

Introducción a Windows Forms

Contenido

Descripción	1
Lección: crear un formulario	2
Lección: añadir controles a un formulario	17
Lección: crear un formulario heredado	26
Lección: organizar controles en un formulario	35
Lección: Crear aplicaciones MDI	43

Descripción

- Crear un formulario
- Añadir controles a un formulario
- Crear un formulario heredado
- Organizar controles en un formulario
- Crear aplicaciones MDI

Introducción

Windows Forms forma parte del nuevo Microsoft® .NET Framework, y utiliza numerosas tecnologías nuevas, incluyendo un marco de trabajo común de aplicaciones, el entorno de ejecución gestionado, seguridad integrada y principios de diseño orientados a objetos. Además, Windows Forms ofrece soporte total para conectar rápida y fácilmente servicios Web XML (Extensible Markup Language) y crear aplicaciones ricas con acceso a datos basadas en el modelo de datos ADO.NET. Con el nuevo entorno de desarrollo compartido Visual Studio® .NET, los desarrolladores pueden crear aplicaciones Windows Forms utilizando cualquiera de los lenguajes soportados por la plataforma .NET, incluyendo Microsoft Visual Basic® .NET y Microsoft Visual C#™ .NET.

Objetivos

En este módulo, estudiaremos cómo:

- Crear un formulario y añadir controles.
- Crear un formulario heredado utilizando herencia visual.
- Organizar controles en un formulario.
- Crear aplicaciones MDI (Multiple Document Interface).

Lección: crear un formulario

- Windows Forms frente a Web Forms
- Cómo crear un formulario
- Cómo establecer las propiedades de un formulario
- Ciclo de vida de un formulario
- Cómo gestionar eventos del formulario
- Código generado por el Diseñador de Windows Forms

Introducción

Los formularios son el elemento básico del interfaz de usuario (IU) en aplicaciones creadas para Microsoft Windows®. Proporcionan un marco de trabajo que puede utilizarse por toda la aplicación para crear un aspecto coherente. Los formularios de aplicaciones basadas en Windows se utilizan para presentar información al usuario y aceptar la introducción de datos por parte del mismo.

Los formularios exponen propiedades que definen su apariencia, métodos que definen su comportamiento, y eventos que definen su interacción con el usuario. Estableciendo las propiedades y escribiendo código para responder a sus eventos, el formulario se personaliza para satisfacer los requerimientos de las aplicaciones. El formulario es un control derivado de la clase **Form**, que a su vez deriva de la clase **Control**. El marco de trabajo también permite heredar de formularios existentes para añadir más funcionalidades o modificar el comportamiento existente. Cuando se añade un formulario a un proyecto, se puede escoger si hereda de la clase **Form** proporcionada por el .NET Framework, o de un formulario creado con anterioridad.

Esta lección cubre los conceptos básicos de los formularios y cómo añadir controles a los formularios.

Objetivos de la lección

En esta lección, aprenderemos a:

- Describir un formulario.
- Determinar si utilizar Windows Forms o Web Forms en un determinado escenario.
- Crear un formulario.
- Establecer las propiedades de un formulario.
- Describir los eventos y métodos en el ciclo de vida de los formularios.

Windows Forms frente a Web Forms

Característica	Windows Forms	Web Forms
Implantación	Puede ejecutarse sin alterar el Registro	No se requiere descarga
Gráficos	Incluye GDI+	Los gráficos interactivos o dinámicos requieren ida y vuelta al servidor para su actualización
Respuesta	Velocidad de respuesta más rápida posible para aplicaciones interactivas	Pueden aprovechar el HTML Dinámico del navegador y crear ricos IU
Plataforma	Requiere el .NET Framework ejecutándose en la máquina cliente	Sólo requiere un navegador
Modelo de programación	Basado en un modo de intercambio de mensajes Win32 en el lado cliente	Los componentes de aplicaciones se invocan mediante HTTP
Seguridad	Seguridad basada en código y basada en roles	Seguridad basada en roles

Introducción

Cuando se diseñan aplicaciones que necesitan de un interfaz de usuario, existen dos opciones: Windows Forms y Web Forms. Ambos disponen de soporte completo en tiempo de diseño por parte del entorno de desarrollo y pueden generar un rico interfaz de usuario y funcionalidad avanzada de aplicaciones para solucionar problemas de negocio. Cuando se dispone de varias opciones, es importante saber qué opción utilizar en cada momento.

Windows Forms

Windows Forms se utiliza para desarrollar aplicaciones en las que se espera que el cliente tenga una parte importante de la carga de proceso de una aplicación. Entre éstas se incluyen las aplicaciones clásicas de escritorio para el interfaz de programación de aplicaciones Microsoft Win32®. Algunos de los ejemplos incluyen aplicaciones de dibujo o gráficas, sistemas de introducción de datos, sistemas punto de venta, y juegos. Todas estas aplicaciones dependen de la potencia del equipo de sobremesa para procesar y mostrar contenido con alto rendimiento.

Web Forms

Web Forms de ASP.NET se utilizan para crear aplicaciones en las que el interfaz de usuario principal es un navegador. Incluyen aplicaciones pensadas para estar disponibles públicamente en la World Wide Web, como las aplicaciones de correo electrónico.

Windows Forms vs. Web Forms

La siguiente tabla ofrece una comparativa de distintos criterios de comparación y cómo las tecnologías Windows Forms y Web Forms satisfacen estos criterios.

Característica / criterio	Windows Forms	Web Forms
Implantación	Las aplicaciones pueden descargarse, instalarse y ejecutarse directamente en el ordenador del usuario sin ninguna alteración del registro.	No tiene una implantación específica en el cliente; éste únicamente requiere un navegador. El servidor debe ejecutar Microsoft .NET Framework. Las actualizaciones de la aplicación se realizan actualizando el código en el servidor.
Gráficos	Windows Forms incluye GDI+ (Graphic Design Interface (GDI+), que permite utilizar gráficos sofisticados para juegos y otros entornos gráficos de gran riqueza.	Los gráficos interactivos o dinámicos requieren comunicación de ida y vuelta al servidor para actualizarse cuando se utilizan en Web Forms. Puede utilizarse GDI+ en el servidor para crear gráficos personalizados.
Respuesta	Windows Forms puede ejecutarse completamente en el equipo cliente; puede proporcionar mayor velocidad de respuesta para aplicaciones que requieren un alto grado de interactividad.	Si se sabe que los usuarios dispondrán de Microsoft Internet Explorer 5 o posterior, una aplicación Web Forms puede aprovechar las capacidades de HTML (DHTML) dinámico del navegador para crear una IU rica y con buena capacidad de respuesta. Si los usuarios tienen otros navegadores, la mayor parte del procesamiento (incluyendo tareas relacionadas con la IU, como la validación) requiere una comunicación de ida y vuelta al servidor Web, lo cual puede afectar a la capacidad de respuesta.
Plataforma	Windows Forms requiere que el .NET Framework se esté ejecutando en el equipo cliente.	Web Forms únicamente requiere un navegador. Los navegadores con capacidad DHTML pueden aprovechar las características adicionales, pero Web Forms puede diseñarse para funcionar con todos los navegadores. El servidor Web debe estar ejecutando el .NET Framework.

(continuación)

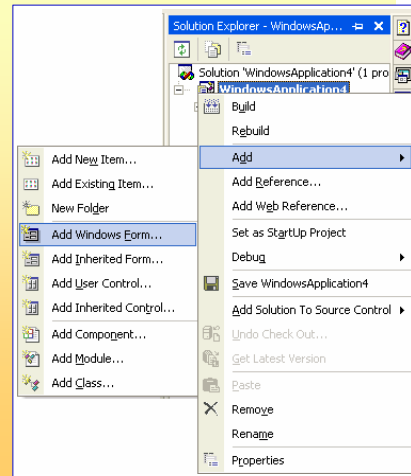
Característica/ criterio	Windows Forms	Web Forms
Modelo de programación	Windows Forms está basado en un modo de mensajería Win32 en el lado cliente, en el que el desarrollador crea, utiliza y desecha las instancias de componentes.	Web Forms depende de un modelo principalmente asíncrono y desconectado en el que los componentes están débilmente acoplados del front end de la aplicación. Normalmente, los componentes de la aplicación son invocados utilizando HTTP. Este modelo puede no ser apropiado para aplicaciones que requieran un rendimiento extremo del extremo de usuario o para aquellas con transacciones de gran volumen. De modo similar, las aplicaciones con Web Forms pueden no ser apropiadas para aplicaciones de bases de datos que requieren altos niveles de control de concurrencia (por ejemplo, bloqueo pesimista).
Seguridad	Windows Forms utiliza permisos granulares en su implementación de seguridad de acceso al código para proteger los recursos e información confidencial del equipo. Esto permite una prudente exposición de las funcionalidades, preservando la seguridad.	Web Forms permite controlar la identidad bajo la cual se ejecuta el código de una aplicación servidora. Las aplicaciones pueden ejecutar código utilizando la identidad de la entidad que realiza la petición, lo que se denomina imitación. Las aplicaciones también pueden adaptar dinámicamente su contenido basándose en la identidad o rol del solicitante. Por ejemplo, un director podría obtener acceso a un sitio o contenido que requiera un mayor nivel de seguridad que alguien cuyas credenciales sean inferiores.

Cómo crear un formulario

■ Cuando se crea un nuevo proyecto, se crea un formulario base

■ Para crear un nuevo formulario

1. Hacer clic con el botón derecho en el Explorador de soluciones
2. Hacer clic en **Agregar**
3. Hacer clic en **Windows Forms**



Introducción

En una aplicación basada en Windows, el formulario es el principal elemento para la interacción con el usuario. Mediante la combinación de controles y nuestras propias acciones, podemos solicitar información al usuario y responder a ella.

En Visual Studio .NET, el formulario es una ventana que se utiliza en la aplicación. Cuando creamos un nuevo proyecto de aplicación para Windows, Visual Studio .NET proporciona una vista de Diseño que contiene un formulario. El formulario predeterminado contiene los elementos mínimos utilizados por la mayoría de formularios: una barra de título, un cuadro de control y los botones **Minimizar**, **Maximizar** y **Cerrar**.

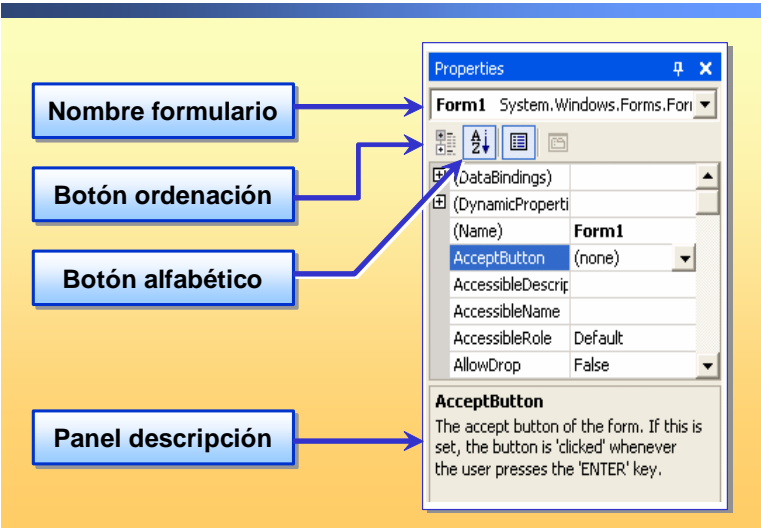
Procedimiento: crear formularios

La mayoría de aplicaciones requieren más de una ventana. Debe agregar un formulario a su proyecto por cada ventana que requiera su aplicación.

