecutar el programa.

Los datos de la tabla `Addresses` se muestran en el control **DataGridView** en el formulario. Puede utilizar los controles en **BindingNavigator** para desplazarse entre las filas e incluso agregar o eliminar registros. También puede realizar cambios en los registros modificando los datos mostrados en la cuadrícula, pero estos cambios no se guardarán a menos que haga clic en el icono **Guardar datos**. En el tema siguiente, obtendrá información sobre cómo guardar los cambios automáticamente en el conjunto de datos.

Agregar o modificar registros: actualizar datos

En esta lección, obtendrá información sobre cómo crear un formulario de entrada de datos para actualizar los datos de la base de datos local.

En las últimas tres lecciones, ha creado una base de datos, ha agregado un archivo de base de datos al proyecto y ha creado una interfaz de usuario básica. Como ha podido observar, pudo hacer cambios en los datos de las direcciones e incluso agregar nuevos registros, pero si cerró el programa y lo inició de nuevo, se han perdido esos cambios.

En realidad estos datos eran una copia de los datos de la base de datos, almacenados en un **DataSet** local. Cada vez que el programa se inicia, **DataSet** recupera sus datos de la base de datos. Cuando los cambios se realizan en el **DataSet**, no se realizan en la base de datos.

Si hace clic en el botón **Guardar** en el control **AddressesBindingNavigator**, todos los cambios se copian del **DataSet** a la base de datos. Como es probable que el usuario no siempre recuerde guardar el trabajo, agregue código para guardar los cambios automáticamente a la base de datos al cerrar el programa. Mientras esté en él, también podrá cambiar la interfaz de usuario para facilitar la entrada de datos.

¡Inténtelo!

Para actualizar su archivo de base de datos local

Abra el proyecto `Addresses` de la lección anterior. Si no ha completado todavía la lección anterior, vaya a Mostrar información al usuario: mostrar datos en la interfaz de usuario y complete los pasos.

En el **Explorador de soluciones**, seleccione **Form1** y, a continuación, en el menú **Ver** elija **Diseñador**.

En el formulario, seleccione el control **AddressesDatGridView** y elimínelo.

En el Explorador de soluciones, haga clic en la ficha Orígenes de datos .

En la ventana **Orígenes de datos**, seleccione la tabla **Direcciones** y, a continuación, seleccione **Detalles** en la lista desplegable.

Arrastre el nodo **Addresses** desde la ventana **Orígenes de datos** al nuevo formulario.

Se agregan controles **TextBox** por cada campo de la tabla, junto con los controles **Label** que describen los campos.

Haga doble clic en el formulario para abrir el Editor de código.

En la lista desplegable **Eventos**, haga clic en **FormClosing**.

En el controlador de eventos **Form1_FormClosing**, escriba el siguiente código:

Copiar código

```
Me.AddressesBindingSource.EndEdit()
Me.AddressesTableAdapter.Update(Me.FirstDatabaseDataSet.Addresses)
```

Este código hace que **AddressesTableAdapter** copie cualquier cambio del conjunto de datos a la base de datos local.

Presione F5 para ejecutar el programa.

Realice cambios en parte de los datos o agregue un nuevo registro y, a continuación, cierre el formulario.

Presione F5 de nuevo. Los cambios deben haberse guardado.

En este conjunto de lecciones, obtuvo información sobre cómo crear una base de datos y un programa para tener acceso a la base de datos. En el conjunto siguiente de lecciones, obtendrá información sobre clases, las guías para objetos que puede reutilizar en sus programas

Programar con objetos: utilizar clases

Como aprendió en una lección anterior, los programas de Visual Basic Expressse generan con objetos como formularios y controles. Los objetos también pueden representar cosas reales como una persona, un equipo, o incluso algo más abstracto como una cuenta bancaria.

Una *clase* es simplemente una representación de un tipo de objeto; piense en él como el plano del objeto. Así como un solo plano puede utilizarse para generar varios edificios, una clase puede utilizarse para crear múltiples copias de un objeto.

En las lecciones siguientes, aprenderá a utilizar las clases en los programas de Visual Basic.

¿Qué es una clase?

En esta lección, aprenderá a utilizar clases para representar objetos en sus programas.

Como aprendió en una lección anterior, los programas de Visual Basic Expressse crean con objetos como formularios o controles. Los objetos también

se pueden utilizar para representar cosas reales como personas, equipos informáticos o incluso algo más abstracto, como una cuenta bancaria.

Una *clase* es simplemente una representación de un tipo de objeto; piense en ella como un plano que describe el objeto. Así como un plano puede utilizarse para construir varios edificios, una clase puede utilizarse para crear varias copias de un objeto.

Aunque puede que no se haya dado cuenta, ya ha utilizado las clases. Por ejemplo, el control **TextBox** lo define una clase **TextBox**, que define su aspecto y sus funciones. Cada vez que arrastra un control **TextBox** a un formulario, realmente está creando una nueva *instancia* de la clase **TextBox**.

Cada control **TextBox** es una copia exacta, aunque distinta, de la clase que lo define, la clase TextBox. Puesto que cada objeto es una "instancia" independiente de una clase, la acción de crear una clase se denomina *creación de instancias*.

Hasta ahora ha agregado los controles **TextBox** a su formulario arrastrándolos desde el **Cuadro de herramientas**, pero también puede crear instancias de un objeto **TextBox** en su código si utiliza la palabra clave **New**.

Visual Basic Express | Copiar código

```
Dim Textbox1 As New TextBox
```

Obtendrá más información sobre crear y utilizar las clases en las lecciones siguientes.

¿Qué hay dentro de una clase?

En una lección anterior, Información detallada: comprender propiedades, métodos y eventos, aprendió que todos los objetos tienen propiedades que describen sus atributos, métodos que definen sus acciones y eventos que definen sus respuestas. Igualmente, la clase que define un objeto tiene sus propias propiedades, métodos y eventos (a veces llamados *miembros*) que se pasan a todas las instancias de esa clase.

Por ejemplo, una clase que representa una cuenta bancaria podría tener propiedades como `AccountNumber` o `AccountBalance`, métodos como `CalculateInterest` y eventos como `BalanceChanged`. Una vez creada la instancia de un objeto de cuenta bancaria, puede tener acceso a sus propiedades, métodos y eventos de igual manera que si se tratara de un objeto **TextBox**.

Algunos miembros de una clase son privados; sólo se tiene acceso a ellos mediante código dentro de la clase. Por ejemplo, una clase de cuenta bancaria

puede tener un método para calcular un saldo. Lo lógico es permitir que un programa lea ese balance pero no que pueda cambiarlo directamente.

Puede ocultar los miembros de una clase si los declara como **Private** o exponerlos si los declara como **Public**. También puede permitir el acceso a una propiedad y a la vez impedir que el programa cambie su valor declarándolo como **ReadOnly**. El código siguiente muestra cómo podría ser una clase `BankAccount`.

Visual Basic Express | Copiar código

```
Class BankAccount
    Private AccountNumber As String
    Private AccountBalance As Decimal
    Public Sub UpdateBalance()
        ' add code to recalculate balance.
    End Sub
    ReadOnly Property Balance() As Decimal
        Get
            Return AccountBalance
        End Get
    End Property
End Class
```

Modelar un objeto en una situación real: crear la primera clase

En esta lección, aprenderá a crear una clase mediante un proyecto de **bibliotecas de clase**.

En la lección anterior, aprendió que las clases se pueden utilizar como un plano que modela objetos del mundo real. Una de las mejores razones para utilizar clases es que una vez que ha creado una clase para cierto tipo de objeto, puede reutilizar esa clase en cualquier proyecto.

Por ejemplo, muchos de los programas que escribe pueden involucrar personas: un programa de libreta de direcciones para mantener seguimiento de amigos, un programa de administrador de contactos para los contactos comerciales o un programa para realizar un seguimiento de empleados.

Aunque los programas pueden ser considerablemente diferentes, los atributos que se aplican a una persona serían los mismos. Cada persona tiene nombre, edad, dirección y número de teléfono.

En esta lección y las siguientes creará una clase que representa una persona; puede guardar esta clase y utilizarla en otros programas que escriba en el futuro.

Las clases se pueden crear de tres maneras: como parte del código en un módulo de formulario en un proyecto de **aplicación para Windows**, como un módulo de clase separado agregado a un proyecto de **aplicación para Windows** o como un proyecto de **bibliotecas de clase** independiente.

Crear clases

Habrá observado que en algunas de las lecciones anteriores al hacer doble clic en un formulario y abrir el Editor de código se veía algo parecido a lo siguiente.

Copiar código

```
Public Class Form1
    Private Sub Form1_Load...


    End Sub
End Class
```

Correcto, el formulario realmente es una clase, marcada por instrucciones **Class** y **End Class** y cualquier código que se haya escrito entre las dos instrucciones es parte de la clase. Aunque de manera predeterminada un módulo de formulario contiene sólo una clase única, puede crear módulos adicionales agregando código debajo de la instrucción **End Class**, tal como se ilustra a continuación:

Copiar código

```
Public Class Form1
    ' Form1 code here
End Class
Public Class MyFirstClass
    ' Your class code here
End Class
```

La desventaja de crear clases de esta manera es que sólo están disponibles dentro del proyecto donde se crearon. Si desea compartir una clase con otros proyectos, puede colocarla en un módulo de clase.

Módulos de clase

Un módulo de clase es un archivo de código separado, que contiene una o más clases. Como es un archivo independiente, se puede reutilizar en otros proyectos. Los módulos de clase se pueden crear de dos maneras: como un módulo agregado a un proyecto de **aplicación para Windows** o como un proyecto de **bibliotecas de clase** independiente.

Puede agregar un nuevo módulo de clase a un proyecto existente seleccionando **Clase** en el cuadro de diálogo **Agregar nuevo elemento**, disponible en el menú **Proyecto**. Para trabajar en esta unidad de lecciones, creará un proyecto de **bibliotecas de clase** independiente.

Inténtelo

Para crear un proyecto de biblioteca de clases

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantillas**, del cuadro de diálogo **Nuevo proyecto**, haga clic en **Biblioteca de clases**.

En el cuadro **Nombre**, escriba `Persons` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de **bibliotecas de clase** y el Editor de código mostrará el módulo de clase `Class1.vb`.

En el **Explorador de soluciones**, haga clic con el botón secundario del mouse en `Class1.vb` y seleccione **Cambiar nombre** y, a continuación, cambie el nombre a `Persons.vb`.

Observe que el nombre en el Editor de código también cambia a `Persons.vb`.

En el menú **Archivo**, elija **Guardar todo**.

En el cuadro de diálogo **Guardar proyecto**, haga clic en **Guardar**.

> Sugerencia
>
> En lugar de guardar el proyecto en la ubicación predeterminada, podría crear un directorio en el cual puede almacenar todas sus clases para reutilizarlas. Puede especificar esa carpeta en el campo **Location** del cuadro de diálogo **Guardar proyecto** antes de guardar.

De momento, mantenga el proyecto abierto, lo utilizará en la lección siguiente.

Agregar propiedades a una clase

En esta lección, aprenderá a agregar propiedades a la clase que creó en la lección anterior.

En una lección anterior, Información detallada: comprender propiedades, métodos y eventos, aprendió que todos los objetos tienen atributos y que las propiedades representan atributos. En esa lección creó una clase `Persons` que representa una persona; las personas tienen atributos como el nombre y la edad, por lo que la clase `Persons` necesita propiedades que representen dichos atributos.

Se pueden agregar propiedades a una clase de dos maneras: como *campo* o como *procedimiento de propiedad*. También puede determinar cómo funciona una propiedad utilizando los modificadores **Public**, **ReadOnly** o **WriteOnly**.

Campos y procedimientos de propiedad

Los campos son variables públicas dentro de una clase que se pueden establecer o leer desde fuera de la clase. Resultan de utilidad para propiedades que no se tienen que validar, por ejemplo, un valor **Boolean** (**True** o **False**). En el caso de la clase `Persons`, se puede tener una propiedad **Boolean** denominada `Alive`, que especifica si una persona está viva o muerta. Puesto que hay sólo dos valores posibles, un campo funciona bien para esta propiedad.

Para agregar un campo a una clase, el código podría ser como el que sigue.

Visual Basic Express | Copiar código

```
Public Alive As Boolean
```

La mayoría de las propiedades, sin embargo, son más complejas; en la mayor parte de los casos deseará utilizar un procedimiento de propiedad para agregar una propiedad a una clase. Los procedimientos de propiedad tienen tres partes: una declaración de una variable privada para almacenar el valor de la propiedad; un procedimiento **Get** que expone el valor; y un procedimiento **Set** que, como indica su nombre, establece el valor.

Por ejemplo, un procedimiento de propiedad para una propiedad `Name` de la clase `Persons` podría ser como el que sigue.

Visual Basic Express | Copiar código

```
Private nameValue As String
Public Property Name() As String
```

```
    Get
        Name = nameValue
    End Get
    Set(ByVal value As String)
        nameValue = value
    End Set
End Property
```

La primera línea de código declara una variable **String** privada, `nameValue` que almacenará el valor de la propiedad. El procedimiento de propiedad en sí comienza con `Public Property` y termina con `End Property`.

