Ferrando Eduardo – Ferrando Matias

WINDOWS FORMS – CONTROLES DE USUARIO

Apunte de Catedra

CONTENIDO

ENTORNO **3**
EXPLORADOR DE SOLUCIONES Y DISEÑADOR DE PROYECTOS 3
CREACIÓN DEL PROYECTO EN VISUAL STUDIO **4**
CUADRO DE PROPIEDADES 5
CUADRO DE EVENTOS 6
AÑADIR CONTROLES A LOS FORMULARIOS 6
PROGRAMANDO EVENTOS **8**
CONSIDERACIONES 9
CONTROLES DE USUARIO **10**
PROPIEDADES COMUNES EN LA MAYORÍA DE LOS CONTROLES 10
EVENTOS COMUNES EN LA MAYORÍA DE LOS CONTROLES 11
CONTROLES PRINCIPALES 11
Form *11*
Propiedades de formulario 12
Métodos de formulario 13
Eventos de formulario 14
TextBox *14*
Las propiedades del control TextBox 15
Los métodos del control TextBox 16
Eventos del Control de TextBox 16
MaskedTextBox *16*
Las propiedades del control MaskedTextBox 17
Eventos del Control de MaskedTextBox 17
NumericUpDown *18*
Las propiedades del control NumericUpDown 18
Eventos del Control de NumericUpDown 19
Label *19*
Propiedades del control de etiquetas 19
Métodos del control de etiquetas 20
Eventos del Control de Label 20
LinkLabel *20*
Propiedades del control de etiquetas de Links 21
Eventos del Control de LinkLabel 21
Ejemplo 21
Button *21*
Propiedades del control de botón 22
Métodos del Control de Botón 22
Eventos del Control Botón 23
ListBox *23*
Propiedades del control ListBox 24
Métodos del Control ListBox 25
Eventos del Control ListBox 25
ComboBox *25*
Propiedades del Control ComboBox 26
Métodos del Control ComboBox 27
Eventos del Control ComboBox 27
RadioButton *28*
Propiedades del control RadioButton 28
Métodos del control RadioButton 29
Eventos del control RadioButton 29
CheckBox *29*
Propiedades del control CheckBox 30
Eventos del control CheckBox 30
PictureBox *31*
Propiedades del control PictureBox 31

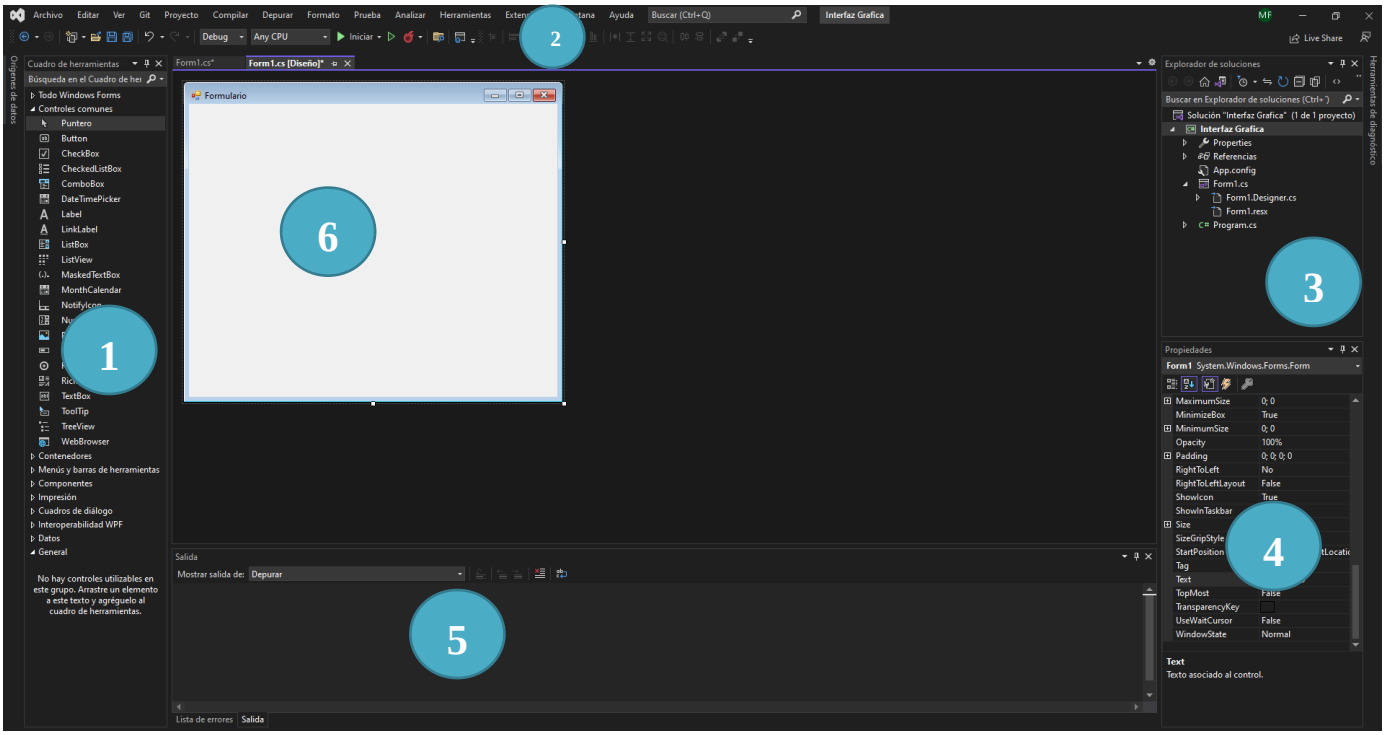
Métodos del PictureBox Control....32
Eventos del PictureBox Control....32
ProgressBar....*32*
Propiedades del control ProgressBar....33
Métodos del control ProgressBar....33
Eventos del control ProgressBar....34
DateTimePicker....*34*
Propiedades del control DateTimePicker....34
Métodos del control DateTimePicker....35
Eventos del control DateTimePicker....35
DataGridView....*36*
Propiedades del control DataGridView....37
Eventos del control DataGridView....39