Para añadir formularios adicionales a nuestro proyecto:

1. Si el Explorador de soluciones no está abierto, en el menú **Ver**, hacer clic en **Explorador de soluciones**.
2. En el Explorador de soluciones, hacer clic con el botón derecho en el nombre del proyecto, seleccionar **Agregar**, y hacer clic en **Windows Form**.
3. En el cuadro de diálogo **Agregar nuevo elemento**, en el cuadro **Nombre**, escribir un nombre adecuado para el formulario y, a continuación, hacer clic en **Abrir**.

Cómo establecer las propiedades del formulario



Introducción Cuando se genera el interfaz de usuario de una aplicación basada en Windows, se deben establecer las propiedades de los objetos que se crean.

Propiedades habituales de los formularios La siguiente tabla describe algunas de las propiedades más habituales de los formularios que normalmente se establecen en el momento del diseño:

Propiedad	Descripción	Configuración predeterminada
(Nombre)	Establece el nombre del formulario en el proyecto (éste no es el nombre que se muestra al usuario en la barra de título, sino el nombre utilizado en el código para hacer referencia al formulario). Importante: Si se cambia la propiedad (Name) del formulario, debe establecer el objeto de inicio para su proyecto al nuevo nombre o el proyecto no se iniciará correctamente. Si desea más información sobre cómo cambiar el objeto de inicio, lea el apartado <i>Ciclo de vida del formulario</i> de este módulo.	Form1 (Form2, Form3, etc.)
AcceptButton	Establece en qué botón se hace clic cuando el usuario presiona la tecla ENTER. Nota: El formulario debe disponer de un botón como mínimo con esta propiedad.	Ninguna
CancelButton	Establece en qué botón se hace clic cuando el usuario presiona la tecla ESC. Nota: El formulario debe disponer de un botón como mínimo con esta propiedad.	Ninguna
ControlBox	Determina si un formulario muestra un cuadro de control en la barra de título. El cuadro de control puede contener los botones Minimizar , Maximizar , Ayuda y Cerrar .	True
FormBorderStyle	Controla el aspecto del borde del formulario. También afecta a cómo aparece la barra de título y qué botones incluye.	Sizable



(continuación)

Property	Descripción	Configuración predeterminada
MaximizeBox	Determina si un formulario dispone de un botón Maximizar en la esquina superior derecha de su barra de título.	True
MinimizeBox	Determina si un formulario dispone de un botón Minimizar en la esquina superior derecha de su barra de título.	True
StartPosition	Determina la posición de un formulario en la pantalla cuando aparece por primera vez.	WindowsDefaultLocation
Text	Establece el texto que se muestra en la barra de título del control.	Form1 (Form2, Form3, etc.)

Procedimiento:
establecer las
propiedades de un
formulario

Podemos establecer las propiedades de un formulario escribiendo código o utilizando la ventana de Propiedades. Las propiedades que establezcamos en el momento del diseño se utilizan como configuración inicial cada vez que la aplicación se ejecuta.

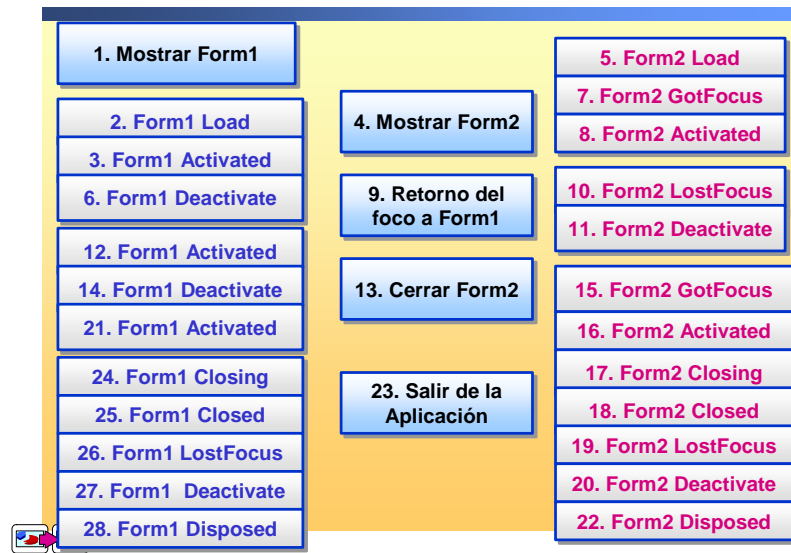
Para establecer las propiedades del formulario en tiempo de diseño:

1. Si la ventana Propiedades no está abierta, en el menú **Ver**, hacer clic en **Ventana Propiedades**.
2. En la vista de Diseño, hacer clic en el formulario para el que deseamos establecer una propiedad. El nombre del formulario aparece en la lista **Objeto** en la parte superior de la ventana Propiedades.
3. Utilizar los botones **Alfabético**  y **Por categorías**  de la ventana Propiedades para elegir las propiedades del formulario alfabéticamente o por categorías.
4. En la ventana **Propiedades**, hacer clic en la propiedad que se desea establecer.

Nota Cuando se selecciona una propiedad, aparece su descripción en la parte inferior de la ventana Propiedades, en el panel de descripción.

5. Escribir o seleccionar la propiedad deseada.

Ciclo de vida de un formulario



Introducción

Después de añadir los formularios necesarios a un proyecto y establecer el formulario de inicio, se debe determinar qué eventos y métodos utilizar. El ciclo de vida completo de un formulario utiliza varios métodos y eventos.

Eventos y métodos de los formularios

Cuando se invoca el método **Show()**, los eventos y métodos del formulario normalmente se invocan en el orden siguiente:

1. Load
2. GotFocus
3. Activated
4. Closing
5. Closed
6. Deactivate
7. LostFocus
8. Dispose()

New

El evento **Initialize** se utiliza normalmente para preparar una aplicación para su uso. Se asignan variables a sus valores iniciales, y puede cambiarse el tamaño o la posición de los controles para hospedar datos de inicialización.

En versiones anteriores de Visual Basic, el evento **Initialize** se utilizaba para ejecutar código antes de que un formulario se cargara. En .NET, el código de inicialización debe añadirse al constructor del formulario (**Sub New()**) después de la invocación de **InitializeComponent()** como se muestra en el siguiente ejemplo:

```
Public Sub New()  
    MyBase.New()  
    ' This call is required by the Windows Forms Designer.  
    InitializeComponent()  
    ' Add your initialization code here
```

Show

El método **Show** incluye un **Load** implícito; esto significa que si el formulario especificado no se ha cargado todavía cuando se invoca el método **Show**, la aplicación carga automáticamente el formulario en memoria y lo muestra al usuario. El método **Show** puede mostrar formularios como modales o no modales.

```
FrmSplash.Show()
```

Podemos utilizar el método **ShowDialog()** para mostrar un formulario como un cuadro de diálogo.

Load

El evento **Load** se utiliza para realizar acciones que deben ocurrir antes de que se muestre el formulario. También se utiliza para asignar valores predeterminados al formulario y sus controles.

El evento **Load** tiene lugar cada vez que un formulario se carga en memoria. El evento **Load** de un formulario puede ejecutarse múltiples veces durante la vida de una aplicación. Se ejecuta cuando un formulario se inicia como resultado de la instrucción **Load**, la instrucción **Show** o cuando se genera una referencia a las propiedades, métodos o controles de un formulario no cargado.

Activated/Deactivate

Cuando el usuario se mueve entre dos o más formularios, podemos utilizar los eventos **Activated** y **Deactivate** para definir el comportamiento de los formularios. El evento **Activated** tiene lugar cuando el formulario se activa mediante código o por el usuario. Para activar un formulario en tiempo de ejecución mediante código, se invoca el método **Activate**. Este evento puede utilizarse para tareas como la actualización del contenido del formulario en base a los cambios realizados en los datos del formulario cuando éste no estaba activado.

El evento **Activated** se ejecuta cuando el formulario recibe el foco de otro formulario del mismo proyecto. Este evento únicamente se ejecuta cuando el formulario es visible. Por ejemplo, un formulario cargado utilizando la instrucción **Load** no es visible a menos que se utilice el método **Show**, o se establezca la propiedad **Visible** del formulario como **True**. El evento **Activated** se ejecuta antes del evento **GotFocus**.

Utilice el siguiente código para establecer el foco a un formulario:

```
FrmSplash.Focus()
```

Deactivate se ejecuta cuando el formulario pierde el foco a otro formulario. Este evento se ejecuta después del evento **LostFocus**.

Ambos eventos, **Activated** y **Deactivate**, se ejecutan únicamente cuando cambia el foco dentro de la misma aplicación. Si cambiamos a otra aplicación y regresamos al programa, ninguno de los dos eventos se ejecuta.

Importante Si se necesita añadir código que se ejecute cuando el formulario se muestra o cuando el formulario está oculto, añadir el código a los controladores de eventos **Activated** y **Deactivate** en lugar de los controladores de eventos **GotFocus** y **LostFocus**.

Closing

El evento **Closing** resulta útil cuando se necesita saber cómo el usuario cierra el formulario. El evento **Closing** ocurre cuando el formulario recibe una solicitud de cierre. La validación de datos tiene lugar en ese momento. En caso de ser necesario mantener abierto el formulario (por ejemplo, si falla la validación de datos), el evento de cierre puede cancelarse.

Closed

El evento **Closed** ocurre cuando el formulario se cierra y antes del evento **Dispose**. El procedimiento del evento **Closed** se utiliza para verificar que el formulario debe cerrarse o para especificar acciones que deben tener lugar cuando se cierra el formulario. También se puede incluir código de validación a nivel de formulario para cerrarlo o para guardar datos en un archivo.

Dispose

El .NET Framework no soporta el evento **Terminate**. El código de terminación debe ejecutarse dentro del método **Dispose**, antes de invocar **MyBase.Dispose()**.

```
Public Overrides Sub Dispose(ByVal Disposing As Boolean)

    ' Termination code goes here.
    ' The following code was automatically added by the
    'Dispose method.
    MyBase.Dispose(Disposing)
    If Disposing Then
        If Not components Is Nothing Then
            components.Dispose(Disposing)
        End If
    End If
End Sub
```

El método **Dispose** es invocado automáticamente para el formulario principal de una aplicación y debe invocarse explícitamente para cualquier otro formulario.

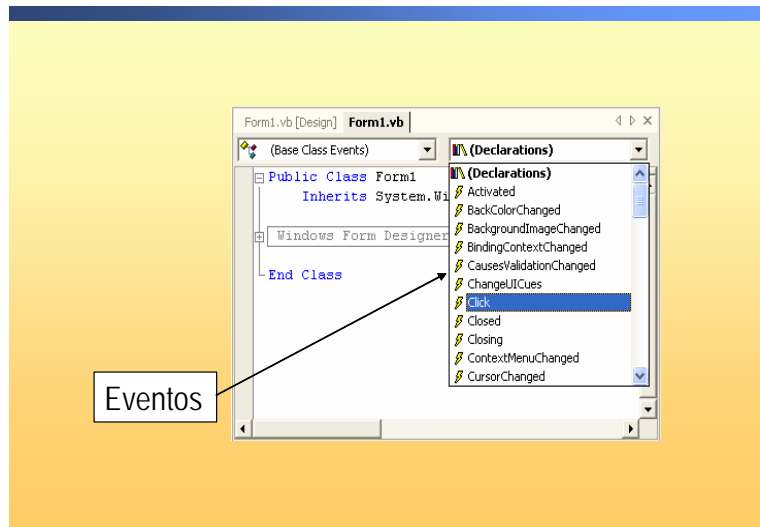
Hide

El método **Hide** elimina un formulario de pantalla pero no de la memoria. El usuario no puede acceder a los controles de un formulario oculto, pero sí están disponibles para la aplicación que se está ejecutando. Cuando un formulario está oculto, el usuario no puede interactuar con la aplicación hasta que finaliza la ejecución de todo el código del procedimiento de evento que ha ocultado el formulario.