El procedimiento **Get** contiene el código que se ejecutará cuando desee leer su valor; por ejemplo, si lee la propiedad `Persons.Name`, el código devolverá el valor almacenado en la variable `nameValue`.

El procedimiento **Set** contiene código que se utiliza para asignar un nuevo valor a la variable `nameValue` usando un valor pasado como argumento `value`. Por ejemplo, si escribió el código `Persons.Name = "John"`, el valor **String** `John` se pasará como argumento `value`; el código del procedimiento **Set** lo asignará a la variable `NameValue` para su almacenamiento.

Se preguntará por qué complicarse tanto en lugar de utilizar un campo que represente la propiedad `Name`. En el mundo real, hay ciertas reglas para los nombres: por ejemplo, los nombres normalmente no contienen números. Puede agregar código al procedimiento **Set** para comprobar el argumento `value` y devolver un error si contiene números.

En el siguiente procedimiento, se agregará un campo y tres propiedades a la clase `Persons`.

Inténtelo

Para agregar propiedades a la clase

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Modelar un objeto en una situación real: crear la primera clase, y realizar hasta el final los procedimientos de esa lección.

En el **Explorador de soluciones**, seleccione **Persons.vb** y en el menú **Ver** seleccione **Código**.

Agregue el siguiente código de declaración debajo de la línea `Public Class Persons`.

Visual Basic Express — Copiar código

```
Private firstNameValue As String
Private middleNameValue As String
Private lastNameValue As String
Public Alive As Boolean
```

Agregue los siguientes procedimientos de propiedad debajo del código de declaración.

Visual Basic Express — Copiar código

```
Public Property FirstName() As String
    Get
        FirstName = firstNameValue
    End Get
    Set(ByVal value As String)
        firstNameValue = value
    End Set
End Property

Public Property MiddleName() As String
    Get
        MiddleName = middleNameValue
    End Get
    Set(ByVal value As String)
        middleNameValue = value
    End Set
End Property

Public Property LastName() As String
    Get
```

```
        LastName = lastNameValue
    End Get
    Set(ByVal value As String)
        lastNameValue = value
    End Set
End Property
```

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Propiedades de sólo lectura y escritura

A veces una propiedad se establece una vez y no cambia nunca durante la ejecución del programa. Por ejemplo, una propiedad que representa un número de empleado nunca debe cambiar, de modo que otro programa lo puede leer, pero no se permitirá que ese programa cambie su valor.

La palabra clave **ReadOnly** se utiliza para especificar que un valor de propiedad se puede leer pero no modificar. Si intenta asignar un valor a una propiedad **ReadOnly** , aparecerá un error en el Editor de código.

Para crear una propiedad de sólo lectura, deberá crearse un procedimiento de propiedad con un procedimiento **Get**, pero sin procedimiento **Set**, tal como se muestra a continuación.

Visual Basic Express — Copiar código

```
Private IDValue As Integer
ReadOnly Property ID() As Integer
    Get
        ID = IDValue
    End Get
End Property
```

De igual forma, la palabra clave **WriteOnly** permite establecer un valor de propiedad pero no permite que se lea; por ejemplo, no permite que otros programas lean una propiedad de contraseña. Puede utilizar ese valor para realizar acciones dentro de la clase, pero deseará que siga siendo privado.

Para crear una propiedad de sólo escritura, se creará una propiedad con un procedimiento **Set** pero sin procedimiento **Get**, tal como se muestra a continuación.

Visual Basic Express Copiar código

```
Private passwordValue As String
WriteOnly Property Password() As String
    Set(ByVal value As String)
        passwordValue = value
    End Set
End Property
```

Los procedimientos de propiedad **ReadOnly** y **WriteOnly** también son útiles cuando se desea tomar un valor de propiedad y convertirlo en un valor diferente. Por ejemplo, pensemos en la edad de una persona. A diferencia del nombre, la edad cambia con el tiempo, si ha asignado la edad a una clase y la lee de nuevo un año después, sería incorrecta.

En la clase `Persons`, puede evitarlo agregando dos propiedades: una propiedad **WriteOnly** `BirthYear` que representa el año de nacimiento, que nunca cambia, y una propiedad **ReadOnly** `Age` que devuelve un valor calculando la diferencia entre el año en curso y el año de nacimiento.

Inténtelo

Para agregar propiedades ReadOnly y WriteOnly a la clase

Agregue el siguiente código de declaración debajo de las otras declaraciones en la parte superior del módulo de clase.

Visual Basic Express Copiar código

```
Private birthYearValue As Integer
```

Agregue los siguientes procedimientos de propiedad debajo del código de declaración.

Visual Basic Express Copiar código

```
WriteOnly Property BirthYear() As Integer
    Set(ByVal value As Integer)
        birthYearValue = value
    End Set
End Property
```

```
ReadOnly Property Age() As String
    Get
        Age = My.Computer.Clock.LocalTime.Year - birthYearValue
    End Get
End Property
```

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Agregar métodos a una clase

En esta lección, aprenderá a agregar métodos a una clase para que pueda realizar acciones.

En una lección anterior, Información detallada: comprender propiedades, métodos y eventos, aprendió que la mayoría de los objetos tiene acciones que puede realizar; estas acciones se conocen como métodos. La clase `Persons` que creó en la lección Modelar un objeto en una situación real: crear la primera clase representa a una persona. Hay muchas acciones que pueden realizar las personas y para la clase `Persons`, esas acciones se pueden expresar como métodos de clase.

Métodos de una clase

Los métodos de una clase son simplemente procedimientos **Sub** o **Function** declarados dentro de la clase. Por ejemplo, una clase `Account` puede tener un procedimiento **Sub** denominado `Recalculate`, que actualizará el balance o un procedimiento **Function** denominado `CurrentBalance` para devolver el último balance. El código para declarar esos métodos puede ser similar al siguiente.

Visual Basic Express — Copiar código

```
Public Sub Recalculate()
    ' add code to recalculate the account.
End Sub
Public Function CurrentBalance(ByVal AccountNumber As Integer) As Double
    ' add code to return a balance.
End Function
```

Si bien la mayoría de los métodos de clase son públicos, también se pueden agregar métodos que sólo la clase en sí puede utilizar. Por ejemplo, la clase `Persons` puede tener su propia función para calcular la edad de una persona. Al declarar la función como **Private**, no se puede ver o llamar desde fuera de la clase.

El código para una función privada puede ser similar al siguiente:

Visual Basic Express — Copiar código

```
Private Function CalcAge(ByVal year As Integer) As Integer
    CalcAge = My.Computer.Clock.LocalTime.Year - year
End Function
```

Más tarde puede cambiar el código que calcula el valor `CalcAge` y el método seguirá funcionando bien sin cambiar ningún código que utilice el método. Ocultar el código que realiza el método se conoce como *encapsulación*.

En la clase `Persons`, se creará un método público que devuelve un nombre completo y una función privada para calcular la edad.

Inténtelo

Para agregar un método a la clase

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Agregar propiedades a una clase y finalizar los procedimientos.

En el **Explorador de soluciones**, seleccione **Persons.vb** y, en el menú **Ver**, elija **Código**.

Agregue el siguiente código a continuación de los procedimientos de propiedad.

Visual Basic Express — Copiar código

```
Public Function FullName() As String
    If middleNameValue <> "" Then
        FullName = firstNameValue & " " & middleNameValue & " " _
                            & lastNameValue
    Else
        FullName = firstNameValue & " " & lastNameValue
    End If
```

```
End Function
```

Visual Basic Express — Copiar código

```
Private Function CalcAge(ByVal year As Integer) As Integer
    CalcAge = My.Computer.Clock.LocalTime.Year - year
End Function
```

Modifique el código en el procedimiento de la propiedad `Age` para utilizar la función privada.

Visual Basic Express — Copiar código

```
ReadOnly Property Age() As String
    Get
        ' Age = My.Computer.Clock.LocalTime.Year - birthDateValue
        Age = CalcAge(birthYearValue)
    End Get
End Property
```

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Pasos siguientes

En esta lección, aprendió a agregar métodos públicos y privados a una clase. Puede obtener más información sobre los métodos en Información detallada: crear varias versiones del mismo método con sobrecarga o puede continuar con la siguiente lección y aprender a utilizar y probar la clase que creó.

Información detallada: crear varias versiones del mismo método con sobrecarga

En esta lección, aprenderá a agregar a la clase varias versiones de un método.

En la lección anterior, aprendió a agregar métodos a la clase `Persons`. A veces hay casos en los que un método único no sirve; por ejemplo, es probable que deba pasar diferentes tipos de datos al método en distintas situaciones o quizá desee devolver formatos diferentes como resultado.

Se pueden crear varias versiones de un método mediante una técnica llamada *sobrecarga*. Cuando una clase tiene más de un método con el mismo nombre pero con un conjunto de argumentos diferente, el método se sobrecarga.

Sobrecarga

Para crear un método sobrecargado, agregue dos o más procedimientos **Sub** o **Function** a la clase, cada uno con el mismo nombre. En las declaraciones de procedimiento, el conjunto de argumentos para cada procedimiento debe ser distinto o se producirá un error.

El siguiente ejemplo muestra un método con dos sobrecargas, una que acepta una **String** y la otra que acepta un **Integer** como argumento.

Visual Basic Express — Copiar código

```
Public Sub TestFunction(ByVal input As String)
    MsgBox(input)
End Sub
Public Sub TestFunction(ByVal input As Integer)
    MsgBox(CStr(input))
End Sub
```

Si se debe llamar a este método desde el código y pasarle una cadena, se ejecutaría la primera sobrecarga y un cuadro de mensaje mostraría la cadena; si se le pasó un número, se ejecutaría la segunda sobrecarga y el número se convertiría en una cadena y aparecería en el cuadro de mensaje.

Puede crear tantas sobrecargas como sea necesario y cada una de ellas puede contener un número diferente de argumentos.

En la clase `Persons`, se agregará un método con dos sobrecargas para devolver la inicial del segundo nombre de una persona; una sólo con la inicial y la otra con la inicial seguida por un punto.

Inténtelo

Para crear un método sobrecargado

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, regrese a la lección anterior, Agregar métodos a una clase y finalice los procedimientos.

En el **Explorador de soluciones**, seleccione **Persons.vb** y, en el menú **Ver**, elija **Código**.

Agregue el siguiente código debajo de los métodos existentes.

Visual Basic Express | Copiar código

```
Public Function MiddleInitial() As String
    MiddleInitial = Left$(middleNameValue, 1)
End Function

Public Function MiddleInitial(ByVal period As Boolean) As String
    MiddleInitial = Left$(middleNameValue, 1) & "."
End Function
```

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Probar una clase

En esta lección, aprenderá a crear una instancia de una clase para probar la clase.

En lecciones anteriores, creó una clase `Persons` y le otorgó propiedades y métodos. Lo que ha hecho hasta ahora es agregar código, ahora es el momento de utilizar la clase `Persons` y asegurarse de que funcione según lo esperado.

Crear una instancia de una clase

Aunque es posible que no se haya dado cuenta, ha estado utilizando clases en muchas de las lecciones anteriores. Los formularios y controles son en realidad clases; cuando arrastra un control **Button** a un formulario, está creando realmente una instancia de la clase Button.

También puede crear instancias de cualquier clase en el código utilizando una declaración con la palabra clave **New**. Por ejemplo, para crear una nueva instancia de la clase **Button**, agregará el código siguiente.

Visual Basic Express | Copiar código

```
Dim aButton As New Button
```

Para utilizar y probar la clase `Persons`, debe crear primero un proyecto de prueba y agregar una referencia al módulo de clase.

Inténtelo

Para crear un proyecto de prueba para la clase

Abra el proyecto Persons que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Agregar métodos a una clase y finalizar los procedimientos.

En el menú **Archivo**, elija **Agregar** y seleccione **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `PersonsTest` y haga clic en **Aceptar**.

En el Explorador de soluciones, seleccione el proyecto `PersonsTest` y en el menú Proyecto, elija Establecer como proyecto de inicio.

En el **Explorador de soluciones**, seleccione el proyecto `PersonsTest` y en el menú **Proyecto**, elija **Agregar referencia**.

Se abrirá el cuadro de diálogo Agregar referencia.

Haga clic en la ficha **Proyectos**, seleccione **Personas** y haga clic en **Aceptar**.

Haga doble clic en el formulario para abrir el Editor de código y escriba la siguiente declaración justo debajo de la línea `Public Class Form1`.

Visual Basic Express — Copiar código

```
Dim person1 As New Persons.Persons
```

Esto declara una nueva instancia de la clase Persons. Se preguntará por qué tuvo que escribir dos veces Persons: la primera instancia es el módulo de clase Persons.vb; la segunda instancia es la clase Persons dentro de ese módulo.

En el menú **Archivo**, elija **Guardar todo**.

Probar una clase

El siguiente paso es agregar una interfaz de usuario y un código que utilice la clase `Persons`. Agregará cuadros de texto donde el usuario especificará los valores para cada una de las propiedades (excepto la propiedad de sólo lectura `Age`), una casilla de verificación para el campo `Alive` y botones para probar cada uno de los métodos públicos.

Inténtelo

Para probar la clase

En el **Explorador de soluciones**, seleccione Form1 y en el menú **Ver**, seleccione **Diseñador**.