CUADROS DE DIALOGO....**44**

ColorDialog....*44*
Propiedades del control ColorDialog....44
Métodos del control ColorDialog....45
Eventos del Control ColorDialog....45
FontDialog....*45*
Propiedades del Control FontDialog....45
Métodos del Control FontDialog....46
Eventos del control FontDialog....46
OpenFileDialog....*46*
Propiedades del control OpenFileDialog....47
Métodos del control OpenFileDialog....48
Eventos del control OpenFileDialog....48
SaveFileDialog....*49*
Propiedades del control SaveFileDialog....49
Métodos del control SaveFileDialog....50
Eventos del control SaveFileDialog....50
PrintDialog....*51*
Propiedades del control PrintDialog....52
Métodos del control PrintDialog....52
MessageBox....*52*
Aceptar....52
Aceptar – Cancelar....53
Si – No....54
Si – No – Cancelar....55
Reintentar – Cancelar....56
Anular – Reintentar – Omitir....57
Cerrar desde la barra de titulo....58

Entorno

El entorno C# está compuesto con herramientas para interactuar a través de ventanas, páginas de propiedades y asistentes.



1. Cuadro de herramientas
2. Barra de herramientas y menú
3. Proyectos abiertos, propiedades y ayuda
4. Cuadro de Propiedades y Eventos.
5. Compilación de la aplicación, listado de errores.
6. Área de diseño y edición

EXPLORADOR DE SOLUCIONES Y DISEÑADOR DE PROYECTOS

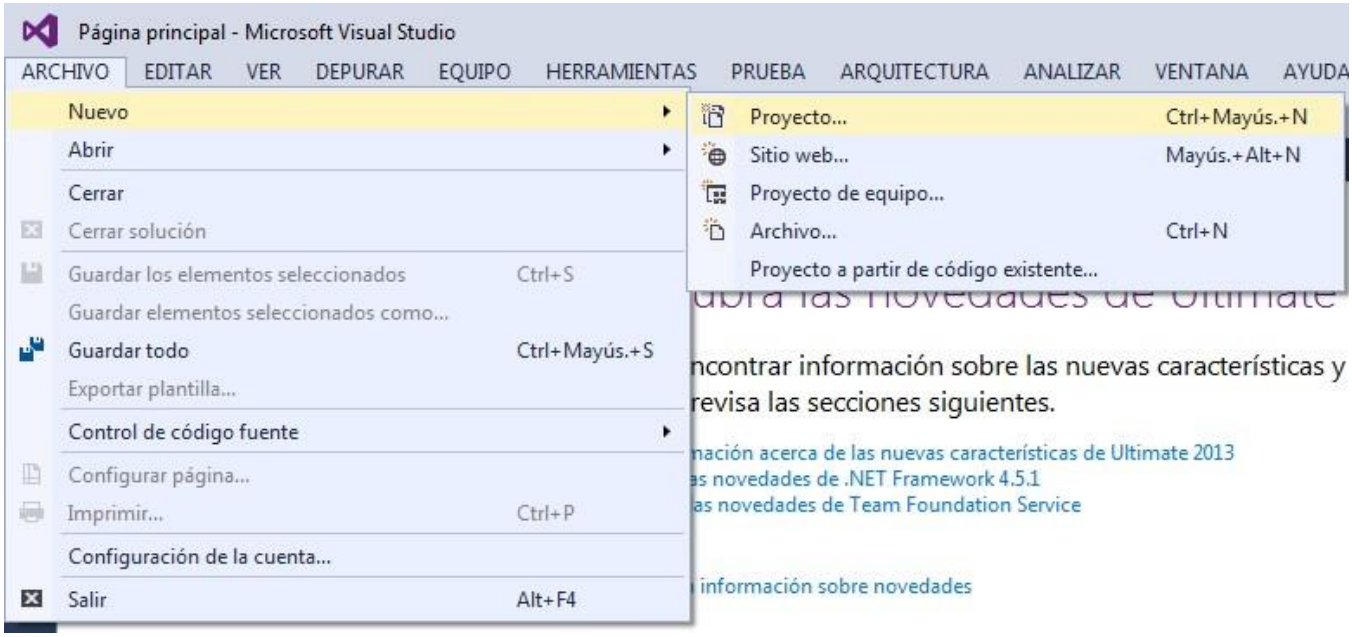
La ventana de la parte superior derecha es el “Explorador de Soluciones”, que muestra todos los archivos del proyecto en una vista de árbol jerárquica.

Cuando se utiliza el menú “Proyecto” para agregar nuevos archivos al proyecto, se verán reflejados en el Explorador de Soluciones. Además de los archivos, el Explorador de soluciones también muestra la configuración del proyecto y las referencias a las bibliotecas externas que necesita la aplicación.