Si el formulario no está cargado en memoria cuando se invoca el método **Hide**, el método **Hide** carga el formulario pero no lo muestra.

```
frmMyForm.Hide()
```

Cómo gestionar los eventos de un formulario



Introducción

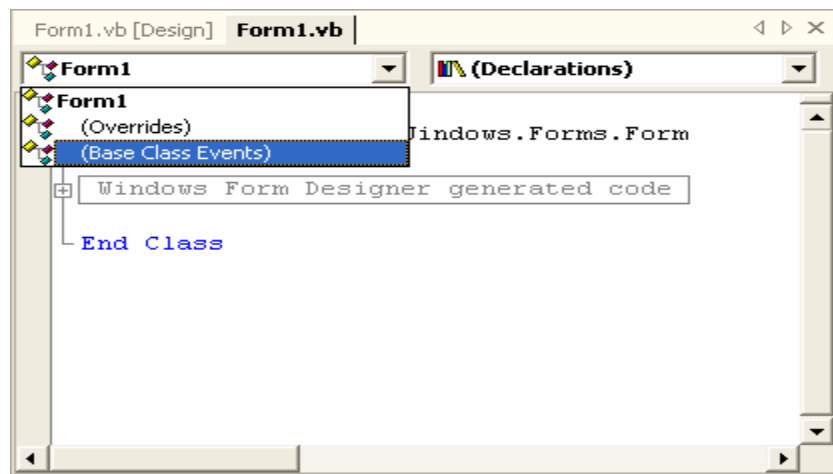
Un *controlador de eventos* es un segmento de código que se invoca cuando ocurre un evento correspondiente. Por ejemplo, podemos escribir código en un controlador de eventos para el evento **Activated** de un formulario y realizar operaciones como actualizar los datos que se muestran en los controles del formulario cuando está activado.

El .NET Framework utiliza una convención de nombres estándar para controladores de eventos. La convención consiste en combinar el nombre del objeto que envía el evento, un guión bajo, y el nombre del evento. Por ejemplo, el evento **Click** de un formulario denominado **Form1** se denominaría **Form1_Click**.

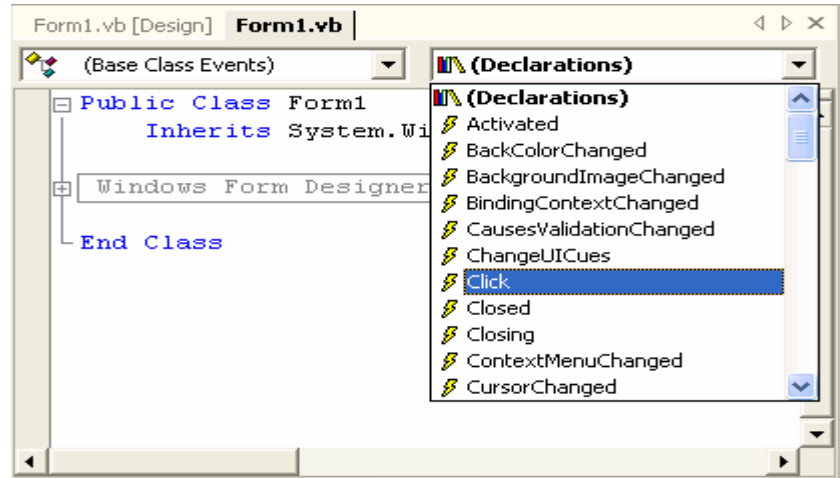
Procedimiento

Para abrir un controlador de eventos:

1. Abrir el Editor de código del formulario para el que desea añadir un controlador de eventos.
2. En el cuadro de lista **Nombre de clase**, hacer clic en **(Form 1 Events)**. La siguiente ilustración muestra el cuadro de lista **Nombre de clase** con **(Form 1 Events)** seleccionado.



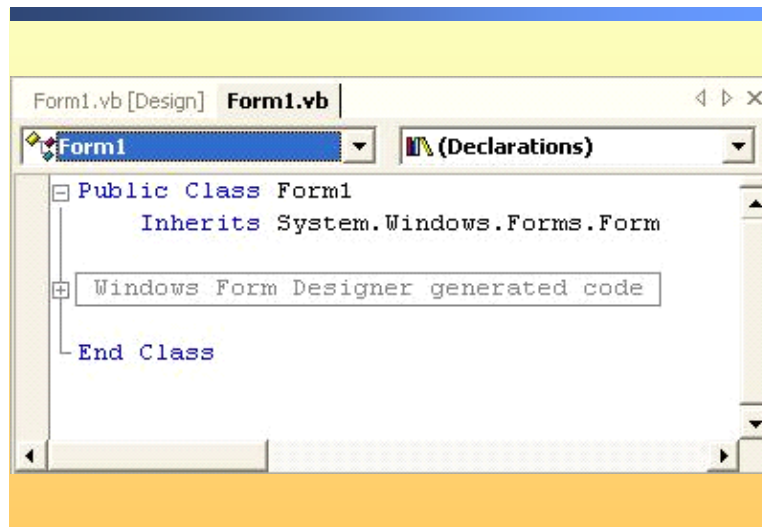
1. Hacer clic en la flecha para desplegar la lista del cuadro **Nombre de método** y ver los eventos disponibles para el formulario. La siguiente ilustración muestra el cuadro de lista **Nombre de método** con la lista de eventos del formulario y el evento **Click** seleccionado. Observar que el icono de Evento ⚡ a la izquierda del nombre indica que es un evento.



2. Hacer clic en el evento para añadir el controlador de eventos.

Estudiaremos más sobre el uso de eventos y controladores de eventos en el .NET Framework en el Módulo 2.

Código generado por el Diseñador de Windows Forms



Introducción

Cuando creamos un formulario utilizando el Diseñador de Windows Forms, el diseñador genera un bloque de código que deberíamos escribir si estuviésemos creando un formulario por nosotros mismos.

Código generado por el Diseñador

Si miramos el código predeterminado del formulario, encontraremos el siguiente código generado por el diseñador:

```
#Region " Windows Form Designer generated code "  
  
    Public Sub New()  
        MyBase.New()  
  
        'This call is required by the Windows Form Designer.  
        InitializeComponent()  
  
        'Add any initialization after the  
        InitializeComponent() call  
  
    End Sub  
  
    'Form overrides dispose to clean up the component list.  
    Protected Overloads Overrides Sub Dispose(ByVal disposing  
As Boolean)  
        If disposing Then  
            If Not (components Is Nothing) Then  
                components.Dispose()  
            End If  
        End If  
        MyBase.Dispose(disposing)  
    End Sub  
  
    'Required by the Windows Form Designer  
    Private components As System.ComponentModel.IContainer  
  
    'NOTE: The following procedimiento is required by the  
    Windows Form Designer  
    'It can be modified using the Windows Form Designer.  
    'Do not modify it using the code editor.  
    <System.Diagnostics.DebuggerStepThrough()> Private Sub  
    InitializeComponent()  
        components = New System.ComponentModel.Container()  
        Me.txt  
  
    End Sub  
  
#End Region
```

**Partes del código
generado****■ InitializeComponent**

Este código es utilizado por el entorno de desarrollo para que persistan los valores de propiedades que establezcamos en el Diseñador de Windows Forms. En versiones anteriores de Visual Basic, esta información no se guardaba en forma de código, sino como instrucciones textuales en la parte superior del archivo .frm que siempre quedaban ocultas al desarrollador.

■ Public Sub New()

Es el constructor de clases. Aunque podemos insertar aquí código de inicialización para el formulario, deberíamos situarlo en el evento **Load** del formulario (a menos que tenga necesariamente que estar aquí).

Importante Evite modificar o eliminar el código generado por el Diseñador de Windows Forms. Modificar o eliminar este código puede generar errores en su proyecto.

Lección: añadir controles a un formulario

- Cómo añadir controles a un formulario
- Cómo añadir menús a un formulario
- Cómo personalizar los controles del Cuadro de herramientas
- Práctica: crear un formulario y añadir controles

Introducción

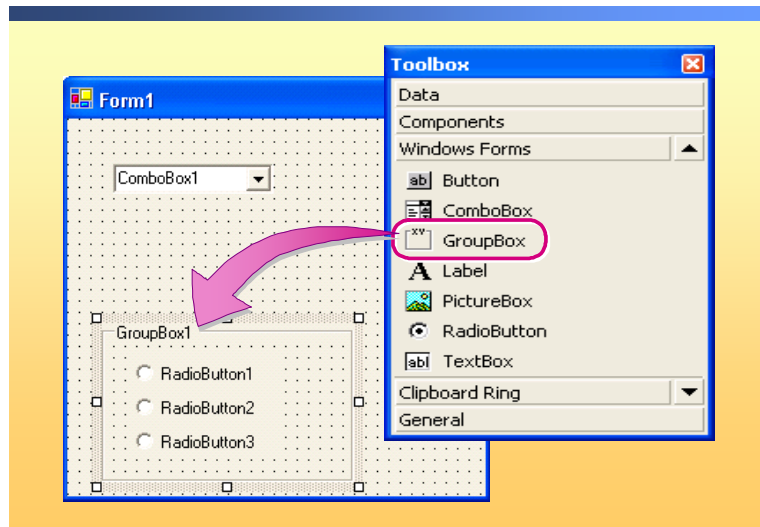
Para crear un interfaz de usuario para una aplicación, debemos añadir controles a un formulario. Esta lección explica cómo añadir controles a un formulario.

Objetivos de la lección

En esta lección, aprenderá a:

- Añadir controles a un formulario.
- Añadir menús a un formulario.
- Personalizar los controles del Cuadro de herramientas.

Cómo añadir controles a un formulario



Introducción

Los controles son objetos contenidos en los objetos formulario. Botones, cuadros de texto y etiquetas son algunos ejemplos de controles.

Procedimiento: Añadir controles a un formulario

Existen dos formas de añadir controles a un formulario. La primera permite añadir varios controles rápidamente y, a continuación, establecer su tamaño y posición individualmente. La segunda forma, ofrece mayor control inicial sobre el tamaño y posición de los controles.

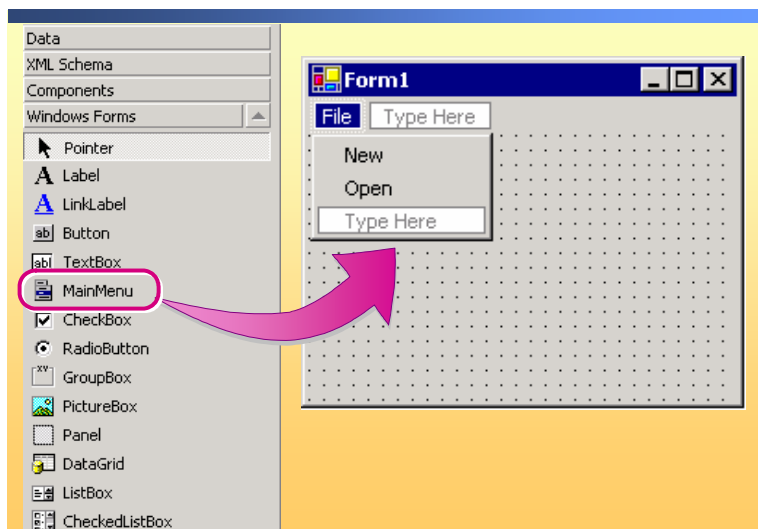
Para añadir controles a un formulario y, a continuación, establecer su tamaño y posición:

1. Si el Cuadro de herramientas no está abierto, en el menú **Ver**, hacer clic en **Cuadro de herramientas**.
2. En el Cuadro de herramientas, hacer doble clic en el control que desea añadir. Se insertará una instancia del control con el tamaño predeterminado en la esquina superior izquierda del objeto activo. Cuando se añaden varios controles de este modo, se insertan uno encima de otro.
3. Una vez añadidos los controles, podemos cambiar su posición y tamaño:
 - a. Para cambiar la posición del control, hacer clic sobre el control para seleccionarlo y arrastrarlo a la posición deseada.
 - b. Para cambiar el tamaño del control, hacer clic sobre el control para seleccionarlo y arrastrar uno de los ocho puntos para modificar el tamaño hasta alcanzar el adecuado.