En el **Cuadro de herramientas**, arrastre cuatro controles **TextBox**, un control **CheckBox** y dos controles **Button** al formulario.

Seleccione el primer control Button y en la ventana Propiedades establezca su propiedad Text en `Update` .

Seleccione el segundo control Button y en la ventana Propiedades establezca su propiedad Text en `Full Name` .

Haga doble clic en el primer botón (**Update**) para abrir el Editor de código y en el controlador de eventos **Button1_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
With person1
    .FirstName = Textbox1.Text
    .MiddleName = Textbox2.Text
    .LastName = Textbox3.Text
    .BirthYear = Textbox4.Text
    .Alive = CheckBox1.Checked
End With
```

Observe que cuando escribe, se muestra una lista que contiene todos los miembros de la clase `Persons`. Puesto que se agregó como una referencia, IntelliSense muestra la información sobre la clase tal como lo haría para cualquier otra clase.

En el controlador de eventos **Button2_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
' Test the FullName method.
MsgBox(person1.FullName)


' test the Age property and CalcAge method.
MsgBox(CStr(person1.Age) & " years old")


' Test the Alive property.
```

```
If person1.Alive = True Then
    MsgBox(person1.FirstName & " is alive")
Else
    MsgBox(person1.FirstName & " is no longer with us")
End If
```

Presione F5 para ejecutar el proyecto y mostrar el formulario.

En el primer cuadro de texto, escriba su nombre.

En el segundo cuadro de texto, escriba su segundo nombre.

En el tercer cuadro de texto, escriba su apellido.

En el cuarto cuadro de texto, escriba el año de cuatro dígitos en el que nació (por ejemplo, 1983).

Después, active la casilla de verificación.

Haga clic en el botón **Actualizar** para establecer las propiedades de la clase y haga clic en el botón **Full Name**.

Se mostrarán tres cuadros de mensaje, con su nombre completo, edad y estado.

En el menú **Archivo**, elija **Guardar todo**.

Probar los métodos sobrecargados

Si finalizó la lección opcional Información detallada: crear varias versiones del mismo método con sobrecarga, también deseará probar los métodos sobrecargados que agregó a la clase `Persons`. Si no finalizó la lección, puede regresar y hacerlo ahora o puede omitir el siguiente procedimiento.

Inténtelo

Para probar los métodos sobrecargados

En el **Explorador de soluciones**, seleccione `Form1` y en el menú **Ver**, seleccione **Diseñador**.

En el **Cuadro de herramientas**, arrastre dos controles más **Button** al formulario.

Seleccione el tercer control **Button** y en la ventana **Propiedades** establezca su propiedad **Text** en `With`.

Seleccione el cuarto control **Button** y en la ventana **Propiedades** establezca su propiedad **Text** en `Without`.

Haga doble clic en el primer botón (**With**) para abrir el Editor de código y escriba el siguiente código en el controlador de eventos **Button3_Click**.

Visual Basic Express — Copiar código

```
MsgBox(person1.FirstName & " " & person1.MiddleInitial(True) & _
                    " " & person1.LastName)
```

Observe que cuando escribe, se muestra una lista que contiene todos los miembros de la clase `Persons`. Puesto que se agregó como una referencia, IntelliSense muestra la información sobre la clase tal como lo haría para cualquier otra clase.

En el controlador de eventos **Button4_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
MsgBox(person1.FirstName & " " & person1.MiddleInitial & _
                    " " & person1.LastName)
```

Presione F5 para ejecutar el proyecto y mostrar el formulario.

En el primer cuadro de texto, escriba su nombre.

En el segundo cuadro de texto, escriba su segundo nombre.

En el tercer cuadro de texto, escriba su apellido.

En el cuarto cuadro de texto, escriba el año de cuatro dígitos en el que nació (por ejemplo, 1983).

Después, active la casilla de verificación.

Haga clic en el botón **Actualizar** para establecer las propiedades de la clase y haga clic en el botón **With**.

Aparecerá un cuadro de mensaje que muestra su nombre con un punto después de la inicial del segundo nombre

Haga clic en el botón **Without**.

Aparecerá un cuadro de mensaje que muestra su nombre sin un punto después de la inicial del segundo nombre

En el menú **Archivo**, elija **Guardar todo**.

Generar una clase en una clase existente: utilización de la herencia

En esta lección, aprenderá a utilizar la herencia para crear una clase basada en una clase existente.

Muchos objetos de la vida real tienen atributos y comportamientos en común, por ejemplo, todos los automóviles tienen ruedas y motores, y pueden avanzar y detenerse (es de esperar). Sin embargo, algunos automóviles tienen atributos que no son comunes, por ejemplo, un descapotable tiene una parte superior que se puede mover y bajar electrónica o manualmente.

Si se creó un objeto para representar un automóvil, se pueden incluir propiedades y métodos para todos los atributos y comportamientos comunes, pero no se podrían agregar atributos como la cubierta de un descapotable, puesto que dicho atributo no es generalizable a todos los automóviles.

Mediante el uso de la *herencia*, se puede crear una clase "descapotable" que deriva de la clase automóvil. Ésta hereda todos los atributos de la clase automóvil y puede agregar los atributos y comportamientos que son únicos de un auto descapotable.

Heredar a partir de una clase existente

La instrucción **Inherits** se utiliza para declarar una nueva clase, denominada *clase derivada*, basada en una clase existente conocida como *clase base*. Las clases derivadas heredan todas las propiedades, los métodos, los eventos, los campos y las constantes definidos en la clase base. El siguiente código muestra la declaración para una clase derivada.

Visual Basic Express — Copiar código

```
Class DerivedClass
    Inherits BaseClass
End Class
```

Se pueden crear instancias de la nueva clase `DerivedClass`, se puede tener acceso a sus propiedades y métodos como `BaseClass` y se pueden agregar nuevas propiedades y métodos que son específicos de la nueva clase. Para ver un ejemplo, observe la clase `Persons` que creó en las lecciones anteriores.

Suponga que desea una clase que represente jugadores de béisbol: los jugadores del béisbol tienen todos los atributos definidos en la clase `Persons`, pero también tienen atributos únicos, como su número y posición. En lugar de agregar esas propiedades a la clase `Persons`, se creará una nueva clase

derivada que se hereda de `Persons`, a la que se agregan las nuevas propiedades.

Inténtelo

Para crear una clase derivada

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, regrese a Probar una clase y finalice los procedimientos.

En el **Explorador de soluciones**, seleccione el nodo del proyecto **Persons**.

En el menú Proyecto, elija Agregar clase.

En el cuadro de diálogo **Agregar nuevo elemento**, escriba `Players` en el cuadro **Nombre,** a continuación, haga clic en **Agregar**.

Se agregará un nuevo módulo de clase al proyecto.

En el Editor de código, agregue lo siguiente justo debajo de la línea `Public Class Players`.

Visual Basic Express — Copiar código

```
Inherits Persons
```

Agregue el siguiente código para definir dos nuevas propiedades.

Visual Basic Express — Copiar código

```
Private numberValue As Integer
Private positionValue As String
Public Property Number() As Integer
    Get
        Number = numberValue
    End Get
    Set(ByVal value As Integer)
        numberValue = value
    End Set
End Property
Public Property Position() As String
    Get
```

```
        Position = positionValue
    End Get
    Set(ByVal value As String)
        positionValue = value
    End Set
End Property
```

En el menú **Archivo**, elija **Guardar todo**.

Probar la clase Players

Habrá creado ahora una clase `Players` derivada de la clase `Persons`. En el procedimiento siguiente, creará un nuevo programa para probar la clase `Players`.

Para crear un proyecto de prueba para la clase

En el menú **Archivo**, elija **Agregar** y, después, seleccione **Nuevo proyecto**.

En el cuadro de diálogo Agregar nuevo proyecto, en el panel Plantillas, seleccione Aplicación para Windows.

En el cuadro **Nombre**, escriba `PlayerTest` y haga clic en **Aceptar**.

Se agregará un nuevo proyecto de formularios Windows Forms al **Explorador de soluciones** y se mostrará un nuevo formulario.

En el Explorador de soluciones, seleccione el proyecto `PlayerTest` y en el menú Proyecto, elija Establecer como proyecto de inicio.

En el **Explorador de soluciones**, seleccione el proyecto `PlayerTest` y en el menú **Proyecto**, elija **Agregar referencia**.

Se abrirá el cuadro de diálogo **Agregar referencia**.

Haga clic en la ficha **Proyectos**, elija **Persons** y haga clic en **Aceptar**.

Haga doble clic en el formulario para abrir el Editor de código y escriba la siguiente declaración justo debajo de la línea `Public Class Form1`.

Visual Basic Express — Copiar código

```
Dim player1 As New Persons.Players
Dim player2 As New Persons.Players
```

Esto declara dos nuevas instancias de la clase `Players`.

En el menú **Archivo**, elija **Guardar todo**.

Para probar la clase derivada

En el **Explorador de soluciones**, seleccione `Form1` en el proyecto `PlayerTest` y en el menú **Ver**, elija **Código**.

En el Editor de código, agregue el siguiente código al procedimiento de evento **Form1_Load**.

Visual Basic Express — Copiar código

```
With player1
    .FirstName = "Andrew"
    .LastName = "Cencini"
    .Number = 43
    .Position = "Shortstop"
End With
With player2
    .FirstName = "Robert"
    .LastName = "Lyon"
    .Number = 11
    .Position = "Catcher"
End With
```

En el **Explorador de soluciones**, seleccione `Form1` en el proyecto `PlayerTest` y en el menú **Ver**, elija **Diseñador**.

En el **Cuadro de herramientas**, arrastre dos controles **Button** al formulario.

Seleccione el primer control **Button** y en la ventana **Propiedades** establezca su propiedad **Text** en `At Bat`.

Seleccione el segundo control **Button** y en la ventana **Propiedades** establezca su propiedad **Text** en `On Deck`.

Haga doble clic en el primer botón (`At Bat`) para abrir el Editor de código y escriba el siguiente código en el controlador de eventos **Button1_Click**.

Visual Basic Express — Copiar código

```
MsgBox(player1.Position & " " & player1.FullName & ", #" & _
  CStr(player1.Number) & " is now at bat.")
```

Observe que está utilizando el método `FullName` que se heredó de la clase base `Persons`.

En el controlador de eventos **Button2_Click**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
MsgBox(player2.Position & " " & player2.FullName & ", #" & _
  CStr(player2.Number) & " is on deck.")
```

Presione F5 para ejecutar el programa. Haga clic en cada botón para ver los resultados.

En el menú **Archivo**, elija **Guardar todo**.

Información detallada: reemplazar miembros

En esta lección, aprenderá a reemplazar un miembro de una clase derivada.

En la lección anterior, aprendió a heredar de una clase base y a extender la clase derivada con nuevas propiedades. Además de agregar nuevas propiedades o métodos a una clase derivada, también es posible que desee cambiar, o *reemplazar*, el comportamiento de propiedades o métodos existentes.

Por ejemplo, podría crear una clase `Truck` que se derive de una clase `Car` con un método `StartEngine`. Si el objeto Truck tiene un motor diesel, el proceso de encendido del motor puede ser distinto al de un objeto Car; en este caso, puede que desee reemplazar el método `StartEngine` para que se adapte mejor al objeto Truck.

Reemplazar propiedades y métodos

De manera predeterminada, no se pueden reemplazar las propiedades ni los métodos en una clase. Para permitir que una clase derivada reemplace una propiedad o un método, se debe marcar como *reemplazable* declarándolo con la palabra clave **Overridable**.

```
Public Overridable Property EngineType As String
```

```
Public Overridable Sub StartEngine(ByVal EngineType As String)
```

Al heredar de una clase base, las propiedades y los métodos que están marcados como **Overridable** se pueden utilizar tal como están o se pueden modificar para satisfacer las necesidades del usuario declarándolos con la palabra clave **Overrides**.

```
Public Overrides Property EngineType As String

Public Overrides Sub StartEngine(ByVal EngineType As String)
```

En la clase `Players` que se creó en la lección anterior, puede que se desee reemplazar el método `FullName` para incluir el número del jugador y eliminar el código que devuelve un segundo nombre.

Inténtelo

Para reemplazar el método FullName

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, regrese a la lección anterior, Generar una clase en una clase existente: utilización de la herencia y complete los procedimientos.

En el **Explorador de soluciones**, seleccione el nodo **Persons.vb** y, en el menú **Ver**, elija **Código**.

En el Editor de código, modifique la declaración del método `FullName` de la siguiente manera.

Visual Basic Express — Copiar código

```
Public Overridable Function FullName() As String
```

En el **Explorador de soluciones**, seleccione el nodo **Players.vb** y, en el menú **Ver**, elija **Código**.

En el Editor de código, agregue el siguiente código a la clase.

Visual Basic Express — Copiar código

```
Public Overrides Function FullName() As String
    FullName = FirstName & " " & LastName & ", #" & numberValue
End Function
```

En el **Explorador de soluciones**, seleccione el nodo **Form1.vb** en el proyecto **PlayerTest** y, a continuación, en el menú **Ver**, elija **Código**.

En el Editor de código, modifique el código de evento `Button1_Click` de la siguiente manera.

Visual Basic Express — Copiar código

```
MsgBox(player1.Position & " " & player1.FullName & _
    " is now at bat.")
```

Modifique el código de evento `Button2_Click` de la siguiente manera.