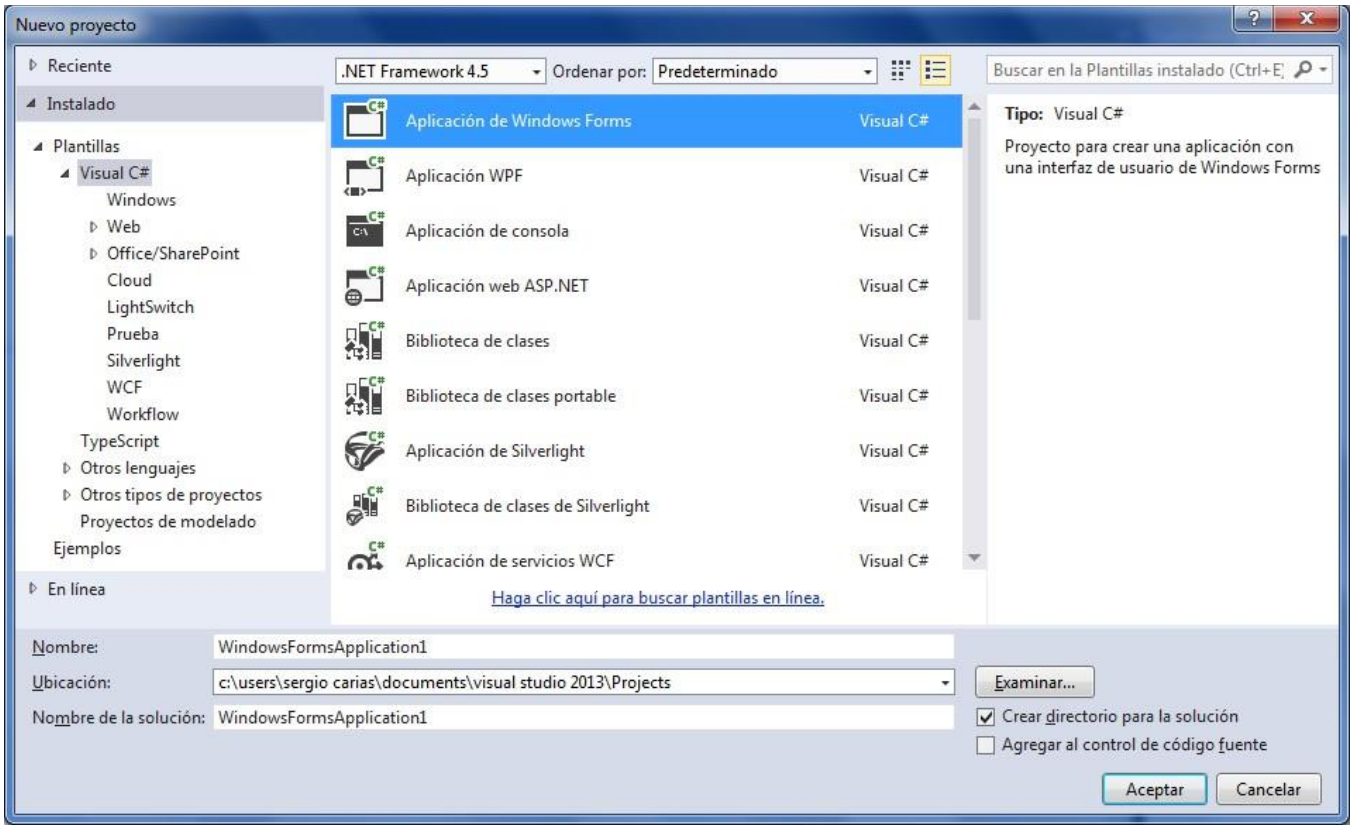
Para obtener acceso a las páginas de propiedades del Diseñador de proyectos, haga clic con el botón secundario del mouse en “Propiedades” del Explorador de soluciones y, a continuación, haga clic en “Abrir”.

Creación del proyecto en Visual Studio

- Inicie Visual Studio.NET
- En el menú "Archivo (FILE)" , seleccione "Nuevo (NEW)" y después seleccione la opción "Proyecto (PROJECT)"

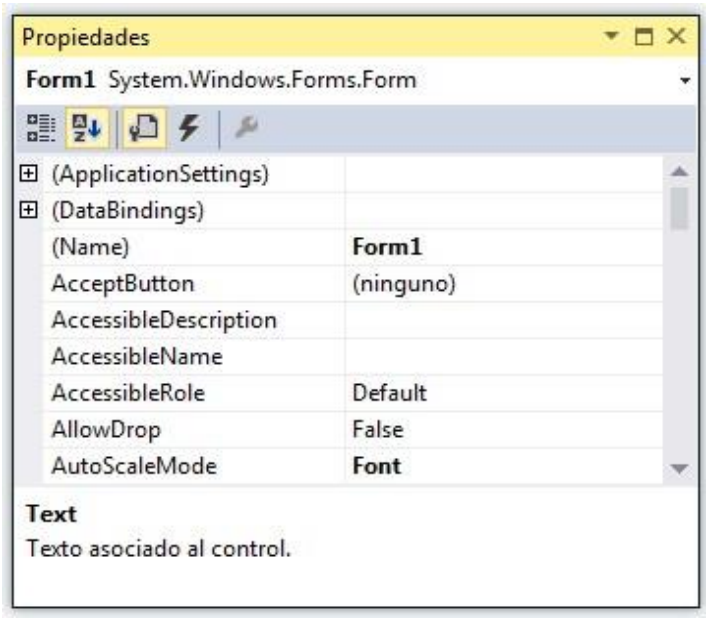


- En el panel "Tipos de Proyecto", seleccione proyectos de Visual C# y en el panel de plantillas, seleccione "Aplicación para Windows Forms (.NET Framework)" y en el cuadro de texto "Nombre del proyecto", escribiremos "Ejemplo1", después hacer clic en el botón "Aceptar (OK)"