Para cambiar el tamaño y posición de los controles mientras los añadimos a un formulario:

1. Si el Cuadro de herramientas no está abierto, en el menú **Ver**, hacer clic en **Cuadro de herramientas**.
2. En el Cuadro de herramientas, hacer clic en el control que desea añadir.
3. Mover el cursor del ratón sobre el formulario. El símbolo del cursor cambia a una cruz.
4. Posicionar la cruz donde desee que aparezca la esquina superior izquierda del control.
5. Hacer clic y arrastrar la cruz donde deseemos que se ubique la esquina inferior derecha. Se dibujará en la pantalla un rectángulo que indica el tamaño y ubicación del control.
6. Cuando el control tenga el tamaño adecuado, liberar el botón del ratón. El control aparece en la ubicación correcta en el formulario.
7. Podemos cambiar el tamaño y la posición del control después de liberar el botón del ratón:
 - a. Para cambiar la posición del control, hacer clic en el control para seleccionarlo y arrastrarlo a la posición adecuada.
 - b. Para cambiar el tamaño del control, hacer clic en el control para seleccionarlo y arrastrar uno de los ocho tiradores de tamaño hasta conseguir el tamaño deseado.

Cómo añadir menús a un formulario



Introducción

Los menús proporcionan a los usuarios un modo estructurado para acceder a los comandos y herramientas que contiene una aplicación. Una planificación y diseño adecuados de los menús y de las barras de herramientas son esenciales y garantizan a los usuarios una correcta funcionalidad y accesibilidad de la aplicación.

Un control de menú tiene numerosas propiedades, como **Name**, **Caption** e **Index**.

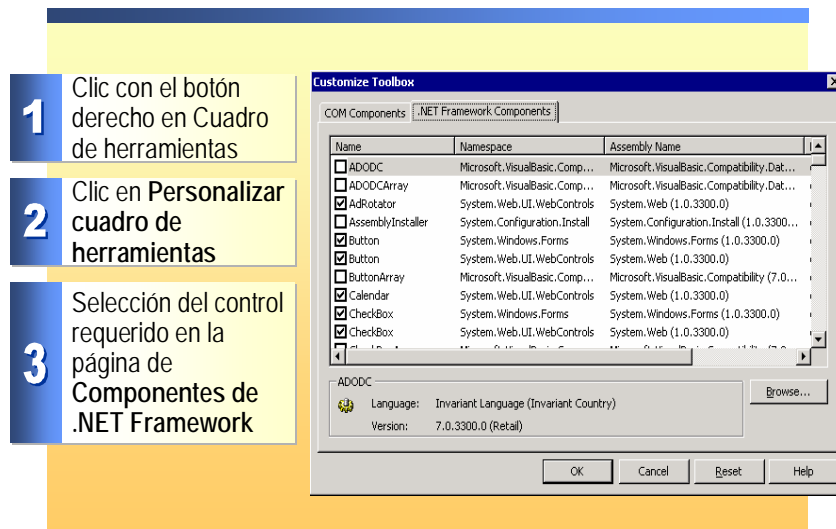
- La propiedad **Name** identifica el control de menú en código.
- La propiedad **Index** identifica controles que comparten el mismo nombre.
- La propiedad **Caption** es el texto que aparece en la barra de menú en tiempo de ejecución.

Procedimiento: Añadir menús a un formulario

Para añadir menús a un formulario:

1. Si el Cuadro de herramientas no está abierto, en el menú **Ver**, hacer clic en **Cuadro de herramientas**.
2. En el Cuadro de herramientas, hacer doble clic en el control **MainMenu**.
3. En el cuadro **Caption** del menú recién creado, escribir el texto del título del primer menú. Este título aparecerá en la barra del menú.
4. En el cuadro **Nombre**, en la ventana Propiedades, escribir el nombre que utilizará para hacer referencia al control de menú en código.

Cómo personalizar los controles del Cuadro de herramientas



Introducción

El Cuadro de herramientas muestra una variedad de elementos para utilizar en proyectos de Visual Studio .NET. Los elementos disponibles incluyen componentes .NET, componentes COM (Component Object Model), objetos HTML (Hypertext Markup Language), fragmentos de código y texto. El Cuadro de herramientas contiene diversos controles que podemos utilizar para añadir ilustraciones, etiquetas, botones, cuadros lista, barras de desplazamiento, menús y formas geométricas a un interfaz de usuario.

Todos los controles que se añaden a un formulario se convierten en objetos del interfaz de usuario programable de la aplicación. Estos objetos están visibles para el usuario cuando la aplicación se ejecuta, y funcionan como objetos estándares de cualquier aplicación basada en Windows. Sin embargo, existen algunos controles que están visibles en el Cuadro de herramientas de modo predeterminado. Debemos personalizar el Cuadro de herramientas para mostrar dichos controles el mismo. Por ejemplo, el control **StatusBarPanel** no está visible en el Cuadro de herramientas de forma predeterminada.

Podemos personalizar el Cuadro de herramientas añadiendo y eliminando elementos. El cuadro de diálogo **Personalizar el Cuadro de herramientas** muestra listas tabuladas de componentes que se reconocen en el equipo. Utilice el cuadro de diálogo **Personalizar el Cuadro de herramientas** para añadir o eliminar controles del Cuadro de herramientas.

Nota El cuadro de diálogo **Personalizar el Cuadro de herramientas** sustituye al cuadro de diálogo **Componentes** de versiones anteriores de Visual Basic.


Procedimiento:
personalizar los
controles del Cuadro de
herramientas

Para personalizar los Controles del Cuadro de herramientas:

1. Hacer clic con el botón derecho en el Cuadro de herramientas.
2. Hacer clic en **Agregar o quitar elementos**.
3. Hacer clic en la pestaña **Componentes de .NET Framework** o en la pestaña **Componentes COM**, y seleccionar los controles deseados.

Nota Podemos añadir fragmentos de código al Cuadro de herramientas seleccionando el fragmento de código o el texto y arrastrándolo al Cuadro de herramientas. Aparecerá una nueva entrada empezando por **Texto:** en los Controles del Cuadro de herramientas.


Practica: crear un formulario y añadir controles



En esta práctica,

- Estableceremos las propiedades del formulario
- Añadiremos controles al formulario
- Estableceremos las propiedades de los controles
- Implementaremos el controlador de eventos del botón Click

Empezar revisando los objetivos de esta actividad práctica

10 min 

Introducción

En esta práctica, abriremos un proyecto existente y modificaremos las propiedades del formulario predeterminado. También añadiremos controles al formulario e implementaremos el evento **Click** para dos botones.

Instrucciones

🔗 Abrir el archivo del proyecto para esta actividad práctica

1. Utilizar el Explorador de Windows Explorer e ir a la carpeta pract01\Starter. Esta carpeta se puede encontrar dentro del fichero practs06.zip.
2. Hacer doble clic en el archivo de solución CreatingForms.sln para abrir el proyecto.

🔗 Establecer las propiedades de un formulario

1. Abrir el archivo CreatingForms.vb en la vista de Diseño.
2. Si la ventana Propiedades no está visible, en el menú **Ver**, hacer clic en **Ventana Propiedades**.
3. Establecer las propiedades restantes del formulario con los valores indicados.

Propiedad	Valor
ControlBox	false
Font	Trebuchet MS, 10pt
FormBorderStyle	Fixed3D
Size	300, 175
Text	Hello World

✚ Añadir controles al formulario

1. Si el Cuadro de herramientas no está visible, en el menú **Ver**, hacer clic en **Cuadro de herramientas**.
2. En el Cuadro de herramientas, hacer doble clic en el control **Label** para añadirlo al formulario.
3. Hacer doble clic en el control **Button** para añadirlo al formulario.
4. Hacer doble clic en el control **Button** de nuevo para añadir un segundo botón al formulario.
5. Posicionar el control **Label** cerca de la parte central superior del formulario.
6. Posicionar los botones uno junto a otro cerca de la parte inferior del formulario, **button1** a la izquierda y **button2** a la derecha.

✚ Establecer las propiedades de los controles

1. Hacer clic en el control **Label1**.
2. Establecer las siguientes propiedades para el control **Label1** con los valores especificados a continuación:

Propiedad	Valor
(Name)	OutputLabel
BorderStyle	Fixed3D
Font	Trebuchet MS, 10pt, Bold
ForeColor	ActiveCaption
Location	14,30
Size	264, 23
Text	(Borrar texto existente y dejarlo en blanco)
TextAlign	MiddleCenter

3. Hacer clic en **Button1**.
4. Establecer las siguientes propiedades para el control **Button1** con los valores que se indican en la siguiente tabla:

Propiedad	Valor
(Name)	HelloButton
Location	57, 87
Size	75, 25
Text	&Say Hello

5. Hacer clic en **Button2**.
6. Establecer las siguientes propiedades para el control **Button2** con los valores que se indican en la siguiente tabla:

Propiedad	Valor
(Name)	ExitButton
Location	161, 87
Size	75, 25

Text**E&xit**

7. Hacer doble clic en el botón **Say Hello** para crear el controlador de eventos **Click**.
8. En el controlador de eventos **Click** de **HelloButton**, añadir la siguiente línea de código:

```
OutputLabel.Text = "Hello, World!"
```

9. Regresar a la vista de Diseño del formulario.
10. Hacer clic con el botón derecho en el botón **Exit** para crear el controlador de eventos **Click**.
11. En el controlador de eventos **Click** de **ExitButton**, añadir la siguiente línea de código:

```
Me.Close()
```

12. Regresar a la vista de Diseño del formulario.
13. Establecer la propiedad **AcceptButton** para **HelloButton** y la propiedad **CancelButton** para **ExitButton**.

🔍 Generar y ejecutar la aplicación

1. Para generar la aplicación, hacer clic en el menú **Generar** y clic en **Generar solución**.
2. Ejecutar la aplicación presionando F5.

Lección: crear un formulario heredado

- Modificadores de acceso
- Cómo crear un formulario heredado
- Práctica: crear un formulario heredado

Introducción

Crear nuevos formularios Windows Forms heredando desde formularios base es una manera eficaz de duplicar el trabajo realizado sin necesidad de completar todo el proceso de volver a crear un formulario por completo cada vez que lo necesitemos. El proceso de heredar un formulario de uno existente se denomina *herencia visual*. Esta lección explica cómo heredar de un formulario existente utilizando la herencia visual.

Objetivo de la lección

En esta lección, aprenderá a crear un formulario heredado.