Visual Basic Express — Copiar código

```
MsgBox(player2.Position & " " & player2.FullName & _
    " is on deck.")
```

Presione F5 para ejecutar el programa y haga clic en cada botón para mostrar los resultados.

Observe que los resultados son los mismos que antes, aun cuando ahora se está utilizando el método `FullName` reemplazado.

En el menú **Archivo**, elija **Guardar todo**.

Realizar seguimientos: utilizar colecciones para administrar varios objetos

En esta lección, aprenderá a utilizar una colección para administrar grupos de objetos.

En una lección anterior, aprendió a utilizar matrices para administrar grupos de variables. Aunque las matrices también se utilizan para administrar grupos de objetos, Visual Basic Expresstiene un tipo de objeto especial denominado *colección*, el cual se puede utilizar para almacenar y recuperar grupos de objetos similares.

Al igual que una matriz, cada elemento de un objeto Collection tiene un índice que se puede utilizar para identificar dicho elemento. Además, cada elemento de un objeto **Collection** tiene una *clave*, un valor **String** que se puede utilizar para identificar el elemento. La ventaja de utilizar una clave es que no necesita recordar el índice de un elemento; en su lugar, puede referirse a él a través de un nombre significativo.

Crear una colección

Las colecciones son útiles cuando el programa utiliza varias instancias de la misma clase. Por ejemplo, examine la clase `Players` que creó en una lección anterior. Es probable que necesite varios objetos `Players` para representar un equipo de béisbol.

El primer paso para crear una colección es crear una instancia de un objeto **Collection**, tal como se muestra en la siguiente declaración.

Visual Basic Express — Copiar código

```
Dim baseballTeam As New Collection
```

Cuando se crea el objeto **Collection**, se puede utilizar el método **Add** para agregarle elementos y el método **Remove** para eliminarlos. Cuando agregue elementos, primero especifique el elemento que se va a agregar y luego el valor **String** que se va a utilizar como clave.

Visual Basic Express — Copiar código

```
baseballTeam.Add(playerObject, "Player's Name")
```

Al quitar un elemento, utilice la clave para especificar el elemento que se va a quitar.

Visual Basic Express — Copiar código

```
baseballTeam.Remove("Player's Name")
```

En el siguiente procedimiento, se agregarán dos nuevos objetos `Players` y, a continuación, se creará una colección `team` y se le agregarán los objetos `Players`, mediante la propiedad `Position` como una clave.

Inténtelo

Para crear una colección de objetos

Abra el proyecto `Persons` que creó en la lección anterior. Si no lo guardó, regrese a la lección anterior, Generar una clase en una clase existente: utilización de la herencia y finalice los procedimientos.

En el **Explorador de soluciones**, en el proyecto **PlayerTest**, seleccione el nodo **Form1.vb** y en el menú **Ver**, seleccione **Código**.

En el Editor de código, agregue lo siguiente a la sección de declaraciones (debajo de la declaración para `player2`).

Visual Basic Express — Copiar código

```
Dim player3 As New Persons.Players
Dim player4 As New Persons.Players
Dim team As New Collection
```

Agregue el siguiente código al procedimiento de evento `Form1_Load`.

Visual Basic Express — Copiar código

```
With player3
    .FirstName = "Eduardo"
    .LastName = "Saavedra"
    .Number = 52
    .Position = "First Base"
End With

With player4
    .FirstName = "Karl"
    .LastName = "Jablonski"
    .Number = 22
    .Position = "Pitcher"
End With

team.Add(player1, player1.Position)
team.Add(player2, player2.Position)
team.Add(player3, player3.Position)
team.Add(player4, player4.Position)
```

En el **Explorador de soluciones**, en el proyecto **PlayerTest**, seleccione el nodo **Form1.vb**. A continuación, en el menú **Ver**, elija **Diseñador**.

En el **Cuadro de herramientas**, arrastre un control ComboBox hasta el formulario.

En la ventana **Propiedades**, seleccione la propiedad **Items** y haga clic en el botón **...**.

En el **Editor de la colección de cadenas**, escriba lo siguiente y haga clic en **Aceptar**.

```
Catcher

First Base

Pitcher

Shortstop
```

Haga doble clic en el control **ComboBox** para abrir el Editor de código y escriba el siguiente código en el controlador de eventos `ComboBox1_SelectedIndexChanged`.

Copiar código

```
Dim SelectedPlayer As Persons.Players
SelectedPlayer = team(ComboBox1.SelectedItem)
MsgBox("Playing " & ComboBox1.SelectedItem & " is " & _
SelectedPlayer.FullName & "!")
```

Presione F5 para ejecutar el programa. Seleccione una posición de la lista desplegable, aparecerá en un cuadro de mensaje el jugador para esa posición.

Información detallada: utilizar un bucle For Each...Next en una colección

En esta lección, obtendrá información sobre cómo utilizar un bucle **For Each...Next** para recorrer una colección.

En una lección anterior, obtuvo información sobre cómo utilizar un bucle **For...Next** para ejecutar un bloque de código un determinado número de veces. Los objetos de la colección de Visual Basic Expressadmiten un tipo especial de bucle, el bucle **For Each...Next**, que se utiliza para ejecutar un bloque de código para cada elemento de la colección, en lugar de ejecutar el bloque un número fijo de veces.

Agregar un bucle For Each... Next

En la lección anterior, agregó manualmente al control ComboBox los valores de la propiedad `Position` de los objetos `Players` en la colección de equipos.

Aunque este sistema funciona para el ejemplo, no es el procedimiento recomendado, pues cada vez que agregue un nuevo jugador, también tendrá que actualizar la colección **Items** del control **ComboBox**.

Una manera mucho más adecuada consiste en agregar los valores `Position` a la colección **Items** recorriendo la colección `team` con un bucle **For Each...Next**.

En un bucle **For...Next**, primero debe declarar una variable de contador; con un bucle **For Each...Next** primero debe declarar una variable de objeto. El código siguiente muestra un bucle **For Each...Next**.

Visual Basic Express — Copiar código

```
Dim player As Persons.Players
For Each player In team
    ComboBox1.Items.Add(player.Position)
Next
```

En este caso, no importa cuántos `Players` tenga, el método **ComboBox1.Items.Add** se ejecutará una vez para cada objeto `Players` de la colección equipos y el valor `Position` se agregará a la lista.

Inténtelo

Para recorrer una colección

Abra el proyecto `Persons` de la lección anterior. Si no lo ha acabado, regrese a la lección anterior, Realizar seguimientos: utilizar colecciones para administrar varios objetos, y complete los procedimientos.

En el **Explorador de soluciones**, seleccione el nodo **Form1.vb** en el proyecto **PlayerTest** y, a continuación, en el menú **Ver**, elija **Diseñador**.

Seleccione el control **ComboBox**. A continuación, en la ventana **Propiedades**, seleccione la propiedad **Items** y haga clic en el botón **...**.

En el **Editor de la colección de cadenas**, elimine las cuatro entradas existentes y, a continuación, haga clic en **Aceptar**.

Haga doble clic en el formulario para abrir el Editor de código.

En el Editor de código, agregue lo siguiente a la sección de declaraciones (debajo de la declaración para `team`).

Visual Basic Express — Copiar código

```
Dim player As Persons.Players
For Each player In team
    ComboBox1.Items.Add(player.Position)
Next
```

Presione F5 para ejecutar el programa. Seleccione una posición de la lista desplegable. El jugador de esa posición se mostrará en un cuadro de mensaje.

Objetos visibles: crear el primer control de usuario

En el conjunto de lecciones anteriores ha aprendido a trabajar con clases. Las clases que ha creado se pueden reutilizar en otros programas, de modo que no tiene que escribir una y otra vez el mismo código.

Los controles también son clases que se pueden reutilizar en varios proyectos. Probablemente se encontrará diseñando la misma interfaz una y otra vez, por ejemplo, agregando controles **TextBox** para especificar el nombre y los apellidos y, a continuación, agregando código para combinarlos en un nombre completo. ¿No sería más práctico evitar todo ese trabajo extra?

De ahí proceden los *controles de usuario*. Imagine que un control de usuario es como una clase para crear objetos visibles (controles personalizados que podrá reutilizar igual que los controles que se incluyen en Visual Basic Express). La mayoría de los controles de usuario son controles *compuestos*, es decir, controles que se componen de uno o varios controles estándar de Visual Basic Express.

En las lecciones siguientes, aprenderá a crear un control de usuario compuesto que puede reutilizar en otros programas.

Comprender el Diseñador de controles de usuario

En esta lección, aprenderá a crear un control con el **Diseñador de controles de usuario**.

En el conjunto anterior de lecciones, aprendió a utilizar un proyecto de **bibliotecas de clase** para crear clases. Un control de usuario simplemente es una clase que se puede ver. Exactamente igual que los controles estándar que vienen con Visual Basic Express , los controles de usuario se puede colocar en formularios durante el diseño y aparecen al ejecutarse el programa.

Cuando diseña programas, organiza los controles y decide su apariencia en el diseñador de formularios. Hay también un diseñador para los controles de usuario, el **diseñador de controles de usuario**, que le permite a usted, al desarrollador, decidir la apariencia del control.

Crear controles de usuario

Un **control de usuario** es similar a cualquier otra clase, pero con la posibilidad agregada de poder colocarlo en el **Cuadro de herramientas** y mostrarlo en un formulario. Donde un módulo de **clase** tiene sólo código, un módulo de control de usuario tiene código y un diseñador. El **Diseñador de controles de usuario** es similar a un diseñador de formularios: tiene las propiedades para controlar el aspecto y comportamiento del control de usuario.

Las maneras de crear controles de usuario son ligeramente distintas, en función de la versión de Visual Basic Expressque esté utilizando: Visual Basic Express tiene un tipo de proyecto de **biblioteca de controles de Windows**; en Visual Basic Express, debe crear primero un proyecto de **bibliotecas de clase** y luego agregarlo a una plantilla de **controles de usuario**.

¡Inténtelo!

Para crear un control de usuario mediante Visual Basic Express

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el panel **Plantillas**, del cuadro de diálogo **Nuevo proyecto**, haga clic en **Biblioteca de clases** y luego en **Aceptar**.

En el menú Proyecto, haga clic en Agregar control de usuario.

En el cuadro de diálogo Agregar nuevo elemento, seleccione Control de usuario.

En el cuadro **Nombre**, escriba `NamesControl` y haga clic en **Agregar**.

Se agregará una nueva plantilla **Control de usuario** al proyecto y se abrirá el **Diseñador de controles de usuario**.

En el **Explorador de soluciones**, haga clic con el botón secundario del mouse en `Class1.vb` y elija **Suprimir**, a continuación, haga clic en **Aceptar**.

En el menú **Archivo**, haga clic en **Guardar todo**.

En el cuadro de diálogo **Guardar proyecto**, especifique `NamesUserControl` y haga clic en **Guardar**.

Para crear un control de usuario en Visual Studio 2008

En el menú **Archivo**, haga clic en **Nuevo proyecto**.

En el panel Plantillas, del cuadro de diálogo Nuevo proyecto, haga clic en Biblioteca de controles de Windows.

En el cuadro **Nombre**, escriba `NamesControl` y, a continuación, haga clic en **Aceptar**.

Se agregará una nueva plantilla **Control de usuario** al proyecto y se abrirá el **Diseñador de controles de usuario**.

En el menú **Archivo**, haga clic en **Guardar todo**.

En el cuadro de diálogo **Guardar proyecto**, especifique `NamesUserControl` y haga clic en **Guardar**.

Agregar controles al control de usuario

En esta lección aprenderá a agregar controles para crear un control de usuario compuesto.

Diseño de un control de usuario

Como se mencionó anteriormente, el tipo de control de usuario más común es un control compuesto, aquél que está formado por uno o más controles de formularios Windows Forms estándar. Se pueden agregar controles a plantillas **Control de usuario** arrastrándolos desde el **Cuadro de herramientas** hasta el **Diseñador de controles de usuario**, del mismo modo que lo haría cuando diseña formularios.

Una vez que ha agregado un control, puede cambiar su tamaño y moverlo en el diseñador, asimismo, puede establecer sus propiedades en la ventana **Propiedades**.

En este ejemplo, agregará un control **Label** para mostrar un nombre completo, y tres controles **TextBox** para especificar el primer nombre, el segundo nombre y el apellido.

Inténtelo

Para agregar controles a un Control de usuario

Abra el proyecto `NamesUserControl` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Comprender el Diseñador de controles de usuario, y realizar hasta el final los procedimientos de esa lección.

En el Explorador de soluciones, seleccione NamesControl.vb y en el menú Ver seleccione Diseñador.

En el **Cuadro de herramientas**, arrastre un control **Label** al diseñador.

> Sugerencia
>
> El Cuadro de herramientas es más fácil de utilizar si mantiene la ventana abierta. Puede hacer esto haciendo clic en el icono Ocultar automáticamente, que parece una chincheta.

En la ventana **Propiedades**, cambie la propiedad **Name** a `FullName`.

En el **Cuadro de herramientas**, arrastre tres controles **Textbox** al diseñador. Puede organizarlos como quiera.

En la ventana **Propiedades**, cambie las propiedades **Name** por `FirstName`, `MiddleName` y `LastName`.