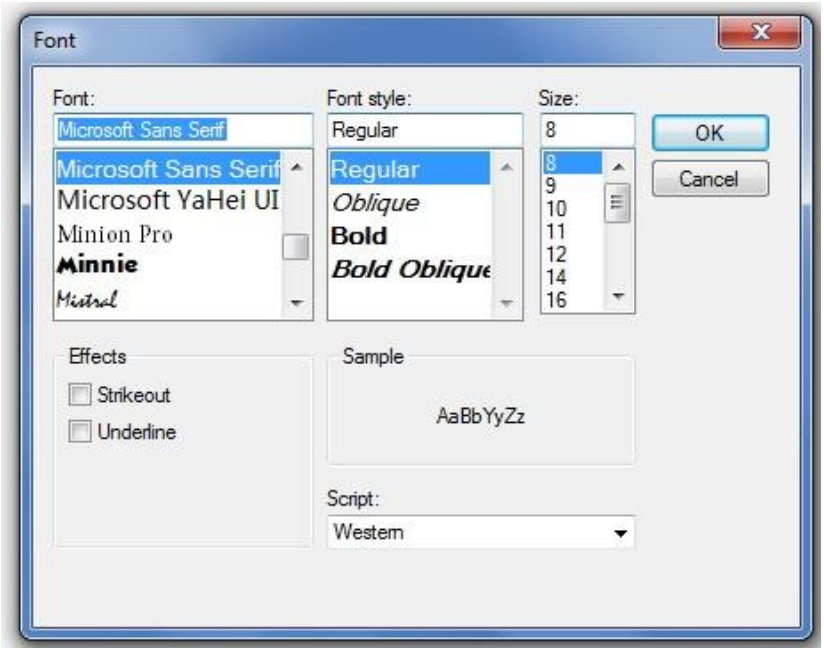
Cuadro de Propiedades

En la ventana “Propiedades” (si no ve la ventana presione (F4), haga clic en la propiedad (Name), y después escriba Ejemplo1 en el cuadro de texto (Name) para cambiar el nombre del control del formulario (esto se hace así, porque con este nombre se hará referencia cuando se esté programando)



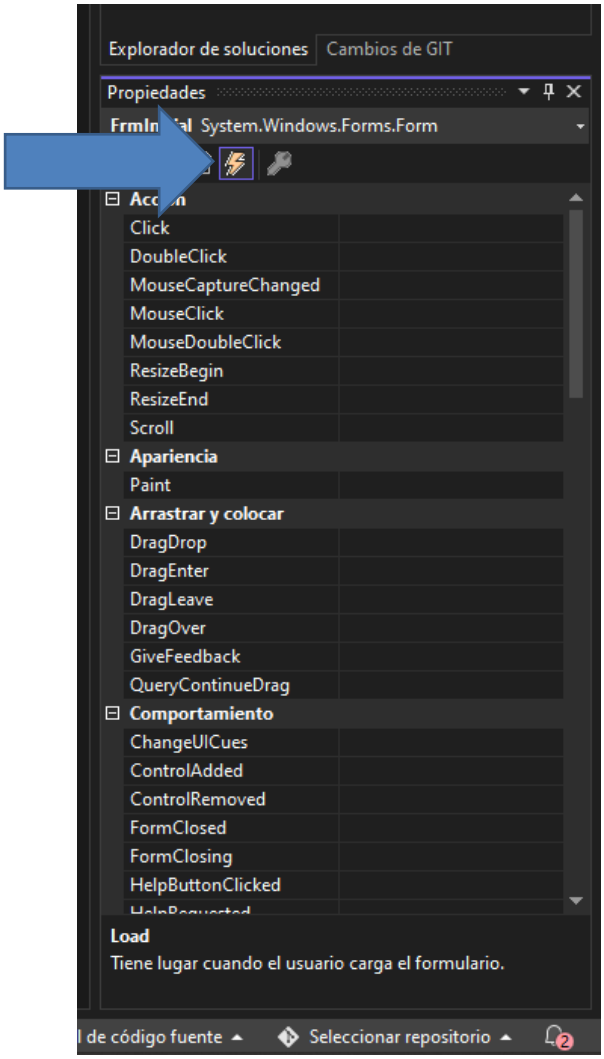
En la misma ventana “Propiedades”, seleccione la propiedad Text y después escriba “Este es un ejemplo”, para cambiar la barra de título del formulario.

Seleccione la propiedad Font y haga clic en el botón “Puntos suspensivos” que aparece al seleccionar la propiedad. Cuando se hace clic en el botón “Puntos suspensivos”, se abre el cuadro de diálogo “Font” y se puede seleccionar la fuente y los efectos que se desean.



Cuadro de Eventos

Dentro de la ventana de propiedades podemos, también, seleccionar diversos eventos haciendo click sobre el icono que muestra un rayo. Dando doble click al evento que deseamos programar nos creara y habilitara la sección de código a tal fin.