Modificadores de acceso

Modificador de acceso	Descripción
Private	De sólo lectura para un formulario secundario, todos sus valores de propiedades en la ventana Propiedades están deshabilitados
Protected	Accesible desde dentro de la clase y desde cualquier clase que herede de la clase que declaró este miembro
Public	Nivel más permisivo; los controles Public tienen accesibilidad total

Introducción

No sólo se pueden heredar controles y propiedades de un formulario base, sino también código. Esto significa que se puede crear una librería de código para facilitar la reutilización de código. La principal ventaja del uso de la herencia es que podemos sobrescribir el código si no es aplicable a una circunstancia en particular.

Para poder sobrescribir un evento en el formulario secundario, debe hacerlo visible para el secundario. Esto significa que debe estar definido como **Public** o **Protected** (*protected* es el valor predeterminado para eventos), y su palabra clave **Modifiers** no puede ser privada.

Modificadores de acceso

La propiedad **Modifiers** determina el nivel de accesibilidad de un control como el modo en que se comporta el control y cuál es su funcionalidad cuando su formulario se utiliza como formulario base. Un formulario heredado muestra los controles en diferente escala de gris dependiendo del valor de esta propiedad. Algunos de los valores de la propiedad **Modifiers** incluyen **Public**, **Private** y **Protected**.

Los valores de las propiedades del control son las mismas que las del objeto principal, y cuando se modifican en el formulario secundario, esa propiedad aparece en negrita en la ventana Propiedades. Para restablecer todos los valores a los guardados por el formulario principal, hacer clic con el botón derecho en la ventana Propiedades y clic en **Reset**.

■ **Private**

Un control **Private** es de sólo lectura para un formulario secundario. Debido a que un control **Private** no puede ser modificado, todos los valores de sus propiedades en la ventana Propiedades están deshabilitados. Copiar este control en otro lugar del formulario o proyecto produce una versión totalmente editable.

■ **Protected**

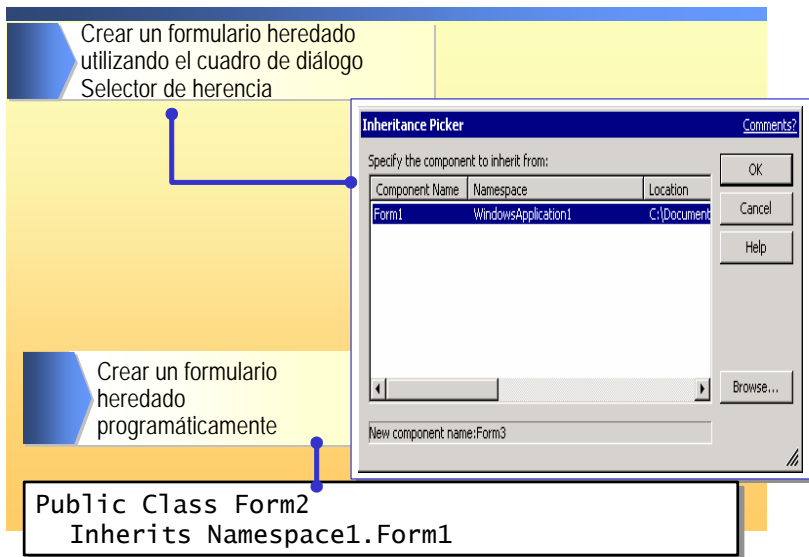
Un miembro *protected* es accesible desde dentro de la clase y desde cualquier clase que herede de la clase que declaró este miembro. Si se modifica un control **Protected** en un formulario secundario, los cambios permanecerán aunque se realice un cambio en el formulario principal. Para los otros tipos de controles, los cambios en el formulario principal invalidarán los realizados al secundario.

■ **Public**

Public es el nivel más permisivo. Los controles **Public** tienen accesibilidad total.

Nota Los valores **Public**, **Protected** y **Private** de las propiedades son los tres valores comunes en todos los proyectos de lenguajes .NET. Visual Basic .NET soporta otros dos valores: **Friend** y **Protected Friend**.

Cómo crear un formulario heredado



Introducción

La herencia visual ofrece numerosas ventajas a los desarrolladores. Si hemos diseñado un formulario para otro proyecto que es similar al que necesitamos para el proyecto en el que estamos trabajando, podemos heredar del primer formulario. Por tanto, también podemos crear un formulario base como plantilla para utilizar en el futuro. Es una forma útil de duplicar las principales funcionalidades de determinados formularios sin necesidad de volver a crearlos desde el principio. Los cambios realizados en el formulario base quedarán reflejados en los formularios que hereden de él; por tanto, los cambios en el formulario plantilla subyacente modificarán todos los formularios basados en ella.

Procedimiento: heredar de un formulario existente

Existen dos formas de implementar la herencia visual en un proyecto Visual Studio .NET.

Para crear un formulario heredado programáticamente:

1. Crear un nuevo proyecto en Visual Studio .NET.
2. Añadir otro formulario, o visualizar el código de Form1, creado por defecto.
3. En la definición de clase, añadir una referencia al formulario del que se va a heredar. La referencia debería incluir el espacio de nombres que contiene el formulario, seguido por un punto y el nombre del formulario base.

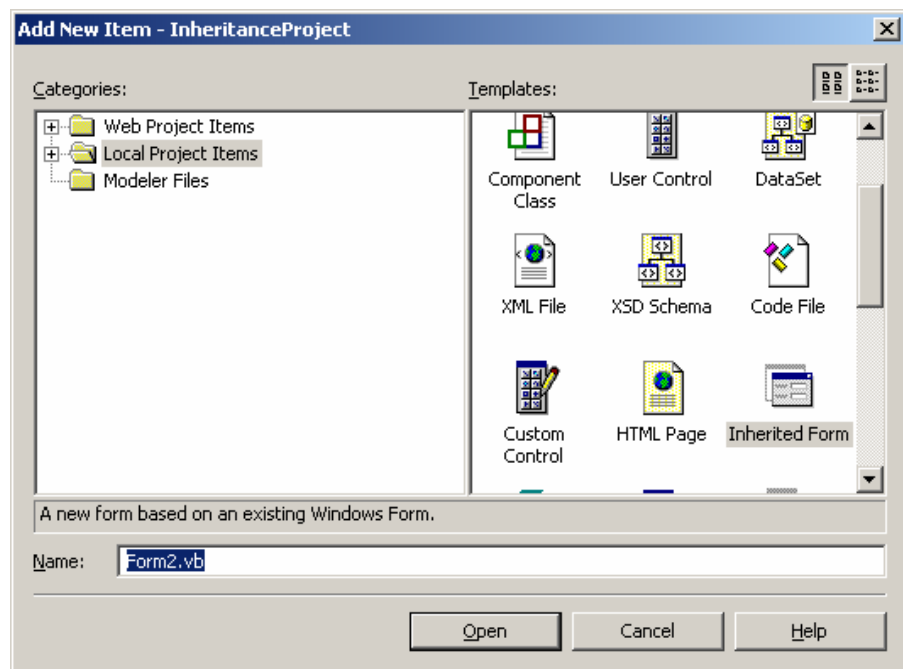
```
Public Class Form2
    Inherits Namespace1.Form1
```

El formulario adquiere las características del formulario heredado. Contiene los controles del formulario heredado, el código y las propiedades.

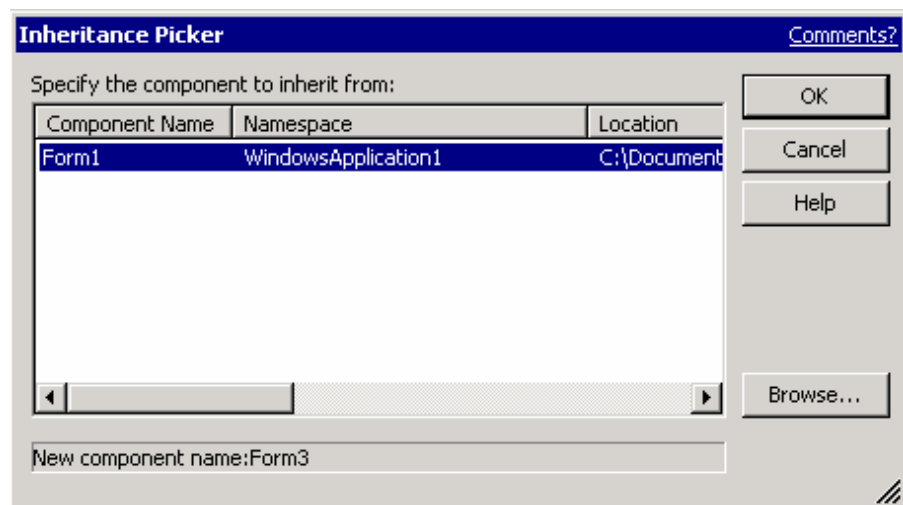
Para crear un formulario heredado utilizando el cuadro de diálogo **Selector de herencia**:

Nota Asegurémonos de generar la solución antes de heredar de un formulario existente en el proyecto.

1. En el menú **Proyecto**, hacer clic en **Agregar formulario heredado**. El cuadro de diálogo es idéntico para C# y para Visual Basic.



2. En el panel de Categorías, hacer clic en **Elementos de proyecto local**, y en el panel de Plantillas, hacer clic en **Formulario heredado**. En el cuadro **Nombre**, escribir un nombre y a continuación hacer clic en **Abrir**. Se abrirá el cuadro de diálogo **Selector de herencia**.




-
3. Hacer clic en **Examinar**, y localice el ejecutable compilado del proyecto que contiene su formulario. Hacer clic en **Aceptar**.

El nuevo formulario debería añadirse al proyecto y estar basado en su formulario heredado. Contiene los controles del formulario base.

Importante Después de que el nuevo formulario se haya añadido y se haya heredado del formulario base, el proyecto debe generarse de nuevo para completar las relaciones de herencia. El nuevo formulario puede entonces mostrarse en la vista de Diseño.

Práctica: crear un formulario heredado




En esta práctica,

- Estableceremos las propiedades de los controles del formulario base para prepararlos para la herencia
- Añadiremos un nuevo formulario al proyecto heredándolo del formulario base
- Estableceremos las propiedades en el formulario heredado y los controles

Empezar revisando los objetivos de esta actividad práctica

10 min



Introducción

En esta práctica, modificaremos un formulario existente para permitir que sea heredado por otros formularios. Crearemos un nuevo formulario y lo heredaremos del formulario base. A continuación, modificaremos el formulario heredado para personalizarlo en una determinada aplicación.

Instrucciones

📌 Abrir el proyecto de la práctica

1. Utilizar Windows Explorer e ir a la carpeta pract02\Starter. Esta carpeta se puede encontrar en el fichero practs06.zip.
2. Hacer doble clic en WindowsCalculator.sln para abrir el proyecto.

📌 Modificar las propiedades de un formulario

1. Si el Explorador de soluciones no está visible, en el menú **Ver**, hacer clic en **Explorador de soluciones**.
2. Si la ventana Propiedades no está visible, en el menú **Ver**, hacer clic en **Ventana Propiedades**.
3. Abrir BaseAboutForm.vb en la vista de Diseño.
4. Hacer clic en la etiqueta **Product Name**, y asegurémonos de que la propiedad **Modifiers** está configurada como **Protected**.
5. Asegurémonos de que la propiedad **Modifiers** está configurada según los valores dados a los siguientes controles.