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo

Agregar código al control de usuario

En esta lección, aprenderá a agregar código al control de usuario para mostrar el nombre completo y exponer nuevas propiedades.

Como los controles estándar, los controles de usuario tienen propiedades, métodos y eventos. Como desarrollador, escribirá código para controlar los eventos del control y decidirá qué propiedades se expondrán al usuario del control.

Controlar eventos en un control de usuario

Para que el control de usuario sea de utilidad, tendrá que escribir algún código que controle los eventos del control. Un procedimiento de control de eventos de un control de usuario no es distinto del que se escribe para un formulario o un control.

En este ejemplo, escribirá un procedimiento de evento que actualizará la etiqueta `FullName` con el contenido de los cuadros `FirstName`, `MiddleName` y `LastName` según escriba, con el controlador de eventos **TextChanged**.

Inténtelo

Para agregar código a un control de usuario

Abra el proyecto `NamesUserControl` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Comprender el Diseñador de controles de usuario, y realizar hasta el final los procedimientos de esa lección.

En el Explorador de soluciones, seleccione NamesControl.vb y en el menú Ver elija Código.

En el Editor de código, agregue el siguiente código para el controlador de eventos **FirstName_TextChanged**.

Copiar código

```
Private Sub FirstName_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles FirstName.TextChanged, MiddleName.TextChanged,
LastName.TextChanged

    ' Display the contents of the three text boxes in the label.

    FullName.Text = FirstName.Text & " " & MiddleName.Text & " " & LastName.Text

End Sub
```

Presione F5 para ejecutar el programa. Se abrirá el **UserControl TestContainer** y se mostrará el control de usuario.

Escriba el nombre, segundo nombre y apellido en los tres cuadros de texto; según escriba, se mostrará el nombre en la etiqueta `FullName`.

Si se fija en el código que ha especificado antes, observará que la cláusula **Handles** de la declaración controla el evento **TextChanged** de los tres controles **TextBox**. No importa qué cuadro de texto escriba primero, siempre se actualizará la etiqueta `FullName` según escriba.

Exponer las propiedades de un control de usuario

Las propiedades de los controles estándar permiten establecer y recuperar valores de un control en tiempo de diseño y en tiempo de ejecución. También deseará que determinadas propiedades del control de usuario estén disponibles para que pueda establecerlas en la ventana **Propiedades** durante el diseño y hacer referencia a ellas en el código.

Exponer propiedades en un control de usuario es muy similar a exponer propiedades en una clase, la diferencia principal es que puede exponer también las propiedades de los controles contenidos en el control de usuario. Como con las clases, puede declarar una propiedad y agregar código a los procedimientos **Get** y **Set**. Si expone una propiedad de un control contenido, no tendrá que declarar una variable privada para almacenar el valor, la propiedad del control lo almacena automáticamente.

Tal y como está ahora, no hay modo de recuperar el texto que se especifica en los controles `FirstName`, `MiddleName` y `LastName` de la etiqueta `FullName`. Necesita exponer los valores como propiedades para que el control resulte útil. Dado que no desea que el valor de la etiqueta `FullName` se

modifique fuera de su propio código, deseará exponerlo como una propiedad de sólo lectura.

Inténtelo

Para agregar propiedades

En el Editor de código, agregue el código siguiente para exponer los valores `FirstName`, `MiddleName` y `LastName` como propiedades.

Copiar código

```
Property FirstNameText() As String
    Get
        Return FirstName.Text
    End Get
    Set(ByVal value As String)
        FirstName.Text = value
    End Set
End Property
Property MiddleNameText() As String
    Get
        Return MiddleName.Text
    End Get
    Set(ByVal value As String)
        MiddleName.Text = value
    End Set
End Property
Property LastNameText() As String
    Get
        Return LastName.Text
    End Get
    Set(ByVal value As String)
        LastName.Text = value
    End Set
End Property
```

Agregue el código siguiente para exponer el valor de la etiqueta `FullName` como una propiedad de sólo lectura.

Copiar código

```
ReadOnly Property FullNameText() As String
    Get
        Return FullName.Text
    End Get
End Property
```

Presione F5 para ejecutar el programa.

En el **UserControl TestContainer**, desplácese a la parte inferior de la cuadrícula **Propiedades** y seleccione la propiedad **FirstNameText**. Escriba su nombre y, a continuación, seleccione la propiedad **FullNameText**; el cuadro de texto **FirstName** debería mostrar el nombre y la propiedad **FullNameText** debería coincidir.

Pruebe a cambiar algunas de las demás propiedades en la cuadrícula **Propiedades** y el propio control para ver cómo se relacionan. Esto es lo que un usuario del control experimentará en tiempo de diseño.

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Información detallada: agregar propiedades con valores con nombre

En esta lección, obtendrá información sobre cómo agregar una propiedad que contiene una lista de valores a su control de usuario.

Valores con nombre

Las propiedades que ha agregado hasta ahora a su control de usuario toman valores de cadena, pero las propiedades pueden ser de muchos tipos diferentes. En ocasiones, deseará proporcionar una lista de valores predefinidos entre los que elegir, como la propiedad **SizeMode** del control **PictureBox** que ha establecido en una lección anterior.

Supongamos, por ejemplo, que desea crear una propiedad para `NamesControl` que le permita elegir cómo la etiqueta `FullName` muestra el nombre. Necesitará una lista de valores entre los que elegir: nombre en primer lugar, apellido en primer lugar, sólo nombre y apellido, etc.

En Visual Basic, puede crear una *enumeración* que contiene los valores que desea. "Enumeración" en realidad significa "lista numerada"; Visual Basic

Expressalmacena los números, de forma que puede hacer referencia a los valores por el nombre. Una enumeración se declara utilizando la palabra clave **Enum**, como en el ejemplo siguiente.

Copiar código

```
Public Enum Display
    FirstMiddleLast
    FirstLast
    LastFirstMiddle
    LastFirst
End Enum
```

Cuando ha creado una enumeración, puede utilizarla como cualquier otro tipo. Para agregar una propiedad que muestre una lista de valores, en primer lugar declara una variable del mismo tipo que **Enum** y, a continuación, declara una propiedad del mismo tipo. En tiempo de diseño, una lista de valores incluida en la enumeración aparecerá en la ventana **Propiedades**.

¡Inténtelo!

Para agregar una propiedad que muestre una lista de valores

Abra el proyecto `NamesUserControl` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Agregar código al control de usuario, y realizar hasta el final los procedimientos de esa lección.

En el **Explorador de soluciones**, seleccione **NamesControl.vb** y en el menú **Ver** haga clic en **Código**.

En el Editor de código, agregue el siguiente código para crear una enumeración.

Copiar código

```
Public Enum Display
    FirstMiddleLast
    FirstLast
    LastFirstMiddle
    LastFirst
End Enum
```

Agregue el siguiente código para agregar una nueva propiedad.

Copiar código

```
Private DisplayStyleList As Display
Property DisplayStyle() As Display
    Get
        Return DisplayStyleList
    End Get
    Set(ByVal value As Display)
        DisplayStyleList = value
    End Set
End Property
```

Elimine el código existente en el controlador de eventos **FirstName_TextChanged** y reemplácelo con el código siguiente.

Copiar código

```
Select Case DisplayStyleList
    Case Display.FirstLast
        FullName.Text = FirstName.Text & " " & LastName.Text
    Case Display.FirstMiddleLast
        FullName.Text = FirstName.Text & " " & MiddleName.Text & " " & LastName.Text
    Case Display.LastFirst
        FullName.Text = LastName.Text & ", " & FirstName.Text
    Case Display.LastFirstMiddle
        FullName.Text = LastName.Text & ", " & FirstName.Text & " " & MiddleName.Text
End Select
```

Presione F5 para ejecutar el programa. Escriba el nombre, el apellido y la inicial en los tres cuadros de texto.

En **UserControl TestContainer**, desplácese a la parte inferior de la cuadrícula **Propiedades** y seleccione la propiedad **DisplayStyle**. Seleccione un valor diferente y, a continuación, cambie el texto de uno de los cuadros de texto para ver cómo afecta a la etiqueta.

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Probar un control de usuario

En esta lección, aprenderá a probar un control de usuario en otro proyecto y observar su comportamiento en tiempo de ejecución.

Comportamiento en tiempo de ejecución

Una vez que ha terminado el control de usuario y probado su comportamiento en tiempo de diseño en **TestContainer**, también deseará saber cómo se comporta cuando se utiliza en un programa. Visual Basic Express facilita la prueba del control de usuario agregando un proyecto de **aplicación para Windows**.

El control de usuario aparece automáticamente en el **Cuadro de herramientas** y puede agregarlo a un formulario y establecer sus propiedades igual que si se tratara de cualquier otro control.

Inténtelo

Para probar el control de usuario

Abra el proyecto `NamesUserControl` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Agregar código al control de usuario, y realizar hasta el final los procedimientos de esa lección.

En el menú **Archivo**, elija **Agregar** y haga clic en **Nuevo proyecto**.

En el cuadro de diálogo Agregar nuevo proyecto, seleccione Aplicación para Windows.

En el cuadro **Nombre**, escriba `UserControlTest` y haga clic en **Aceptar**.

Se agregará un nuevo proyecto en el **Explorador de soluciones** y se mostrará un nuevo formulario.

En el Explorador de soluciones, seleccione el proyecto UserControlTest y en el menú Proyecto, seleccione Establecer como proyecto de inicio.

En el **Cuadro de herramientas**, seleccione **NamesControl** y arrástrelo hasta el formulario.

En la ventana **Propiedades**, establezca las propiedades **FirstNameText**, **MiddleNameText** y **LastNameText** que correspondan a su nombre.

Presione F5 para ejecutar el programa. Cambie los nombres de los cuadros de texto para asegurarse de que la etiqueta se actualiza correctamente.

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Información detallada: personalizar el control de usuario

En esta lección, obtendrá información sobre cómo personalizar el control de usuario para hacerlo más útil.

Agregar etiquetas

En la última lección, probó que el control de usuario **NamesControl** funcionaba debidamente. Quizá también haya pensado posibles formas para mejorarlo. Por ejemplo, no resulta obvio qué nombre debe escribirse en qué cuadro de texto y no hay modo de saber con certeza si los usuarios han especificado tres nombres.

Para que el control de usuario resulte de mayor utilidad, puede agregar etiquetas que identifiquen todos los cuadros de texto. Podría establecer el texto para que las etiquetas indicaran "Nombre", "Segundo nombre" y "Apellido", pero, ¿qué sucede si después decide que prefiere una "Inicial de segundo nombre"? Es preferible crear propiedades para el texto de etiqueta de modo que pueda cambiar el texto en tiempo de diseño y dar un valor predeterminado a cada propiedad.

Inténtelo

Para personalizar el control de usuario

Abra el proyecto `NamesUserControl` que creó en la lección anterior. Si no lo guardó, primero deberá regresar a la lección anterior, Probar un control de usuario, y realizar hasta el final los procedimientos de esa lección.

En el Explorador de soluciones, seleccione NamesControl.vb y en el menú Ver seleccione Diseñador.

En el **Cuadro de herramientas**, arrastre tres controles **Label** al diseñador y coloque uno sobre cada **TextBox**.

En el Explorador de soluciones, seleccione NamesControl.vb y en el menú Ver elija Código.

En el Editor de código, agregue el código siguiente para crear las propiedades para el texto de la etiqueta.

Copiar código

```
Private text1 As String = "First Name"

Property Label1Text() As String
```

```
    Get
        Return text1
    End Get
    Set(ByVal value As String)
        text1 = value
        Label1.Text = text1
    End Set
End Property
Private text2 As String = "Middle Name"
Property Label2Text() As String
    Get
        Return text2
    End Get
    Set(ByVal value As String)
        text2 = value
        Label2.Text = text2
    End Set
End Property
Private text3 As String = "Last Name"
Property Label3Text() As String
    Get
        Return text3
    End Get
    Set(ByVal value As String)
        text3 = value
        Label3.Text = text3
    End Set
End Property
```

Observe que el código declara tres variables **Private** para el texto de etiqueta y que las declaraciones incluyen el valor predeterminado que se va a mostrar.

En el Editor de código, seleccione **(NamesControl Events)** del cuadro de lista desplegable izquierdo y, a continuación, seleccione el evento **Load** del cuadro de lista desplegable derecho.

Agregue el código siguiente al controlador de eventos **NamesControl_Load**.

Copiar código

```
' Initialize the three labels
Me.Label1.Text = Label1Text
Me.Label2.Text = Label2Text
Me.Label3.Text = Label3Text
```

En el menú Generar, elija Generar solución.

En el Explorador de soluciones, seleccione Form1.vb y en el menú Ver elija Diseñador.

Compruebe que las etiquetas tienen el texto predeterminado. Intente cambiar la propiedad **Label1Text** en la ventana **Propiedades** y compruebe que también cambia en el control.

En el menú **Archivo**, elija **Cerrar** para cerrar el Diseñador de Windows Forms.

Agregar validación

Otra personalización de utilidad sería agregar código para *validar* lo que se escribe con el fin de asegurarse de que es correcto. En lugar de validar cada uno de los controles **TextBox**, puede escribir código de validación para el control de usuario completo.