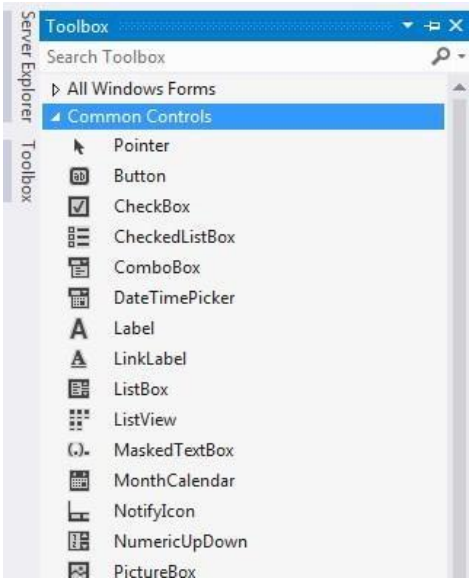


Añadir controles a los formularios

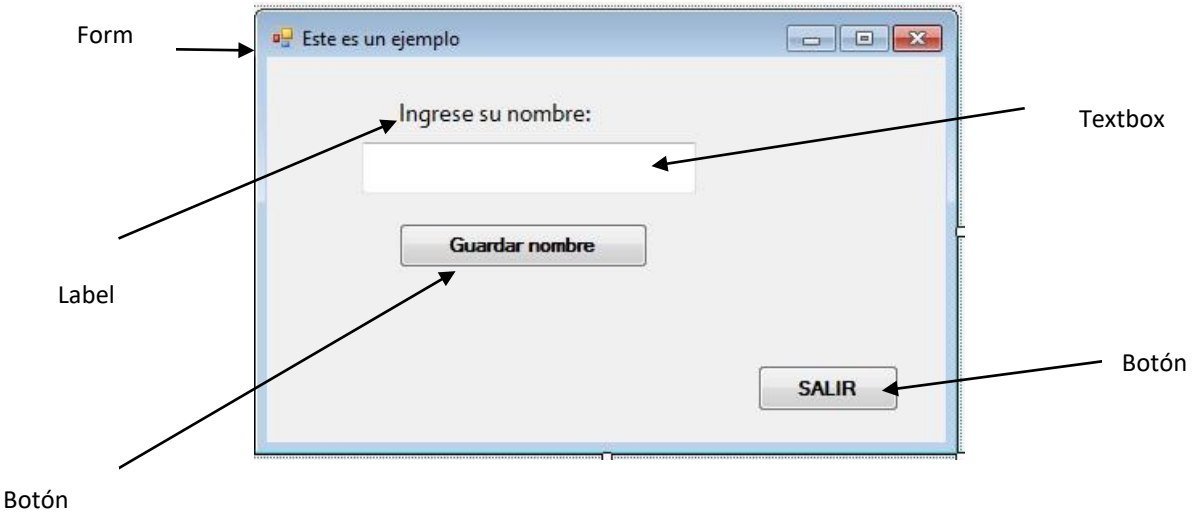
Hasta ahora se ha creado un formulario, se han establecido algunas de sus propiedades. Para que el formulario sea útil, se necesita agregar los controles y escribir algo de código propio.

Agregar controles de Windows Forms

En la siguiente figura se muestra el cuadro de herramientas, donde podemos arrastrar los controles que necesitamos para nuestra aplicación, (si no aparece la venta presionar Ctrl + w +x)

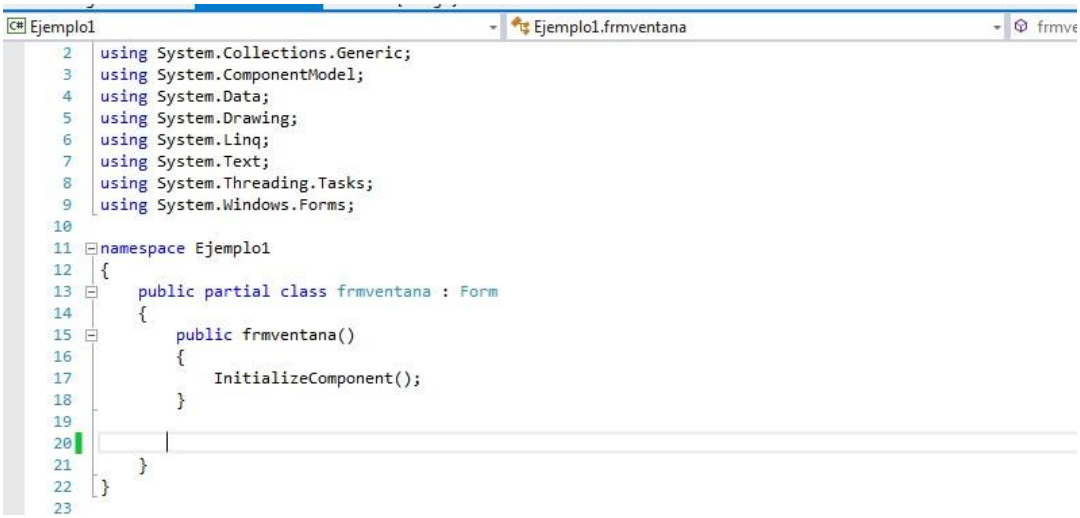


Arrastre los controles necesarios, para construir un formulario similar al mostrado en la figura siguiente:



Control	Propiedad	Valor
form1	Text	Este es un ejemplo
	Name	frmventana
label1	Text	Ingrese su nombre
	Name	lblmensaje
textbox1	Text	(dejar vacio)
	Name	txtnombre
button1	Text	Guardar nombre
	Name	btnguardar
button2	Text	SALIR
	Name	btnsalir

Ahora que tenemos toda la interfaz diseñada procedamos a la codificación:



Entremos al código dando **F7** o clic derecho sobre el form y eligiendo < >View Code. Para regresar al diseño cambiamos de pestaña y seleccionamos Form1.cs [Design]

Programando eventos

Hasta ahora se ha creado un formulario, se han establecido algunas de sus propiedades. Como siguiente paso codificaremos los eventos de cada herramienta, de forma que logremos obtener nuestro mensaje.

Damos doble clic sobre el btnguardar y nos mostrará algo como esto:

Al hacer doble click creamos un evento para esa herramienta y esto genera una función (o procedimiento) para decirle qué hacer

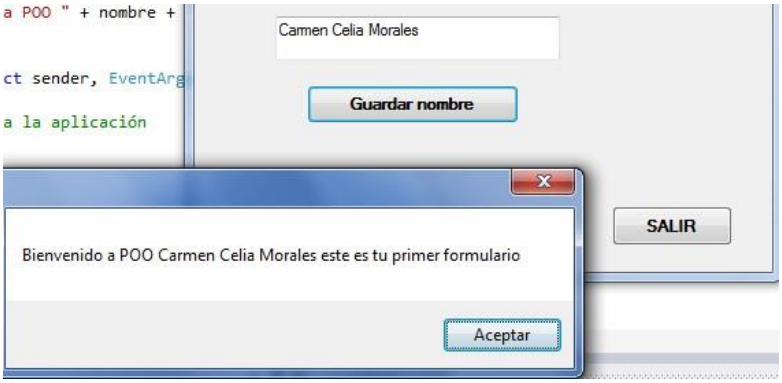
1.1 Ingresamos el siguiente fragmento de código en este método

```
string nombre = txtnombre.Text;
MessageBox.Show("Bienvenido a POO " + nombre + " este es tu primer formulario");
```

Como segundo paso hagamos la misma rutina, pero con el btnsalir y dentro de su método escribamos:

```
Application.Exit(); //termina la aplicación
```

Estamos listos para probar el programa, ejecútelo con Start o **F5** e ingrese su nombre en el textbox. Al hacer esto debe aparecernos esta ventana



Para concluir, de clic en Aceptar y luego clic en el botón salir que usted programó.