Control	Propiedad Modifiers
Etiqueta Version <1.0.0000>	Protected
Etiqueta Copyright © yyyy Contoso, Ltd.	Protected
Etiqueta All Rights Reserved	Protected
Botón Aceptar	Protected

6. Guarde el proyecto.

🔍 Añadir un formulario heredado

1. En el menú **Proyecto**, hacer clic en **Agregar nuevo elemento**.
2. En el panel Categorías, hacer clic en **Elementos de proyecto local**.
3. En el panel de Plantillas, hacer clic en **Formulario heredado**.
4. En el campo **Nombre**, escribir **AboutForm.vb** y hacer clic en **Abrir** para abrir el formulario.
5. En el cuadro de diálogo **Selector de herencia**, seleccionar **BaseAboutForm**.
6. Generar el proyecto.
Esto crea la relación de herencia entre ambos formularios, incluyendo los modificadores de acceso para los formularios y controles.
7. Abrir **AboutForm.vb** e ir a la vista de Diseño.
8. Establecer la propiedad **Size** del formulario' a **504,216**.
9. Establecer la propiedad **BackColor** del formulario como **Control**.
10. Establecer la propiedad **Text** como **About Simple Windows Calculator**.
11. Hacer clic en la etiqueta **Product Name**. Establecer la propiedad **Text** como **Simple Windows Calculator**.
12. Establecer las propiedades para los siguientes controles.

Control	Propiedades
Tag Version <1.0.0000>	Text: Version 2.53.1892
Tag Copyright © yyyy Contoso, Ltd.	Text: Copyright © 2002 Contoso, Ltd.

⚡ Implementar el elemento de menú *About* y probar la aplicación

1. Abrir el código fuente de `CalcUI.vb`.
2. En el menú **Ver**, hacer clic en **Mostrar tareas**, y hacer clic en **Comentario**.
3. En la ventana de Lista de tareas, localizar el comentario TODO. Añadir código para mostrar la ventana *About* cuando se hace clic en el elemento de menú:

```
Dim aboutForm As New AboutForm()  
aboutForm.ShowDialog()
```

4. Generar y ejecutar la aplicación.
5. Cuando aparezca la ventana Simple Calculator, en el menú de la **Ayuda**, hacer clic en **About**. Aparecerá el cuadro **About** heredado. Observe que el fondo del botón **System Info** sigue siendo el fondo establecido en el formulario base, pero el resto de controles tienen el color de fondo establecido en el formulario heredado.

Lección: organizar controles en un formulario

- Cómo organizar controles en un formulario utilizando el menú **Formato**
- Cómo establecer el orden de tabulación de los controles
- Cómo delimitar (Anchor) un control en Windows Forms
- Cómo acoplar (Dock) un control en Windows Forms
- Demostración: organizar controles en un formulario

Introducción

Es habitual que deseemos cambiar la posición y dimensiones de los controles en tiempo de ejecución. Actualmente, utilizamos el evento **resize** (WM_SIZE para los desarrolladores que utilizan APIs (Application Programming Interface)), calculamos la nueva posición, ancho y alto, e invocamos algunos métodos como **Move** o **SetWindowPos**. La biblioteca de Windows Forms ofrece dos conceptos muy útiles para simplificar estos procedimientos: delimitar y acoplar. Además, Visual Studio .NET proporciona el menú **Formato**, que nos permite organizar los controles de un formulario.

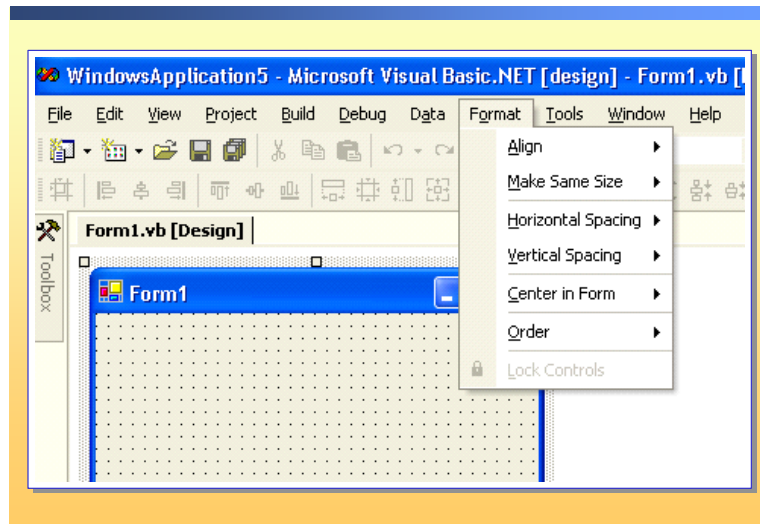
Esta lección explica el menú **Formato** y cómo delimitar y acoplar controles en un formulario. También explica cómo establecer el orden de tabulación para controles.

Objetivos de la lección

En esta lección, aprenderá a:

- Organizar controles en un formulario utilizando el menú **Formato**.
- Establecer el orden de tabulación para los controles de un formulario.
- Delimitar controles de un formulario.
- Acoplar controles en un formulario.

Cómo organizar los controles de un formulario utilizando el menú Formato



Introducción

Podemos utilizar el menú **Formato** o la barra de Presentación del entorno de desarrollo integrado (IDE) de Visual Studio para alinear, disponer en capas y bloquear los controles de un formulario.

Opciones del menú Formato

El menú **Formato** ofrece numerosas opciones para organizar los controles. Cuando utilizemos las opciones del menú **Formato** para organizar controles, es conveniente seleccionarlos de forma que el último control seleccionado sea el control primario respecto al cual se alineará el resto. Los cuadros de tamaño alrededor del perímetro del control primario son de color oscuro, mientras que los cuadros de tamaño del resto de controles son de color claro.

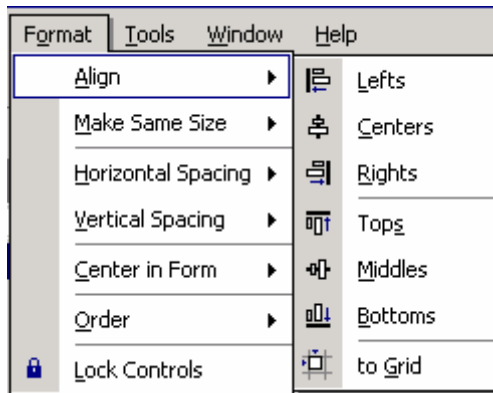
La siguiente tabla muestra las opciones y sus funciones:

Opción	Descripción
Alinear	Alinea todos los controles respecto al control primario
Igualar tamaño	Cambia el tamaño de múltiples controles de un formulario
Espaciado horizontal	Incrementa el espaciado horizontal entre controles
Espaciado vertical	Incrementa el espaciado vertical entre controles
Centrar en el formulario	Centra los controles de un formulario
Ordenar	Dispone en capas los controles de un formulario
Bloquear controles	Bloquea todos los controles de un formulario

Procedimiento:
organizar los controles
de un formulario

Para alinear varios controles:

1. En el Diseñador de Windows Forms, abrir el formulario que contiene los controles que desea posicionar.
2. Seleccionar los controles que deseamos alinear de forma que el último control que seleccione sea el control primario respecto del que se alineará el resto.
 - En el menú **Formato**, seleccionar **Alinear**, y hacer clic en cualquiera de las siete opciones disponibles.



Cuando creamos complejos interfaces de usuario, es posible que desee disponer en capas los controles de un formulario. Para disponer en capas los controles de un formulario:

1. Seleccionar un control.
2. En el menú **Formato**, seleccionar **Ordenar** y hacer clic en **Traer al frente** o **Enviar al fondo**.

Puede bloquear todos los controles de un formulario. De este modo, se evita mover o cambiar accidentalmente el tamaño de los controles mientras se están estableciendo otras propiedades. Para bloquear todos los controles de un formulario, en el menú **Formato**, hacer clic en **Bloquear controles**.

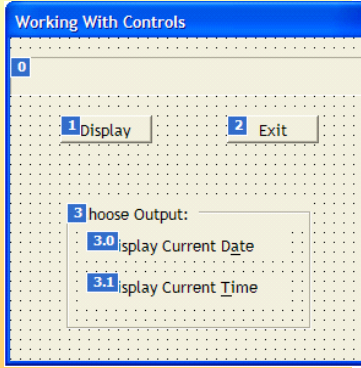
Cómo establecer el orden de tabulación de los controles

■ Para establecer el orden de tabulación de los controles

- En el menú **Ver**, seleccionar **Orden de tabulación**
- Hacer clic en un control para cambiar su orden de tabulación

-- 0 --

- Establecer la propiedad **TabIndex**
- Configurar la propiedad **TabStop** como **True**



Introducción

El orden de tabulación es el orden en que el usuario mueve el foco de un control a otro presionando la tecla TAB. Cada formulario tiene su propio orden de tabulación. Por defecto, el orden de tabulación es el mismo que el orden en que se crearon los controles. La numeración del orden de tabulación empieza por cero.

Procedimiento: establecer el orden de tabulación

Podemos establecer el orden de tabulación en la ventana Propiedades utilizando la propiedad **TabIndex**. La propiedad **TabIndex** de un control determina su posición en el orden de tabulación. De forma predeterminada, el valor de TabIndex del primer control que se dibuja es **0**, el valor de TabIndex del segundo es **1**, etc.

Para establecer el orden de tabulación utilizando el menú **Ver**:

1. En el menú **Ver**, hacer clic en **Orden de tabulación**.
2. Hacer clic en los controles secuencialmente para establecer el orden de tabulación deseado.
3. Al finalizar, en el menú **Ver**, hacer clic en **Orden de tabulación**.

Para establecer el orden de tabulación utilizando la propiedad **TabIndex**:

1. Seleccionar el control.
2. Establecer el valor requerido para la propiedad **TabIndex**.
3. Configurar la propiedad **TabStop** como **True**.

Deshabilitando la propiedad **TabStop**, el control es ignorado en el orden de tabulación del formulario. Un control en el que la propiedad **TabStop** se configure como **False** mantiene su posición en el orden de tabulación, aunque sea omitido al recorrer los controles con la tecla TAB.

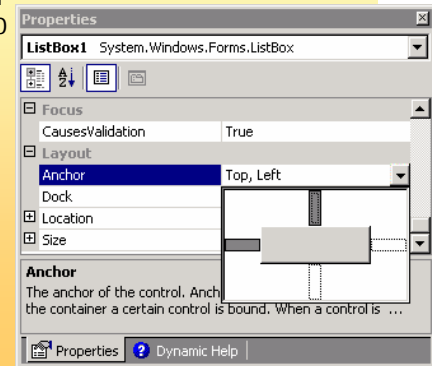
Como delimitar un control en Windows Forms

■ Delimitar

- Garantiza que los bordes del control permanecen en la misma posición respecto al contenedor principal

■ Delimitar un control al formulario

- Establecer su propiedad **Anchor**
- Valor predeterminado: **Superior, Izquierda**
- Otros valores: **Inferior, Derecha**



Introducción

Si estamos diseñando un formulario cuyo tamaño puede ser modificado por el usuario en tiempo de ejecución, el tamaño y posición de los controles del formulario deberían ajustarse correctamente. Cuando se delimita un control en un formulario (u otro contenedor) y se cambia el tamaño del formulario, el control mantiene la distancia entre el control y las posiciones delimitadas (la posición inicial). Para delimitar, utilizamos el editor de la propiedad **Anchor**.

Procedimiento: delimitar un control en un formulario

Para delimitar un control en un formulario:

1. Seleccionar el control que desea delimitar.
2. En la ventana Propiedades, hacer clic en la propiedad **Anchor**, y hacer clic en la flecha **Anchor**.

Se mostrará el editor de la propiedad **Anchor**; contiene una barra superior, barra izquierda, barra derecha y barra inferior.

3. Para establecer una delimitación, hacer clic en las barras superior, izquierda, derecha o inferior del editor de la propiedad **Anchor**. Los controles están delimitados hacia arriba y hacia la izquierda de forma predeterminada. Para borrar un lado delimitado del control, hacer clic en la barra correspondiente de ese lado.