La mayoría de los controles tiene un evento **Validating** que se produce cuando se desplaza el foco fuera del control; aquí es donde especificará el código de validación. En este caso, deseará escribir el código para asegurarse de que cada cuadro de texto contiene un nombre.

Si están vacíos uno o varios cuadros de texto, deseará mostrar un cuadro de mensaje para recordar al usuario que especifique su nombre. Puede exponer una propiedad que contenga un mensaje predeterminado; de ese modo, el usuario del control puede cambiar el mensaje lo que dice el mensaje.

También es posible que el usuario del control no requiera un segundo nombre, de modo que también deseará agregar una propiedad **Boolean** para desactivar la validación para el cuadro de texto **MiddleName**.

Inténtelo

Para agregar la validación

En el Editor de código, agregue código para dos propiedades relacionadas con la validación, una para especificar si el segundo nombre es necesario y otra para especificar el mensaje que se mostrará si no se supera la validación.

Copiar código

```
Private required As Boolean = True
Property MiddleNameRequired() As Boolean
    Get
        Return required
    End Get
    Set(ByVal value As Boolean)
        required = value
    End Set
End Property
Private errormessage As String = "Please enter your name."
Property ValidationErrorMessage() As String
    Get
        Return errormessage
    End Get
    Set(ByVal value As String)
        errormessage = value
    End Set
End Property
```

En el Editor de código, seleccione **(NamesControl Events)** del cuadro de lista desplegable izquierdo y, a continuación, seleccione el evento **Validating** del cuadro de lista desplegable derecho.

Agregue el código siguiente al controlador de eventos **NamesControl_Validating**.

Copiar código

```
If MiddleNameRequired = True Then
    If FirstName.Text = "" Or MiddleName.Text = "" Or _
LastName.Text = "" Then
```

```
        MsgBox(ValidationErrorMessage)
    End If
Else
    ' Middle name isn't required.
    If FirstName.Text = "" Or LastName.Text = "" Then
        MsgBox(ValidationErrorMessage)
    End If
End If
```

En el menú Generar, elija Generar solución.

En el Explorador de soluciones, seleccione Form1.vb y en el menú Ver elija Diseñador.

Seleccione el control de usuario en el formulario y compruebe que aparecen las dos nuevas propiedades en la ventana **Propiedades**.

En el **Cuadro de herramientas**, arrastre un control **Button** al formulario.

Presione F5 para ejecutar el programa.

Escriba el nombre y apellido, pero no escriba un segundo nombre. Haga clic en el botón y se mostrará un cuadro de mensaje que contiene el**ValidationErrorMessage**.

En el menú **Archivo**, elija **Guardar todo** para guardar el trabajo.

Dibujar imágenes: utilizar gráficos

En una lección anterior, aprendió a crear la interfaz de usuario utilizando formularios y controles. En ocasiones, es posible que desee personalizar el aspecto del programa con imágenes o efectos que no se pueden crear con controles.

En Visual Basic, puede utilizar *métodos gráficos* para dibujar prácticamente cualquier cosa en un formulario o en un control. En este conjunto de lecciones, se darán a conocer las funciones gráficas en Visual Basic.

Ver gráficos

En esta lección, aprenderá a utilizar los métodos de gráficos en Visual Basic Expresspara dibujar en un formulario.

En una lección anterior, aprendió a mostrar imágenes en un formulario utilizando un control **PictureBox**. Eso funciona bien si ya tiene una imagen, pero a veces deseará dibujar algo directamente en el formulario. Por ejemplo, puede desear dibujar una línea para separar dos campos o un círculo para resaltar una etiqueta importante.

En Visual Basic, puede utilizar *métodos de gráficos* para dibujar prácticamente cualquier cosa en un formulario o en un control.

Fundamentos de gráficos

Antes de comenzar a dibujar, hay algunas cosas que debe saber. La pantalla de un equipo se compone de miles de puntos diminutos llamados *píxeles*; al definir el color de cada píxel, el programa controla lo que se muestra en la pantalla. Por supuesto, la mayoría de este trabajo ya se realiza automáticamente en el código que define formularios y controles.

Piense en un formulario como en un lienzo en el que puede dibujar o pintar: al igual que un lienzo real, un formulario tiene dimensiones. Mientras un lienzo real se mide en pulgadas o centímetros, un formulario se mide en píxeles. Un sistema de *coordenadas* determina donde se ubica cada píxel, con la *coordenada X* que mide de izquierda a derecha y la *coordenada Y* que mide de arriba a abajo.

Las coordenadas se inician en la esquina superior izquierda del formulario, de manera que si desea dibujar un punto único 10 píxeles desde la izquierda y 10 píxeles hacia abajo, las coordenadas X e Y se expresarán como `10, 10`.

Los píxeles también se utilizan para expresar el ancho y alto de los gráficos. Para definir un cuadrado que tiene 100 píxeles de ancho y 100 píxeles de alto, cuya esquina superior izquierda se encuentra 10 píxeles a la izquierda y 10 píxeles hacia abajo, se expresará en coordenadas como `10, 10, 100, 100`.

El acto de dibujar en la pantalla se conoce como *pintar*. Los formularios y controles tienen un evento **Paint** que aparece siempre que es necesario volver a dibujarlo, por ejemplo cuando se muestra un formulario por primera vez o cuando otra ventana lo ha cubierto. Generalmente, cualquier código que escribe para mostrar los gráficos lo tiene el controlador de eventos **Paint**.

Dibujar una línea

Para dibujar una línea en un formulario, hay dos cosas que debe definir: las coordenadas y el color. Como se observó anteriormente, las coordenadas X e Y se expresan en píxeles. Para una línea, hay dos conjuntos de coordenadas: la ubicación inicial seguida por la ubicación final.

Al igual como utilizaría un lápiz para dibujar una línea en una página de papel, Visual Basic Expressutiliza un objeto **Pen** para dibujar en el formulario. **Pen**

define el aspecto de la línea; en este caso, el color. En el siguiente procedimiento, dibujará líneas horizontales, verticales y diagonales en un formulario.

Inténtelo

Para dibujar líneas

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantilla**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Lines` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código y seleccione **Pintar** de la lista desplegable **Eventos**.

En el controlador de eventos **Form1_Paint**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
' Draw a 400 pixel black line 25 pixels from the top of the form.
e.Graphics.DrawLine(Pens.Black, 0, 25, 400, 25)
' Draw a 500 pixel red line 100 pixels from the left of the form.
e.Graphics.DrawLine(Pens.Red, 100, 0, 100, 500)
' Draw a diagonal blue line from the upper left to the lower right.
e.Graphics.DrawLine(Pens.Blue, 0, 0, Me.Width, Me.Height)
```

Presione F5 para ejecutar el programa. Debe ver tres líneas en el formulario.

Dibujar formas en un formulario

En esta lección, aprenderá a dibujar formas como rectángulos o círculos en un formulario.

En la lección anterior, aprendió a dibujar líneas en un formulario utilizando el método gráfico **DrawLine** y un objeto **Pen**. Además del método **DrawLine**, Visual Basic Expresstambién tiene métodos gráficos para dibujar formas y objetos gráficos conocidos como *pinceles* para rellenar formas.

Dibujar formas simples

Dibujar una forma es similar a dibujar una línea: se deben definir las coordenadas y el color con los que se va a dibujar. Mientras que una línea adopta las coordenadas que definen un punto inicial y final, una forma como un cuadrado o un rectángulo adopta coordenadas que describen su esquina superior izquierda, su ancho y su alto.

Los círculos y óvalos (también conocidos como *elipses*) no tienen esquina superior izquierda, por lo que en su lugar las coordenadas describen la esquina superior izquierda de su rectángulo delimitador: un rectángulo imaginario del mismo ancho y alto del círculo u óvalo.

Inténtelo

Para dibujar formas

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, seleccione **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Shapes` y, a continuación, haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código y seleccione **Pintar** en la lista desplegable **Eventos**.

En el controlador del evento **Form1_Paint**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
' Draw a 200 by 150 pixel green rectangle.
e.Graphics.DrawRectangle(Pens.Green, 10, 10, 200, 150)
' Draw a blue square
e.Graphics.DrawRectangle(Pens.Blue, 30, 30, 150, 150)
' Draw a 150 pixel diameter red circle.
e.Graphics.DrawEllipse(Pens.Red, 0, 0, 150, 150)
' Draw a 250 by 125 pixel yellow oval.
e.Graphics.DrawEllipse(Pens.Yellow, 20, 20, 250, 125)
```

Presione F5 para ejecutar el programa. Debería ver cuatro formas en el formulario.

Mantenga el proyecto abierto: le agregará elementos en el procedimiento siguiente.

Dibujar formas con relleno

Hasta ahora, las formas que ha dibujado son sólo contornos. Para dibujar formas con colores sólidos, debe utilizar uno de los métodos de *relleno*, como **FillRectangle** o **FillEllipse**. Los métodos de relleno utilizan un objeto **Brush**, otro tipo de objeto gráfico que puede pintar.

Al rellenar una forma con un color diferente, deberá definir coordenadas que sean más pequeñas que la forma; de lo contrario se cubrirá el borde. Por ejemplo, para rellenar un cuadrado con las coordenadas `0, 0, 150, 150`, se especificará un relleno con las coordenadas `1, 1, 148, 148`, que tiene en cuenta el grosor de un píxel de la línea.

Para dibujar formas con relleno

En el controlador del evento **Form1_Paint**, agregue el siguiente código debajo del código que escribió antes.

Visual Basic Express — Copiar código

```
' Fill the circle with the same color as its border.
e.Graphics.FillEllipse(Brushes.Red, 0, 0, 150, 150)
' Fill the square with a different color.
e.Graphics.FillRectangle(Brushes.Aquamarine, 31, 31, 148, 148)
```

Presione F5 para ejecutar el programa.

Observe que el cuadrado con relleno aparece por encima del círculo con relleno, pero esa parte de su borde ha desaparecido. El orden en el que se llama a los métodos gráficos determina el orden en el que se dibujan; en este caso, el círculo con relleno se dibujó después del rectángulo con borde azul.

Trate de cambiar el orden de los métodos y vea lo que pasa.

Dibujar texto en un formulario

En esta lección, aprenderá a dibujar un texto en un formulario mediante la utilización de métodos gráficos.

En una lección anterior, aprendió a mostrar texto mediante un control **Label**. Sin embargo, hay casos en los que podrá o deberá dibujar el texto personalmente utilizando métodos gráficos. Por ejemplo, si desea que el texto

esté inclinado, no puede utilizar un control **Label**, pero sí puede utilizar métodos gráficos para dibujar texto en cualquier ángulo.

Dibujar texto

Para dibujar texto en un formulario o control, se utiliza el método gráfico **DrawString**. Al igual que los otros métodos de dibujo, **DrawString** toma un objeto **Brush** que determina el color y las coordenadas que especifican dónde dibujar el texto, en este caso, las coordenadas X e Y de la esquina superior izquierda del rectángulo delimitador para el texto.

El método **DrawString** también tiene dos argumentos adicionales: la cadena que desea dibujar y la fuente que determina el aspecto del texto. Para especificar la fuente, primero debe crear un objeto **Font** y utilizar dicho objeto como un argumento al método **DrawString**.

Inténtelo

Para dibujar texto

En el menú **Archivo**, elija **Nuevo proyecto**.

En el cuadro de diálogo **Nuevo proyecto**, en el panel **Plantillas**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `DrawText` y haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código y seleccione **Pintar** de la lista desplegable **Eventos**.