CONSIDERACIONES

- Los label y los messagebox, suelen ser utilizados como los encargados de mostrar mensajes para el usuario (hacen la función de un Console.WriteLine( ) pero en el entorno gráfico). Aunque no son los únicos sí son de los más frecuentes
- Hay varias formas de capturar información: textbox, radiobutton, listbox, comboBox. (Muchas formas de un Console.ReadLine( ))
- Todos los datos recogidos son considerados texto (string), así que si deseamos hacer cálculos con ellos (como una suma) hay que convertir eso a int, float, double o a lo que necesitemos.

Controles de Usuario

En Winforms todos los controles de usuario, incluidos los formularios, tienen una estructura común basada en la abstracción "Control".

Todos los controles tienen tres características principales:

- **Comportamiento:** Son funciones que tienen los controles. Poseen las funcionalidades de "Control" y las funcionalidades específicas de cada tipo de control.
- **Eventos:** Reacciones que tienen los controles ante distintos contextos. Por ejemplo:
  - Una interacción con el usuario por medio de un click.
  - Al momento de cargar la pantalla.
  - Al momento de cerrar la pantalla.
  - Al cambiar un valor.
  - Al cambiar un estado (por ejemplo, de "verdadero" a "falso").
  - Entre otros.
- **Propiedades:** Son atributos que permiten definir el estado de un control, puede ser, por ejemplo, el color, tamaño, un valor, el nombre, el texto que muestra en pantalla, entre otros.

Propiedades comunes en la mayoría de los controles

- **(Name):** Se utiliza para modificar el nombre del control con el fin de identificarlo correctamente. Como norma general se utilizan tres letras minúsculas para identificar el control seguido de su función comenzando con mayúscula, por ejemplo: btnSalir para un botón que salga de la aplicación, o txtNombre para una caja de texto donde el usuario deberá ingresar su nombre.
- **Text:** Modifica el texto que va a mostrar el control seleccionado.
- **TextAlign:** Modifica la alineación del texto del control.
- **TabIndex:** Seleccionamos el orden en el que van a ir tomando el foco los controles al presionar la tecla TAB. Se debe recordar que, como todos los índices, la primera posición será la 0 (cero).
- **Visible:** Indica si un control es visible (valor TRUE) o no es visible (valor FALSE) para el usuario.
- **BackColor:** Selecciona el color de fondo del control seleccionado.
- Font: Modifica el tipo de fuente, su estilo y tamaño del control seleccionado.
- **ForeColor:** Modifica el color de texto del control.
- **Cursor:** Modifica el tipo de cursor al colocarse sobre el control.
- **Enabled:** Permite que el control este (valor TRUE) o no (valor FALSE) habilitado para el usuario.
- **Location:** Permite modificar las coordenadas donde se verá el control dentro de su contenedor.
- **Size:** Modifica el tamaño del control.

- **BorderStyle:** Controla que tipo de borde se dibuja alrededor del control.

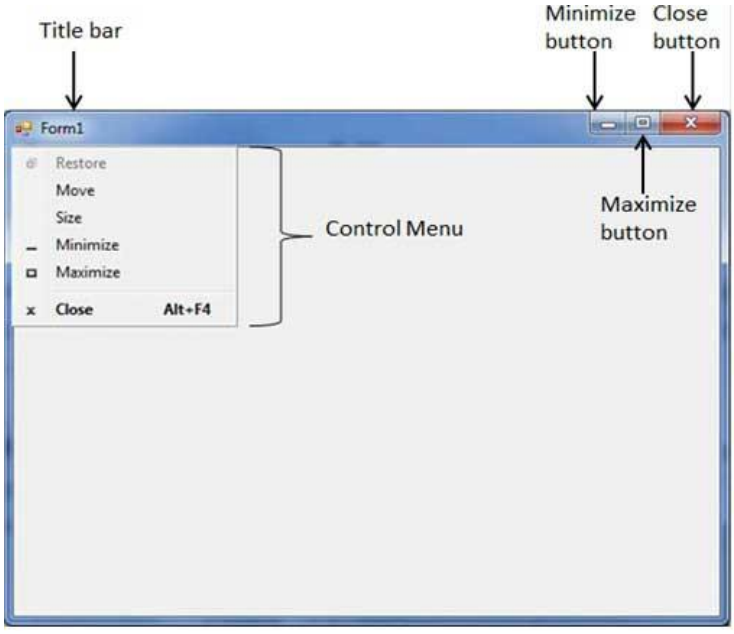
Eventos comunes en la mayoría de los controles

- **Click:** Sera lo que sucede cuando el usuario realice un click sobre el control
- **MouseHover:** Tiene lugar cuando el mouse permanece quieto dentro del control durante un tiempo.
- **MouseEnter:** Sera lo que sucede cuando el cursor entra en la parte visible del control.
- **MouseLeave:** Sera lo que sucede cuando el cursor sale de la parte visible del control.
- **MouseUp:** Ocurre cuando el puntero del mouse esta sobre el formulario y se suelta un botón del mouse.
- **TextChanged:** Evento que se desencadena cuando se cambia el valor de la propiedad Text del control.
- **KeyPress:** Ocurre cuando se presiona una tecla mientras el formulario tiene el foco.
- **KeyUp:** Ocurre cuando se suelta una tecla mientras el formulario tiene el foco.
- **KeyDown:** Ocurre cuando se presiona una tecla mientras el formulario tiene el foco.
- **DoubleClick:** Ocurre cuando se hace doble click en el control de formulario.
- **VisibleChanged:** Se produce cuando cambia el valor de la propiedad Visible.