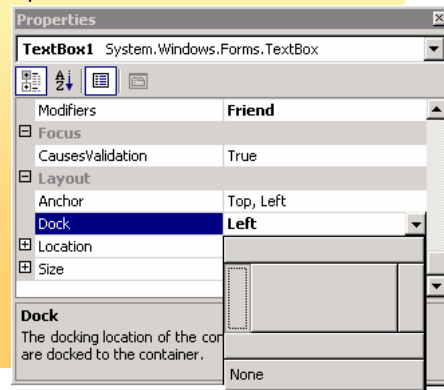
Cómo acoplar un control en Windows Forms

■ Acoplar

- Permite pegar los bordes de un control a los bordes de su control principal

■ Acoplar un control

- Establecer la propiedad **Dock**



Introducción

Podemos acoplar controles a los bordes de un formulario. Por ejemplo, Windows Explorer acopla el control **TreeView** al lado izquierdo de la ventana y el control **ListView** al lado derecho de la ventana. Utilice la propiedad **Dock** para todos los controles visibles de formularios Windows Forms para definir el modo de acoplamiento.

Cuando utilizamos la propiedad **Dock**, se acopla un control a dos bordes del contenedor. A continuación, el tamaño del control cambia horizontal o verticalmente cuando cambia el tamaño del contenedor. En la vida real, no indicamos los márgenes, indicamos un borde. Si acoplamos un control al borde izquierdo, se conecta con el borde superior izquierdo e inferior izquierdo. El valor de la propiedad **Dock** es uno de los valores **DockStyle**.

Un caso especial es **DockStyle.Fill**. Este valor acopla el control a todos los bordes. El control rellena toda el área cliente del contenedor.

Procedimiento: acoplar un control en un formulario

Para acoplar un control en un formulario:


1. Seleccionar el control que desea acoplar.
2. En la ventana Propiedades, hacer clic en la flecha a la derecha de la propiedad **Dock**.

Se mostrará el editor de la propiedad **Dock**; contiene una serie de botones que representan los bordes y el centro del formulario.

3. Hacer clic en el botón que representa el borde del formulario donde desea acoplar el control. Para rellenar el contenido del formulario del control o del control contenedor, hacer clic en el botón **Fill** (centro). Hacer clic en **None** para deshabilitar el acoplamiento.

El tamaño del control cambia automáticamente para ajustarse a los límites del borde acoplado.

Demostración: organizar controles en un formulario



En esta demostración, veremos cómo

- Alinear los controles de un formulario
- Disponer en capas los controles de un formulario
- Delimitar controles en un formulario
- Acoplar controles en un formulario

Introducción

En esta demostración, aprenderemos a cómo organizar controles en un formulario.

Instrucciones

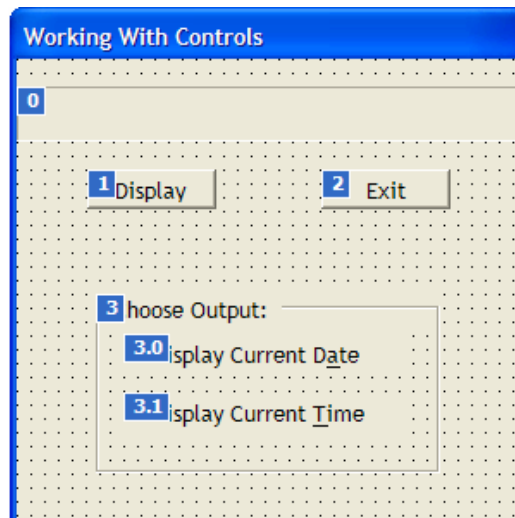
✚ Organizar controles utilizando el menú Formato

1. Abrir el proyecto WorkingWithControls de Visual Studio .NET en la carpeta Starter dentro del fichero demos06.zip.
2. Si el formulario WorkingWithControls.vb no está visible, mostrarlo en la vista de Diseño. Observaremos que los controles de los formularios no están muy bien organizados.
3. Organizar los botones del formulario de modo que el botón **Display** quede posicionado a la izquierda del botón **Exit**.
4. Hacer clic en el botón **Exit**, mantener presionada la tecla CTRL y hacer clic en el botón **Display**. Ambos botones quedarán seleccionados.
5. En el menú **Formato**, hacer clic en **Alinear** y, a continuación clic en **Lados superiores**. El botón **Exit** se alinea con la parte superior del botón **Display**.
6. Mientras los dos botones están seleccionados, en el menú **Formato**, hacer clic en **Centrar en el formulario** y clic en **Horizontalmente**.
7. Seleccionar el cuadro de grupo **Choose Output**.
8. En el menú **Formato**, hacer clic en **Centrar en el formulario** y, a continuación, en **Horizontalmente**.
9. Mientras el cuadro de grupo está seleccionado, presione dos veces la tecla de **flecha hacia arriba** para mover hacia arriba el cuadro de grupo.
10. Seleccionar el botón de opción **Display Current Time**, mantenga presionada la tecla CTRL, y hacer clic en el botón de opción **Display Current Date**.
11. En el menú **Formato**, hacer clic en **Alinear** y, a continuación, en **Lados izquierdos**.

12. Mientras los dos botones de opción están seleccionados, en el menú **Formato**, hacer clic en **Espaciado vertical** y clic en **Aumentar**. Repetir este paso para incrementar el espacio entre ambos controles.
13. Con los botones de opción seleccionados, en el menú **Formato**, hacer clic en **Centrar en el formulario** y, a continuación, en **Horizontalmente**.
14. Mientras los botones de opción están seleccionados, en el menú **Formato**, hacer clic en **Centrar en el formulario** y, a continuación, en **Verticalmente**.

✎ Establecer el orden de tabulación

1. En el menú **Ver**, hacer clic en **Orden de tabulación**.
2. Cambiar el orden de tabulación haciendo clic en cada uno de los controles. Hacer clic en los controles en el siguiente orden: **Label**, botón **Display**, botón **Exit**, cuadro de grupo **Choose Output**, botón de opción **Display Current Date** y botón de opción **Display Current Time**. El orden de tabulación resultante será como el de la ilustración siguiente:



3. En el menú **Ver**, hacer clic en **Orden de tabulación**.

✎ Delimitar y acoplar los controles

1. Hacer clic en el control **Label**, y establecer la propiedad **Dock** como **Top**.
2. Hacer clic en el botón **Exit**, y establecer la propiedad **Anchor** como **Top, Right**.
3. Hacer clic en el control **Groupbox**, y establecer la propiedad **Anchor** como **Bottom, Left, Right**.
4. Hacer clic en el botón de opción **Display Current Date** y establecer la propiedad **Anchor** como **Top**.
5. Hacer clic en el botón **Display Current Time** y establecer la propiedad **Anchor** como **Bottom**.
6. Generar y ejecutar la aplicación.
7. Cuando aparezca el formulario, cambiar su tamaño para comprobar el comportamiento de los controles respecto a su posicionamiento.

Lección: Crear aplicaciones MDI

- Aplicaciones SDI frente a aplicaciones MDI
- Cómo crear aplicaciones MDI
- Cómo interactúan los formularios principal y secundario
- Práctica: crear una aplicación MDI

Introducción

Cuando creamos aplicaciones basadas en Windows, podemos utilizar diferentes estilos para el interfaz de usuario. Nuestra aplicación puede tener un interfaz de un solo documento (*single-document interface*, SDI) o un interfaz de múltiples documentos (*multiple-document interface*, MDI), o podemos crear un interfaz de estilo explorador. Encontraremos más información sobre los diferentes tipos de interfaces de aplicaciones en el kit de desarrollo de software (SDK) del .NET Framework.

Esta lección explica cómo crear y utilizar aplicaciones MDI.

Objetivos de la lección

En esta lección, aprenderemos a:

- Enumerar las diferencias entre aplicaciones SDI y aplicaciones MDI.
- Crear aplicaciones MDI.
- Explicar cómo interactúan los formularios principal y secundario.

Aplicaciones SDI frente a aplicaciones MDI



Introducción

Antes de crear una aplicación basada en Windows, debemos determinar el estilo de interfaz de usuario para la aplicación.

SDI frente a MDI

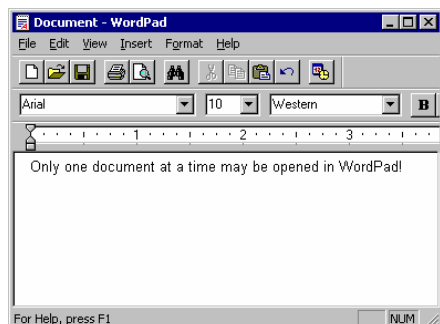
Como el nombre sugiere, las aplicaciones de interfaz de un solo documento (*single-document interface*, SDI) únicamente pueden soportar documentos de uno en uno, mientras que las aplicaciones de interfaz de múltiples documentos (*multiple-document interface*, MDI) pueden soportar varios documentos simultáneamente. La siguiente tabla muestra las diferencias entre las aplicaciones SDI y MDI y ofrece además ejemplos de qué estilo de interfaz debemos utilizar dependiendo del escenario.

SDI

Sólo un documento visible.

Se debe cerrar un documento antes de abrir otro.

Ejemplo: Microsoft WordPad

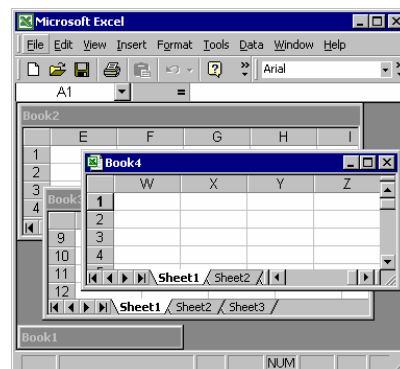


MDI

Varios documentos visibles al mismo tiempo.

Cada documento se muestra en su propia ventana.

Ejemplo: Microsoft Excel



Escenario: Una aplicación de calendario (no es necesario tener abierto más de una instancia de un calendario).

Escenario: Una aplicación de una aseguradora, en la que el usuario necesita trabajar con varios formularios de aplicación.

Cómo crear aplicaciones MDI

- Crear un formulario primario
 - Crear un nuevo proyecto
 - Configurar la propiedad **IsMdiContainer** como **True**
 - Añadir un elemento de menú para invocar el formulario secundario
- Crear un formulario secundario
 - Añadir un nuevo formulario al proyecto
- Invocar un formulario secundario desde uno primario

```
Protected Sub MenuItem2_OnClick(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles MenuItem2.Click
    Dim NewMdiChild As New Form2()
    'Set the Parent Form of the Child window.
    NewMdiChild.MdiParent = Me
    'Display the new form.
    NewMdiChild.Show()
End Sub
```

Introducción

La creación de una aplicación MDI implica tres pasos principales: crear un formulario primario, crear un formulario secundario e invocar el formulario secundario desde el formulario primario.

Procedimiento: crear aplicaciones MDI

Crear el formulario primario en el momento de diseño:

El formulario primario de una aplicación MDI es el formulario que contiene las ventanas MDI secundarias. Las ventanas secundarias se utilizan para que los usuarios interactúen con la aplicación MDI.

1. Crear un nuevo proyecto.
2. En la ventana Propiedades, establecer la propiedad **IsMdiContainer** en **True**.

De este modo, el formulario se designa como contenedor MDI para las ventanas secundarias.

Nota Cuando se establecen las propiedades de la ventana Propiedades, también puede establecerse la propiedad **WindowState** en **Maximized**, lo cual permite manipular fácilmente las ventanas MDI secundarias cuando el formulario primario está maximizado.