En el controlador de eventos **Form1_Paint**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
' Create a font object.
Dim aFont As New System.Drawing.Font("Arial", 22, FontStyle.Bold)
' Display the text with the DrawString method.
e.Graphics.DrawString("Graphics are fun!", aFont, Brushes.Black, _
    20, 10)
```

Presione F5 para ejecutar el programa. Debe poder ver el texto que se muestra en el formulario.

Mantenga abierto el proyecto: continuará utilizándolo en el siguiente procedimiento.

Dibujar texto girado

Para dibujar texto en un ángulo, debe utilizar otro tipo de método gráfico que se llama *transformación*. Hay varios tipos de transformaciones disponibles para diversos efectos gráficos; en este caso, se utilizará el método **RotateTransform**.

El método **RotateTransform** toma un argumento único, el ángulo en que se girará el texto. La transformación se realiza en la línea del código a continuación del método **RotateTransform**; también se puede utilizar para girar formas o líneas dibujadas con otros métodos de dibujo.

Inténtelo

Para dibujar texto girado

En el controlador de eventos **Form1_Paint**, agregue el siguiente código debajo del código que escribió antes.

Visual Basic Express | Copiar código

```
' Rotate the text 45 degrees.
e.Graphics.RotateTransform(45)
e.Graphics.DrawString("And exciting too!", aFont, Brushes.Red, _
    100, 0)
```

Presione F5 para ejecutar el programa. Debe poder ver el texto girado que se muestra en el formulario.

Dibujar una imagen en un formulario

En esta lección, aprenderá a mostrar una imagen utilizando llamadas a gráficos.

En una lección anterior, aprendió a mostrar una imagen mediante un control **PictureBox**. También es posible mostrar una imagen de un archivo utilizando métodos gráficos de Visual Basic. Al igual que en la lección anterior, es necesario utilizar los métodos gráficos en lugar de un control **PictureBox** si desea hacer algo especial como girar la imagen.

Mostrar una imagen

Para mostrar una imagen en un formulario o en un control, utilice el método gráficos **DrawImage**. El método **DrawImage** toma una imagen de mapa de bits

como argumento, junto con las coordenadas X e Y que definen la esquina superior izquierda de la imagen.

Inténtelo

Para mostrar una imagen girada

En el menú **Archivo**, elija **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `DrawImage` y haga clic en **Aceptar**.

Se abrirá un nuevo proyecto de formularios Windows Forms.

En el Explorador de soluciones, haga doble clic en el nodo Mi proyecto para abrir el Diseñador de proyectos.

En el Diseñador de proyectos, haga clic en la ficha Recursos, seleccione Agregar recursos y, a continuación, Agregar archivo existente.

En el cuadro de diálogo **Agregar archivo existente a los recursos**, vaya a cualquier archivo de imagen, selecciónelo y, a continuación, haga clic en **Abrir**.

En el **Explorador de soluciones**, seleccione el nodo **Form1** y, en el menú **Ver**, seleccione **Código** para abrir el Editor de código.

En el Editor de código, seleccione la opción **Pintar** de la lista desplegable **Eventos**.

En el controlador de eventos **Form1_Paint**, agregue el siguiente código.

Visual Basic Express — Copiar código

```
e.Graphics.RotateTransform(45)
e.Graphics.DrawImage(My.Resources.picture, 50, 0)
```

Nota

Reemplace picture con el nombre del recurso que agregó en un paso anterior.

Presione F5 para ejecutar el programa. Debería ver la imagen girada en el formulario.

Distribuir un programa

Hay varias maneras de compartir el programa con otros. Mediante el uso de la publicación de ClickOnce, se puede poner el programa a disposición en un CD-ROM o DVD-ROM, o bien puede publicarlo en un sitio Web donde otros usuarios puedan descargarlo y ejecutarlo. También puede enviarlo por correo electrónico o simplemente copiarlo en un disco.

Compartir un programa: introducción a la implementación

En esta lección, aprenderá diferentes maneras de compartir programas con otras personas.

Una vez que termina de escribir, probar y depurar, es probable que desee compartir su obra maestra con los demás. El proceso de realizar copias del programa y distribuirlas se conoce como *implementación*.

Tal vez piense que puede copiar los archivos de programa en otro equipo y ejecutar el programa. Sin embargo, en muchos casos se encontrará con que el programa no se ejecuta. Esto es porque la mayoría de los programas depende de otro software conocidos como *componentes* que se deben instalar en el otro equipo. Si intenta ejecutar el programa y falta un componente, el programa no se ejecutará.

Publicación de ClickOnce

Visual Basic Expressincluye herramientas para implementar el programa con un método conocido como publicación de ClickOnce, que facilita la implementación incluyendo e instalando automáticamente todos los componentes necesarios junto con el programa. ClickOnce permite publicar el programa en un CD-ROM o DVD que puede compartir con otros usuarios.

Si tiene acceso a un servidor Web, también puede utilizar ClickOnce para publicar el programa en un sitio Web; el programa se puede descargar a través de Internet. Si cambia el programa más adelante, puede publicar la nueva versión y cualquier persona que lo haya descargado podrá descargar automáticamente la nueva versión la siguiente vez que lo ejecute.

Para publicar un programa utilizando ClickOnce, debe tener acceso a un grabador de CD o DVD o a un servidor Web. Si no tiene acceso a ninguno de estos dispositivos, puede compartir el programa comprimiéndolo y copiándolo en un disquete o enviándolo a través del correo electrónico.

Distribuir un programa en CD: publicar con ClickOnce

En esta lección, aprenderá a publicar el programa en un CD-ROM o DVD para compartirlo con otros usuarios.

Para compartir el programa con otros usuarios, primero deberá crear un programa de instalación y copiarlo en un CD-ROM u otro soporte. Además del programa en sí, el programa de instalación deberá incluir otros componentes o archivos necesarios para que el programa se ejecute; éstos se conocen como *requisitos previos*.

Como imaginará, determinar exactamente qué requisitos necesita puede ser una tarea difícil; afortunadamente las herramientas de publicación de ClickOnce en Visual Basic Expressrealizan estas y otras tareas. También puede utilizar la publicación de ClickOnce para publicar el programa en Internet o en una red local. Sin embargo, en esta lección, publicará el programa en un CD.

Nota

Para publicar un programa en un CD-ROM o DVD, debe tener el hardware y software adecuados en el equipo que está utilizando para escribir el programa. Si puede grabar CDs de música, también debería poder publicar en un CD. Si no dispone de esta posibilidad, quizá aún pueda compartir el programa utilizando otro método (para obtener más información, vea Enviar un programa por correo electrónico: crear un archivo comprimido).

Publicar con ClickOnce

Publicar el programa mediante ClickOnce es un proceso bastante sencillo, en el que se realizan algunas elecciones en el **Asistente para publicación** y se graban los archivos resultantes en un CD.

Antes de publicar el programa, pruébelo y asegúrese de que se ejecuta sin ningún error. Cuando está listo para publicar, puede iniciar el **Asistente para publicación** eligiendo el comando **Publicar** en el menú **Generar**.

El **Asistente para publicación** consta de tres pasos. El primer paso es seleccionar dónde desea ubicar el programa de instalación y cualquier archivo asociado. Si está publicando en un CD, seleccione una carpeta en el disco local; más adelante volverá a seleccionar esta ubicación para grabar el programa en un CD. El segundo paso es especificar cómo instalarán los usuarios el programa; en este caso, desde un CD-ROM.

El paso final implica especificar si el programa buscará automáticamente una versión más reciente del programa cada vez que se inicie. Si tiene acceso a un servidor Web, es posible publicar versiones actualizadas del programa en él, tal como se describe en Información detallada: distribuir un programa en Internet. Sin embargo, en este caso, como está publicando en un CD, el programa no tendrá la capacidad para buscar actualizaciones.

Cuando se ejecuta el **Asistente para publicación**, éste determina automáticamente cualquier requisito previo para el programa. De manera predeterminada, los requisitos previos como .NET Framework no se empaquetan realmente con el programa de instalación; durante la instalación, el programa de instalación busca los requisitos previos y, si es necesario, los descarga e instala desde Internet.

Para incluir los requisitos previos del programa de instalación en el CD, establezca una propiedad en el **Diseñador de proyectos**. Sin embargo, debe descargar primero los archivos redistribuibles para los requisitos previos en su equipo local. Para obtener más información, vea Información detallada: incluir requisitos previos en el programa.

Inténtelo

Para publicar en un CD

En el menú **Archivo**, seleccione **Abrir proyecto**.

En el cuadro de diálogo **Abrir proyecto**, vaya a cualquier proyecto **Aplicación para Windows** que haya creado en una lección anterior y haga clic en **Abrir**.

Presione F5 para ejecutar el proyecto. Si hay algún error, deberá corregirlo antes de continuar.

En el menú Depurar, elija Detener depuración.

En el menú **Generar**, seleccione **Publicar nombre de proyecto**, donde **Nombre de proyecto** es el nombre del proyecto.

Se iniciará el Asistente para publicación.

En la página **¿Dónde desea publicar la aplicación?** del **Asistente para publicación**, escriba la ruta de acceso donde desea publicar el programa, por ejemplo , `C:\My Programs`. Si la carpeta no existe, se le solicitará que la cree.

Haga clic en **Siguiente** para ir a la siguiente página del asistente.

En la página ¿Cómo instalarán los usuarios la aplicación?, seleccione Desde un CD-ROM o un DVD-ROM y haga clic en Siguiente.

En la página ¿Dónde buscará la aplicación las actualizaciones?, seleccione La aplicación no comprobará si hay actualizaciones.

Haga clic en **Finalizar**. El programa se publicará en la ubicación que especificó en la primera página del asistente.

Ahora puede utilizar la aplicación de grabación de CD o DVD para grabar un CD o DVD para el programa. Debe incluir todos los archivos en la carpeta donde publicó el programa.

Lleve el CD o DVD listos a otro equipo y ejecute el archivo Setup.exe. Si se debe instalar cualquier requisito previo como .NET Framework, se le solicitará descargarlo e instalarlo.

Una vez finalizada la instalación, puede ejecutar el programa desde el acceso directo que se encuentra en el menú **Inicio**.

Para aprender a empaquetar los requisitos previos con el programa, mantenga abierto el proyecto, lo utilizará en la lección Información detallada: incluir requisitos previos en el programa.

Información detallada: incluir requisitos previos en el programa

En esta lección, aprenderá a empaquetar los componentes necesarios con el programa mediante la publicación de ClickOnce.

De manera predeterminada, los programas publicados mediante la tecnología ClickOnce descargarán cualquier requisito previo necesario desde Internet durante la instalación. Si intenta instalar el programa en un equipo que no tiene acceso a Internet, la instalación puede fallar.

Para evitar esto, se pueden empaquetar los requisitos previos, como por ejemplo, los archivos redistribuibles de .NET Framework junto con el programa.

Nota

Si todas las personas que van a instalar el programa tienen acceso a Internet, se debe utilizar el método predeterminado; de esta forma, si una versión más reciente del requisito previo está disponible, se obtendrá la última versión.

Incluir requisitos previos

Para incluir los requisitos previos con el programa, cambie la propiedad **Publishing** en el **Diseñador de proyectos**. Observe que si incluye un requisito previo, debe incluir todos los requisitos previos; no puede incluir un requisito previo y haber descargado otro de Internet.

Inténtelo

Para incluir requisitos previos

Abra el proyecto de la lección anterior, Distribuir un programa en CD: publicar con ClickOnce.

En el **Explorador de soluciones**, seleccione el nodo del proyecto y en el menú **Proyecto**, elija **Propiedades**.

Se abrirá el Diseñador de proyectos.

En el **Diseñador de proyectos**, haga clic en la ficha **Publicar**.

Haga clic en el botón **Requisitos Previos** para abrir el cuadro de diálogo **Requisitos Previos**.

En el cuadro de diálogo Requisitos previos, active la casilla de verificación Descargar los requisitos previos desde la misma ubicación que mi aplicación y haga clic en Aceptar

En el menú **Generar**, seleccione **Publicar nombre de proyecto**, donde **Nombre de proyecto** es el nombre del proyecto.

Se inicia el Asistente para publicación.

Haga clic en **Finalizar** para publicar el programa.

> Nota
>
> Es posible que se le solicite descargar los archivos redistribuibles de .NET Framework si aún no lo ha hecho.

En el Explorador de Windows, vaya a la ubicación donde se publicó el programa y compruebe que los archivos Setup.exe y Dotnetfx.exe están allí.

Información detallada: distribuir un programa en Internet

En esta lección, aprenderá a utilizar la publicación de ClickOnce para implementar el programa en un servidor Web.

En la lección anterior, aprendió a publicar un programa en un CD-ROM o DVD-ROM. Si tiene acceso a un servidor Web, también puede utilizar ClickOnce con el fin de publicar el programa para que esté disponible en Internet.

Además de conseguir que el programa tenga una difusión más amplia, la publicación en un servidor Web permite aprovechar las funciones de actualización automática de ClickOnce. Si posteriormente publica una nueva versión del programa, cuando un usuario trate de ejecutarlo se le pedirá que descargue e instale la nueva versión.

> Nota
>
> Para publicar en un servidor Web, el servidor Web debe ejecutar IIS (Servicios de Internet

Information Server), las Extensiones de FrontPage deben estar instaladas y debe tener privilegios administrativos en IIS.

Publicar en un servidor Web

Publicar en un servidor Web es similar a publicar en un CD-ROM o DVD-ROM; sólo debe realizar algunas elecciones en el **Asistente para publicación**.

Una opción es si el programa estará disponible sin conexión, es decir, cuando el equipo no esté conectado a Internet. Si el programa está disponible con y sin conexión, se agregará una entrada en el menú **Inicio** de Windows para que el usuario pueda iniciar el programa. Si el programa está disponible sólo con conexión, se descargará cada vez que un usuario desee ejecutarlo y no se agregará al menú **Inicio**.

¡Inténtelo!

Para publicar en un servidor Web

En el menú **Archivo**, seleccione **Abrir proyecto**.

En el cuadro de diálogo **Abrir proyecto**, vaya a cualquier proyecto de **Aplicación para Windows** y haga clic en **Abrir**.

Presione F5 para ejecutar el proyecto. Si hay algún error, deberá corregirlo antes de continuar.

En el menú Depurar, elija Detener depuración.

En el menú **Generar**, seleccione **Publicar nombre de proyecto**, donde **Nombre de proyecto** es el nombre del proyecto.

Se iniciará el Asistente para publicación.

En la página **¿Dónde desea publicar la aplicación?** del **Asistente para publicación**, escriba la dirección URL del sitio Web donde desea publicar el programa, por ejemplo `http://www.microsoft.com/myprogram`.

> **Nota**
>
> Para publicar en un servidor Web, el servidor Web debe ejecutar IIS (Servicios de Internet Information Server), las Extensiones de FrontPage deben estar instaladas y debe tener privilegios administrativos en IIS.

Haga clic en **Siguiente** para ir a la siguiente página del asistente.

En la página ¿La aplicación estará disponible sin conexión?, seleccione el valor predeterminado Sí, esta aplicación está disponible con o sin conexión.

Haga clic en **Finalizar** para publicar el programa.

El programa se publicará en el sitio Web especificado y se creará una página HTML.

En otro equipo, abra Internet Explorer, vaya a la dirección URL que escribió en el paso 6 y haga clic en el vínculo **Instalar** para instalar el programa.

Avanzar: ¿A dónde puedo ir desde aquí?

Ha finalizado las lecciones de Paseo con guía por Visual Basic. Todavía no es un experto en Visual Basic, pero ya conoce lo suficiente como para comenzar a escribir sus programas.

Si ya tiene una idea para un programa, comience a ponerla en marcha. Si desea obtener más información sobre Visual Basic, consulte lo siguiente

Aumentar la productividad: desarrollo rápido de aplicaciones

En los primeros días de la programación, finalizar un programa sencillo llevaba días o incluso semanas. Cuando se presentó por primera vez Visual Basic Expressen 1991, revolucionó la programación, ya no era necesario escribir código para crear una interfaz de usuario ni había que preocuparse por la administración de memoria. Esta nueva manera de programar se denominó *desarrollo rápido de aplicaciones* o RAD (Rapid Application Development).

La ventaja principal de la programación RAD es el aumento de la productividad. Visual Basic Express presenta muchas características que ayudan a crear mejores aplicaciones en menos tiempo. A continuación se enumeran algunas de esas características.

Nota

Si utiliza Visual Basic Express, quizá algunos vínculos de Ayuda en esta página no estén disponibles, en función de las opciones que seleccione durante la instalación. Para obtener más información, vea Solución de problemas de Visual Basic Express.

Fragmentos de código

Un modo de aumentar la productividad es evitar escribir el mismo código una y otra vez. Visual Basic Express incluye una biblioteca de código con 500 miniprogramas aproximadamente, denominados *fragmentos de código de IntelliSense*, listos para ser insertados en una aplicación. Cada miniprograma realiza una tarea de programación completa, como crear un archivo, enviar un mensaje de correo electrónico o dibujar un círculo. Puede insertar un miniprograma en el código fuente con unos pocos clics del mouse (ratón).

Una vez que se inserta el miniprograma, se resaltan los fragmentos de código que es preciso sustituir; puede especificar sus valores si lo prefiere. Por ejemplo, un fragmento de código que dibuja una línea en un formulario tendrá valores para el color, la ubicación y la longitud. Puede cambiar estos valores según sus necesidades personales, o no hacer nada y dibujar una línea con los valores predeterminados.

También puede crear miniprogramas que satisfagan sus necesidades, agregarlos a la biblioteca y utilizarlos cuando necesite. Cuando cree miniprogramas, deberá decidir qué partes del código se resaltarán y cuáles serán los valores predeterminados. Para obtener más información, vea Crear y utilizar fragmentos de código de IntelliSense.

Una tarea común que se puede llevar a cabo con fragmentos de código es leer y escribir texto en un archivo. El procedimiento siguiente muestra cómo los fragmentos de código pueden hacerle más productivo.

Inténtelo

Para utilizar fragmentos de código

En el menú **Archivo**, seleccione **Nuevo proyecto**.

En el panel **Plantillas**, en el cuadro de diálogo **Nuevo proyecto**, haga clic en **Aplicación para Windows**.

En el cuadro **Nombre**, escriba `Snippets` y, a continuación, haga clic en **Aceptar**.

Se abre un nuevo proyecto de formularios Windows Forms.

Haga doble clic en el formulario para abrir el Editor de código.

En el Editor de código, haga clic con el botón secundario del mouse en el controlador de eventos `Form1_Load` y elija **Insertar fragmento de código** en el menú desplegable.

Se mostrará una lista de categorías de miniprograma.

Haga doble clic en Procesar unidades, carpetas y archivos

Se mostrará una lista de miniprogramas.

Haga doble clic en Escribir texto en un archivo.

Se insertará el código siguiente y se resaltarán `"C\Test.txt"` y `"Text"`.

Copiar código