Controles principales

Form

Form es el contenedor de todos los controles que componen la interfaz de usuario. Cada ventana que ve en una aplicación de Visual Studio en ejecución es un formulario, por lo que los términos formulario y ventana describen la misma entidad. Visual Studio crea un formulario predeterminado para usted cuando crea una aplicación de Windows Forms. Cada formulario tendrá una barra de título en la que se muestra el título del formulario y habrá botones para cerrar, maximizar y minimizar el formulario que se muestra a continuación:



Propiedades de formulario

La siguiente tabla enumera varias propiedades importantes relacionadas con un formulario. Estas propiedades se pueden establecer o leer durante la ejecución de la aplicación. Puede consultar la documentación de Microsoft para obtener una lista completa de las propiedades asociadas con un control de formulario:

- **AcceptButton:** El botón que se activa automáticamente cuando presiona Enter, sin importar qué control tenga el foco en ese momento. Por lo general, el botón Aceptar de un formulario se establece como BotónAceptar para un formulario.
- **CancelButton:** El botón que se activa automáticamente cuando presionas la tecla Esc. Por lo general, el botón Cancelar de un formulario se establece como CancelButton para un formulario.
- **AutoScroll:** Esta propiedad booleana indica si las barras de desplazamiento se adjuntarán automáticamente al formulario si se cambia el tamaño hasta el punto de que no todos sus controles son visibles.
- **AutoScrollMinSize:** Esta propiedad le permite especificar el tamaño mínimo del formulario, antes de que se adjunten las barras de desplazamiento.
- **BackColor:** Establece el color de fondo del formulario.
- **ControlBox:** De forma predeterminada, esta propiedad es True y puede establecerla en False para ocultar el icono y deshabilitar el menú Control.
- **Enabled:** Si es True, permite que el formulario responda a los eventos del mouse y del teclado; si es falso, deshabilita el formulario.
- **FormBorderStyle:** Indica la apariencia y el comportamiento del borde y de la barra de título del formulario.
- **HelpButton:** Determina si se debe mostrar un botón de Ayuda en el cuadro de título del formulario.
- **Icon:** Indica el icono para un formulario. Este icono se muestra en el cuadro de menú del sistema del formulario y cuando el formulario se minimiza.
- **MinimizeBox:** De forma predeterminada, esta propiedad es True y puede establecerla en False para ocultar el botón Minimizar en la barra de título.
- **MaximizeBox:** De forma predeterminada, esta propiedad es True y puede establecerla en False para ocultar el botón Maximizar en la barra de título.
- **MinimunSize:** Esto especifica la altura y el ancho mínimos de la ventana que puede minimizar.
- **MaximunSize:** Esto especifica la altura y el ancho máximos de la ventana que maximiza.
- **StartPosition:** Esta propiedad determina la posición inicial del formulario cuando se muestra por primera vez. Tendrá cualquiera de los siguientes valores:
  - **CenterParent**: el formulario se centra en el área de su formulario principal.
  - **CenterScreen**: el formulario se centra en el monitor.
  - **Manual**: la ubicación y el tamaño del formulario determinarán su posición inicial.
  - **WindowsDefaultBounds**: el formulario se coloca en la ubicación y el tamaño predeterminados determinados por Windows.

- **WindowsDefaultLocation**: el formulario se coloca en la ubicación predeterminada de Windows y tiene las dimensiones que estableció en el momento del diseño.

- **Text:** El texto, que aparecerá en la barra de título del formulario.
- **TopMost:** Esta propiedad es un valor Verdadero/Falso que le permite especificar si el formulario permanecerá encima de todos los demás formularios en su aplicación. Su propiedad predeterminada es Falsa.

Métodos de formulario

Los siguientes son algunos de los métodos comúnmente utilizados de la clase Form. Puede consultar la documentación de Microsoft para obtener una lista completa de los métodos asociados con el control de formularios:

- **Actívate:** Activa el formulario y le da foco.
- **ActivateMdiChild:** Activa el MDI secundario de un formulario.
- **AddOwnedForm:** Agrega un formulario propio a este formulario.
- **BringToFront:** Lleva el control al frente en el eje z.
- **CenterToParent:** Centra la posición del formulario dentro de los límites del formulario principal.
- **CenterToScreen:** Centra el formulario en la pantalla actual.
- **Close:** Cierra el formulario.
- **Contains:** Recupera un valor que indica si el control especificado es un elemento secundario del control.
- **Focus:** Establece el foco de entrada al control.
- **Hide:** Oculta el control del usuario.
- **Refresh:** Obliga al control a invalidar su área de cliente e inmediatamente se vuelve a dibujar a sí mismo y a cualquier control secundario
- **Scale(SizeF):** Escala el control y todos los controles secundarios según el factor de escala especificado.
- **ScaleControl:** Escala la ubicación, el tamaño, el relleno y el margen de un control.
- **Select:** Activa el mando.
- **SendToBack:** Envia el control al final del eje z.
- **SetAutoScrollMargin:** Establece el tamaño de los márgenes de desplazamiento automático.
- **SetDesktopBounds:** Establece los límites del formulario en coordenadas de escritorio.
- **SetDesktopLocation:** Establece la ubicación del formulario en coordenadas de escritorio.
- **SetDisplayRectLocation:** Posiciona la ventana de visualización en el valor especificado.
- **Show:** Muestra el control al usuario.
- **ShowDialog:** Muestra el formulario como un cuadro de dialogo modal.