3. Desde el Cuadro de herramientas, arrastrar un componente **MainMenu** al formulario.

Necesitamos un menú para invocar los formularios secundarios desde el formulario primario. Como ejemplo, crear un elemento de menú de nivel superior con la propiedad **Text** establecida en **&Archivo** con elementos de submenú denominados **&Nuevo** y **&Cerrar**. Cree también un elemento de menú de nivel superior denominado **&Ventana**.

4. Establecer la propiedad **MdiList** del elemento de menú **Ventana** en **True**.

Los formularios MDI secundarios son esenciales para las aplicaciones MDI ya que los usuarios interactúan con la aplicación a través de ellos.

Para crear el formulario secundario en tiempo de diseño:

- En el mismo proyecto que contiene el formulario primario, crear un nuevo formulario.

Para invocar el formulario secundario desde el formulario primario:

1. Crear un controlador de eventos **Click** para el elemento de menú **Nuevo** del formulario primario.
2. Insertar código similar al siguiente para crear un nuevo formulario MDI secundario cuando el usuario haga clic en el elemento de menú **Nuevo**.

```
Protected Sub MenuItem2_OnClick(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MenuItem2.Click
    Dim NewMdiChild As New Form2()
    'Set the Parent Form of the Child window.
    NewMdiChild.MdiParent = Me
    'Display the new form.
    NewMdiChild.Show()
End Sub
```


Cómo interactúan los formularios primarios y secundarios

- Para ver una lista de las ventanas secundarias disponibles del formulario primario
 - Crear un elemento de menú (Windows) y configurar su propiedad **MdiList** como **True**
 - Para determinar el MDI secundario activo
 - Utilizar la propiedad **ActiveMdiChild**
- ```
Dim activeChild As Form = Me.ActiveMdiChild
```
- Para organizar ventanas secundarias en el formulario primario
    - Invocar el método **LayoutMdi**

\*\*\*\*\*

### Introducción

En una aplicación MDI, un formulario primario tiene varios formularios secundarios, y cada uno de los formularios secundarios interactúa con el formulario primario. Visual Studio .NET incluye varias propiedades que permiten que interactúen el formulario primario y los formularios secundarios de una aplicación MDI.

### Procedimiento: listar las ventanas secundarias de un formulario primario

Una forma fácil de mantener un seguimiento de las diferentes ventanas MDI secundarias abiertas por una aplicación es utilizar una lista **Window**. La funcionalidad de mantener un seguimiento de todos los formularios MDI secundarios abiertos además de determinar cual es el formulario secundario que tiene el foco es parte de Visual Studio .NET y se establece con la propiedad **MdiList** de un elemento de menú.

Para listar las ventanas secundarias de un formulario primario utilizando la lista *Window*:

1. Añadir un componente **MainMenu** al formulario primario.
2. Añadir los siguientes elementos de nivel superior al componente **MainMenu** utilizando el Diseñador de menús.

| Elemento de menú | Text    |
|------------------|---------|
| MenuItem1        | &File   |
| MenuItem2        | &Window |

3. Establecer la propiedad **MdiList** del elemento de menú **Ventana** como **True**.

### Procedimiento: determinar el formulario secundario activo

Cuando finalizamos determinados procedimientos de una aplicación, es importante determinar el formulario activo.

Debido a que una aplicación MDI puede tener numerosas instancias del mismo formulario secundario, el procedimiento debe saber qué formulario utilizar. Para especificar el formulario correcto, utilice la propiedad **ActiveMdiChild**, la

cual devuelve el formulario secundario que tiene el foco o el último que estaba activo.

Utilice el código siguiente para determinar el formulario secundario activo:

```
Dim activeChild As Form = Me.ActiveMDIChild
```

**Procedimiento:**  
organizar las ventanas secundarias del formulario primario

Para organizar las ventanas secundarias de un formulario primario, podemos utilizar el método **LayoutMdi** con la enumeración **MdiLayout** para reorganizar los formularios secundarios en un formulario MDI primario.

Existen cuatro valores de enumeración **MdiLayout** distintos que el método **LayoutMdi** puede utilizar. Estos valores ayudan a mostrar el formulario en cascada, en mosaico horizontal o en mosaico vertical, o como iconos de formulario secundario organizados a lo largo de la parte inferior del formulario MDI.

Para organizar formularios secundarios, en un evento, utilizar el método **LayoutMdi** para configurar la enumeración **MdiLayout** para el formulario MDI primario.


Podemos utilizar los siguientes miembros de la enumeración **MdiLayout** cuando invoquemos el método **LayoutMdi** de la clase **Form**:

| Miembro        | Descripción                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------|
| ArrangeIcons   | Todos los iconos MDI secundarios están organizados en la región cliente del formulario MDI primario.                   |
| Cascade        | Todas las ventanas MDI secundarias se muestran en cascada en la región cliente del formulario MDI primario.            |
| TileHorizontal | Todas las ventanas MDI secundarias se muestran en mosaico horizontal en la región cliente del formulario MDI primario. |
| TileVertical   | Todas las ventanas MDI secundarias se muestran en mosaico vertical en la región cliente del formulario MDI primario.   |

El siguiente ejemplo utiliza la configuración en cascada (*Cascade*) de la enumeración **MdiLayout** para las ventanas secundarias del formulario MDI primario (Form1).

```
Protected Sub CascadeWindows_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
 Me.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade)
End Sub
```

## Práctica: Creación de una aplicación MDI



En esta Práctica,

- Crearemos el formulario primario
- Crearemos el formulario secundario
- Mostraremos el formulario secundario desde el formulario primario

Empezar examinando los objetivos de esta actividad práctica

15 min

\*\*\*\*\*

### Introducción

En esta práctica, crearemos una aplicación MDI.

### Instrucciones

#### 🔗 Abrir el proyecto para la práctica

1. Utilizar Windows Explorer e ir a la carpeta pract03\Starter. Esta carpeta se puede encontrar dentro del fichero practs06.zip.
2. Hacer doble clic en el archivo de solución MdiApplication.sln para abrir el proyecto.

#### 🔗 Crear el formulario primario

1. Abrir ParentForm.vb en vista de Diseño.
2. Configurar la propiedad **IsMdiContainer** como **True**.

#### 🔗 Crear el menú File

1. Abrir el Cuadro de herramientas, añadir el control **MainMenu** al formulario, y configurar su propiedad **Name** como **MdiMenu**.
2. Hacer clic en el menú de la parte superior del formulario y configurar la propiedad **Text** del primer elemento de menú como **&File**.
3. Configurar la propiedad **Name** del menú **File** como **FileMenuItem** y la propiedad **MergeOrder** como **0**.
4. Abrir el menú **File**.
5. Hacer clic en el elemento de menú que aparece debajo de **File**, y configurar su propiedad **Text** como **&New**.
6. Configurar la propiedad **Name** del menú **New** como **NewMenuItem**.
7. Hacer clic en el elemento de menú que aparece debajo de **New**, y configurar su propiedad **Text** como **&Exit**.

8. Configurar la propiedad **Name** del menú **Exit** como **ExitMenuItem**.
9. Hacer doble clic en el elemento de menú **Exit** para crear el controlador de eventos **Click**.
10. En el controlador de eventos **exit**, añadir el siguiente código:

```
Me.Close()
```

#### ⚡ Crear el menú Window

1. Ir a la vista de Diseño.
2. Hacer clic en el segundo elemento de menú a la derecha de **File**, y establecer la propiedad **Text** como **&Window**.
3. Establecer la propiedad **Name** del menú **Window** como **WindowMenuItem** y la propiedad **MergeOrder** como **2**.
4. Establecer la propiedad **MdiList** del elemento de menú **Window** como **True**.
5. Abrir el menú **Window**.
6. Hacer clic en el elemento de menú que aparece debajo de **Window**, y establecer su propiedad **Text** como **&Cascade**.
7. Establecer la propiedad **Name** del menú **Cascade** como **WindowCascadeMenuItem**.
8. Hacer clic en el elemento de menú que aparece debajo de **Cascade**, y establecer su propiedad **Text** como **&Tile**.
9. Establecer la propiedad **Name** del menú **Tile** como **WindowTileMenuItem**.
10. Hacer doble clic en el menú **Cascade** y añadir el siguiente código al controlador de eventos **Click**:

```
Me.LayoutMdi (System.Windows.Forms.MdiLayout.Cascade)
```

11. Regresar a la vista de Diseño y hacer doble clic en el elemento de menú **Tile**.
12. Añadir el siguiente código al controlador de eventos **Click** del elemento de menú **Tile**:

```
Me.LayoutMdi _
(System.Windows.Forms.MdiLayout.TileHorizontal)
```

#### ⚡ Crear el formulario secundario

1. Abrir el menú **Proyecto** y hacer clic en **Agregar Windows Forms**.
2. Establecer el nombre del formulario como **ChildForm.vb**.
3. Establecer la propiedad **Text** del formulario como **Child Form**.
4. Desde el Cuadro de herramientas, arrastrar un control **RichTextBox** al formulario, y establecer su propiedad **Name** como **ChildTextBox**.
5. Establecer la propiedad **Dock** de **RichTextBox** como **Fill**.
6. Eliminar el valor existente de la propiedad **Text** de **RichTextBox** y dejarlo en blanco.
7. Desde el Cuadro de herramientas, arrastrar un control **MainMenu** al formulario.

8. Establecer la propiedad **Name** del control **MainMenu** como **ChildWindowMenu**.
9. Hacer clic en el menú de la parte superior del formulario y establecer el texto como **F&ormat**.
10. Establecer la propiedad **Name** del menú **Format** como **FormatMenuItem**, y establecer la propiedad **MergeOrder** como **1**.
11. Hacer clic en la entrada debajo del menú **Format** y establecer el texto como **&Toggle Foreground**.
12. Establecer la propiedad **Name** del menú **Toggle Foreground** como **ToggleMenuItem**.
13. Hacer doble clic en el menú **Toggle Foreground** y añadir el siguiente código al controlador de eventos **Click**:

```
If ToggleMenuItem.Checked Then
 ToggleMenuItem.Checked = False
 ChildTextBox.ForeColor = System.Drawing.Color.Black
Else
 ToggleMenuItem.Checked = True
 ChildTextBox.ForeColor = System.Drawing.Color.Blue
End If
```

#### ✚ **Mostrar el formulario secundario desde el formulario primario**

1. Abrir el formulario principal en la vista de Diseño.
2. Hacer doble clic en el elemento de menú **New** del menú **File** para crear el controlador de eventos **Click**.
3. Añadir el siguiente código al evento **Click** del elemento de menú **New**:

```
Dim newChild As New ChildForm()
newChild.MdiParent = Me
newChild.Show()
```

#### ✚ **Generar y ejecutar la aplicación**

1. Generar la aplicación y ejecutarla.
2. Cuando aparezca el formulario primario, en el menú **File**, hacer clic en **New**.  
  
Aparecerá una nueva ventana secundaria dentro de la ventana primaria. Observar cómo el menú de la ventana secundaria se fusiona con el menú de la ventana primaria y ordena el menú de acuerdo con las propiedades **MergeOrder** establecidas en el procedimiento **Display child form from parent form**.
3. Escribir un texto en el formulario secundario y utilizar el menú **Format** para cambiar el color del texto.
4. Abrir algunas ventanas secundarias adicionales.
5. Hacer clic en el menú **Window** y seleccionar **Tile**. Observar cómo las ventanas secundarias se reorganizan en modo mosaico.
6. Cerrar todas las ventanas secundarias. Observar que cuando se cierra la última ventana secundaria, el menú del formulario primario cambia eliminando el menú **Format**.
7. En el menú **File**, hacer clic en **Exit** para cerrar la aplicación.