```
My.Computer.FileSystem.WriteAllText("C:\Test.txt", "Text", True)
```

Nota

El método WriteAllText creará el archivo si no existe. Si ya existe, agregará el texto al final del archivo.

Reemplace `"C\Test.txt"` con `"C\MySnippetTest.txt"` y `"Text"` por `"This is really fast!"`.

Agregue un segundo miniprograma, haga clic con el botón secundario del mouse y seleccione **Insertar fragmento de código** en el menú.

Haga doble clic en Procesar unidades, carpetas y archivos

Haga doble clic en Leer texto desde un archivo.

Se insertará el código siguiente y se resaltará `"C\Test.txt"`.

Copiar código

```
Dim fileContents As String
fileContents = My.Computer.FileSystem.ReadAllText("C:\Test.txt")
```

Reemplace `"C\Test.txt"` por `"C\MySnippetTest.txt"`.

Agregue el código siguiente debajo del último miniprograma para mostrar el resultado.

Copiar código

```
MsgBox(fileContents)
```

Presione F5 para ejecutar el programa.

Se creará un archivo con el texto especificado y se mostrará un cuadro de mensaje con el contenido del archivo.

Dedique algún tiempo a familiarizarse con los miniprogramas de código incluidos en Visual Basic Express , le ahorrarán mucho tiempo y esfuerzo cuando escriba código. Para obtener más información, vea Cómo: Administrar fragmentos de código.

Desarrollo con la función My

Otra característica RAD incluida en Visual Basic Express se llama **My**. **My** es un conjunto de objetos que contienen las funciones más utilizadas relacionadas con el equipo, la aplicación, el usuario, etc. Puede considerar **My** una marcación rápida para llegar a funciones que, de otro modo, requerirían mucho código adicional.

Por ejemplo, suponga que desea determinar el número de versión de la aplicación. En la versión anterior de Visual Basic, el código se parecería al siguiente.

Copiar código

```
Dim VersionNumber As String

VersionNumber = System.Diagnostics.FileVersionInfo.GetVersionInfo _
(System.Reflection.Assembly.GetExecutingAssembly.Location).FileVersion
```

Con el nuevo objeto **My.Application**, se parece a éste.

Copiar código

```
Dim VersionNumber As String

VersionNumber = My.Application.Info.Version.ToString
```

Como puede ver, el procedimiento **My** es mucho más sencillo (y mucho más fácil de descubrir), lo que ahorra tiempo y esfuerzo. Aun así, podría utilizar la otra manera de determinar el número de versión, pero ¿por qué hacerlo?

Aunque quizá no se haya dado cuenta, ya ha utilizado **My** en varias lecciones anteriores. Cuando escriba código para una próxima aplicación, explore los objetos **My** escribiendo `My` y desplazándose por la lista de elementos que aparece. Para obtener más información, vea Desarrollo con la función My.

IntelliSense

A medida que avanzaba por las lecciones y escribía código, quizá haya observado que según escribía, aparecía una lista desplegable de opciones en el Editor de código. Éste es un ejemplo de la característica conocida como IntelliSense.

IntelliSense ofrece varias características que facilitan el acceso a referencias del lenguaje. Al escribir el código, no necesita abandonar el Editor de código para obtener información sobre los elementos del lenguaje. Puede quedarse donde está, buscar la información que necesite, insertar elementos del lenguaje directamente en el código e, incluso, dejar que IntelliSense termine de escribir el texto automáticamente.

IntelliSense también es de utilidad en la depuración. En el Editor de código, puede mover el cursor sobre una variable para mostrar información sobre herramientas con el valor actual de la variable. IntelliSense también está disponible al escribir el código en la ventana **Inmediato**. Para obtener más información, vea Utilizar IntelliSense.

Sugerencias y trucos: no sabía que pudiera hacerlo

En el paseo guiado de Visual Basic Expressha aprendido a hacer diversas tareas, pero sólo son una pequeña muestra de las posibilidades de Visual Basic Express . Incluso los expertos en Visual Basic Expressdescubren constantemente nuevas sugerencias y trucos. A continuación se presentan cosas menos conocidas que se pueden hacer con el producto.

Nota

Si utiliza Visual Basic Express, es posible que algunos vínculos de esta página de Ayuda no estén disponibles, en función de las opciones seleccionadas durante la instalación. Para obtener más información, vea Solución de problemas de Visual Basic Express.

Crear formularios Windows Forms con forma

¿Se ha cansado de los formularios rectangulares? ¿Desee crear una apariencia "decapada" para su aplicación, como el Reproductor de Windows Media? Es sencillo con Visual Basic Express : puede crear una imagen de mapa de bits con la forma que desee y utilizarla como un formulario, agregando código de modo que se pueda mover y cerrar. Para obtener más información, vea Cómo: Crear formularios Windows Forms no rectangulares.

Crear ventanas divisorias

¿Desea crear un formulario como el Document Explorer que está utilizando ahora, con dos o más áreas que el usuario pueda cambiar de tamaño? El control **SplitContainer** de formularios Windows Forms permite hacerlo sin ningún código. Sólo tiene que soltar un control **SplitContainer** en el formulario y luego agregar controles sobre éste; el comportamiento de cambio de tamaño está disponible automáticamente cuando se ejecuta la aplicación.

También puede agregar varios controles **SplitContainer** al formulario para tener regiones de tamaño variable dentro de regiones, lo que permite crear una aplicación parecida a Microsoft Outlook. Para obtener más información, vea SplitContainer (Control, formularios Windows Forms).

Reproducir sonidos

Si crea un juego, probablemente deseará que el programa reproduzca sonidos en respuesta a distintos eventos. El objeto **My.Computer.Audio** permite hacer exactamente eso, reproducir archivos de onda que puede incluir en la

aplicación o incluso reproducir archivos directamente desde Internet. Para obtener más información, vea My.Computer.Audio (Objeto).

Guardar las preferencias del usuario

Probablemente haya observado que muchas aplicaciones basadas en Windows "recuerdan" sus preferencias, como el modo en que se organizan las ventanas o las barras de herramientas que se mostraron la última vez que utilizó la aplicación. Puede hacer lo mismo en sus programas creando y utilizando la *configuración de la aplicación* para almacenar información y recuperarla la próxima vez que se ejecuta la aplicación. Para obtener más información, vea Información general sobre la configuración de la aplicación.

Agregue personalización

¿Se ha preguntado alguna vez cómo las páginas Web recuerdan el nombre de un usuario y muestran mensajes como "Bienvenido de nuevo (*insert your name here*)" ? Puede hacer algo similar en su aplicación utilizando la propiedad My.User.Name (Propiedad) para obtener el nombre del usuario que ha iniciado la sesión en curso en el equipo. Para obtener más información, vea My.User (Objeto).

Utilizar código de Visual Basic Express6.0

¿Tiene un ejemplo de código de Visual Basic Express6.0 que le gustaría utilizar en Visual Basic Express ? La herramienta **Actualizar código de Visual Basic Express6** convierte el código de Visual Basic Express6.0 y lo inserta en el código de Visual Basic Express . Si no se puede traducir completamente el código, se agregarán comentarios con vínculos a temas de Ayuda que describen lo que necesita hacer para que funcione el código. Para obtener más información, vea Cómo: Actualizar código de Visual Basic Express6.0 con el cuadro de diálogo Actualizar código de Visual Basic Express6.

Utilizar el subprocesamiento múltiple para mejorar el rendimiento

Las aplicaciones de Visual Basic Expresspueden realizar varias tareas a la vez utilizando una técnica llamada *subprocesamiento múltiple*. El subprocesamiento múltiple es un proceso en el que una tarea se ejecuta en un subproceso de ejecución independiente, lo que mejora el rendimiento y la capacidad de respuesta del programa.

Por ejemplo, supongamos que tiene un programa que descarga un archivo de Internet: la descarga podría llevar mucho tiempo e impediría que el usuario realizara nada más hasta que terminara. Si se realiza la descarga en un subproceso independiente, el usuario puede realizar otras operaciones mientras se descarga el archivo en segundo plano. Para obtener más información, vea Subprocesamiento múltiple en Visual Basic.

Visual Basic Express también tiene un componente **BackgroundWorker** que facilita realizar tareas en segundo plano. Para obtener más información, vea Tutorial: Implementar un formulario que utiliza una operación en segundo plano.

Crear documentación XML

La documentation XML permite agregar comentarios a una clase o control de usuario para que otro programador entienda cómo utilizarlos. Por ejemplo, suponga que el control de usuario tiene una propiedad denominada "Stretch", el nombre no le indica lo que la propiedad realmente hace. La documentación XML permite agregar una descripción como "Determina si se expandirá el texto para rellenar la pancarta"; se mostrará la descripción en la ventana **Propiedades** y en IntelliSense. Para obtener más información, vea Documentar el código con XML (Visual Basic).

Instalar .NET Framework junto con su programa

Todo programa creado con Visual Basic Express requiere que el motor en tiempo de ejecución de .NET Framework esté instalado en el equipo en el que se ejecutará el programa; algunos programas pueden requerir otros archivos o *requisitos previos*. Si comparte el programa usando la publicación ClickOnce, puede utilizar una característica llamada *requisito previo de inicio* para incluir estos archivos e instalarlos automáticamente. Para obtener más información, vea Cómo: Instalar requisitos previos mediante una aplicación ClickOnce.