Eventos de formulario

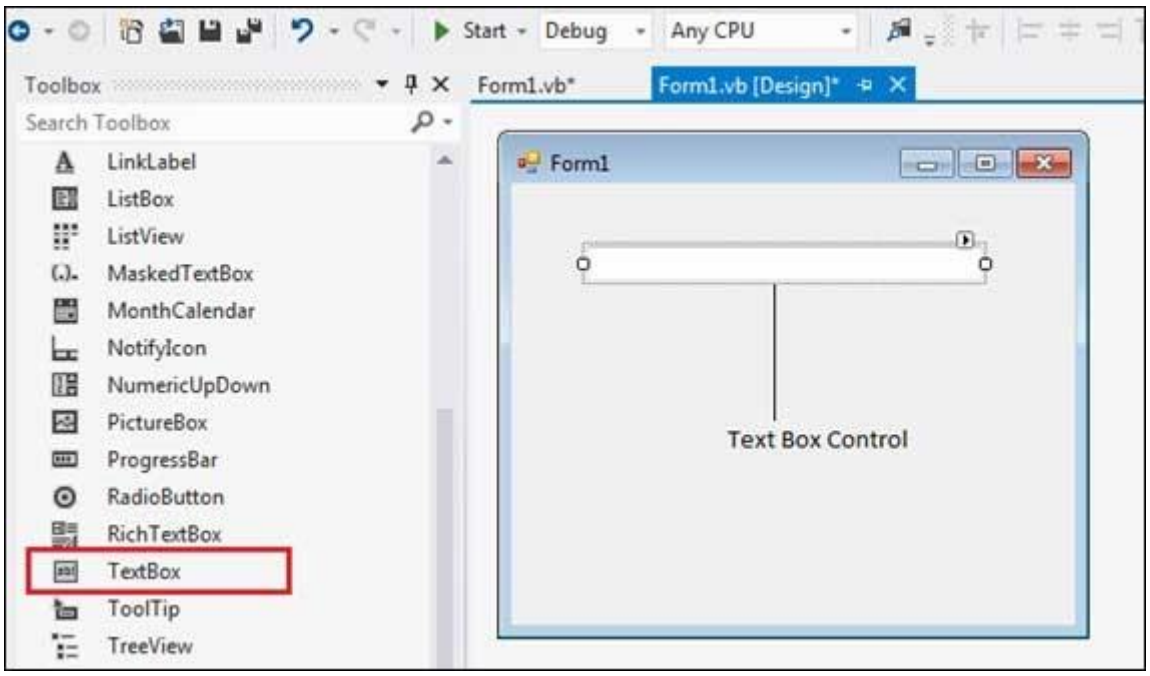
La siguiente tabla enumera varios eventos importantes relacionados con un formulario. Puede consultar la documentación de Microsoft para obtener una lista completa de eventos asociados con el control de formularios:

- **Activated:** Ocurre cuando el formulario es activado en código o por el usuario.
- **FormClosed:** Se produce antes de que se cierre el formulario.
- **FormClosing:** Ocurre cuando el formulario se está cerrando.
- **DragDrop:** Se produce cuando se completa una operación de arrastrar y soltar.
- **Enter:** Ocurre cuando se ingresa el formulario.
- **HelpButtonClicked:** Ocurre cuando se hace click en el botón "Ayuda".
- **Load (Evento por defecto):** Ocurre antes de que se muestre un formulario por primera vez.
- **Move:** Se produce cuando se mueve el formulario.
- **Resize:** Se produce cuando se cambia el tamaño del control.
- **Scroll:** Ocurre cuando el usuario o código se desplaza por el área del cliente.
- **Shown:** Ocurre cada vez que se muestra el formulario por primera vez.

TextBox

Los controles de cuadro de texto permiten ingresar texto en un formulario en tiempo de ejecución. De manera predeterminada, toma una sola línea de texto, sin embargo, puede hacer que acepte varios textos e incluso agregarle barras de desplazamiento.

Vamos a crear un cuadro de texto arrastrando un control de TextBox desde el cuadro de herramientas y soltándolo en el formulario.



Las propiedades del control TextBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control TextBox:

- **Name**
- **AcceptsReturn:** Obtiene o establece un valor que indica si al presionar ENTRAR en un control TextBox de varias líneas se crea una nueva línea de texto en el control o se activa el botón predeterminado del formulario.
- **AutoCompleteCustomSource:** Obtiene o establece una System.Collections.Specialized.StringCollection personalizada para usar cuando la propiedad AutoCompleteSource se establece en CustomSource.
- **AutoCompleteMode:** Obtiene o establece una opción que controla cómo funciona la finalización automática para TextBox.
- **AutoCompleteSource:** Obtiene o establece un valor que especifica el origen de las cadenas completas utilizadas para la finalización automática.
- **CharacterCasing:** Obtiene o establece si el control TextBox modifica las mayúsculas y minúsculas de los caracteres a medida que se escriben.
- **Lines:** Obtiene o establece las líneas de texto en un control de cuadro de texto.
- **MultiLine:** Obtiene o establece un valor que indica si se trata de un control TextBox de varias líneas.
- **PasswordChar:** Obtiene o establece el carácter que se usa para enmascarar los caracteres de una contraseña en un control TextBox de una sola línea.
- **ReadOnly:** Obtiene o establece un valor que indica si el texto del cuadro de texto es de solo lectura.
- **ScrollBars:** Obtiene o establece qué barras de desplazamiento deben aparecer en un control TextBox de varias líneas. Esta propiedad tiene valores:
  - Ninguna
  - Horizontal
  - Vertical
  - Ambas cosas
- **WordWrap:** Indica si un control de cuadro de texto de varias líneas ajusta automáticamente las palabras al principio de la siguiente línea cuando es necesario.
- **Font**
- **ForeColor**
- **TabIndex**
- **TextAlign**

Los métodos del control TextBox

Los siguientes son algunos de los métodos comúnmente utilizados del control TextBox:

- **AppendText:** Añade texto al texto actual de un cuadro de texto.
- **Clear:** Borra todo el texto del control de cuadro de texto.
- **Copy:** Copia la selección actual en el cuadro de texto al **portapapeles**.
- **Cut:** Mueve la selección actual en el cuadro de texto al **Portapapeles**.
- **Paste:** Reemplaza la selección actual en el cuadro de texto con el contenido del **Portapapeles**.
- **Paste(String):** Establece el texto seleccionado en el texto especificado sin borrar el búfer de deshacer.
- **ResetText:** Restablece la propiedad Texto a su valor predeterminado.
- **ToString:** Devuelve una cadena que representa el control TextBoxBase.
- **Undo:** Deshace la última operación de edición en el cuadro de texto.

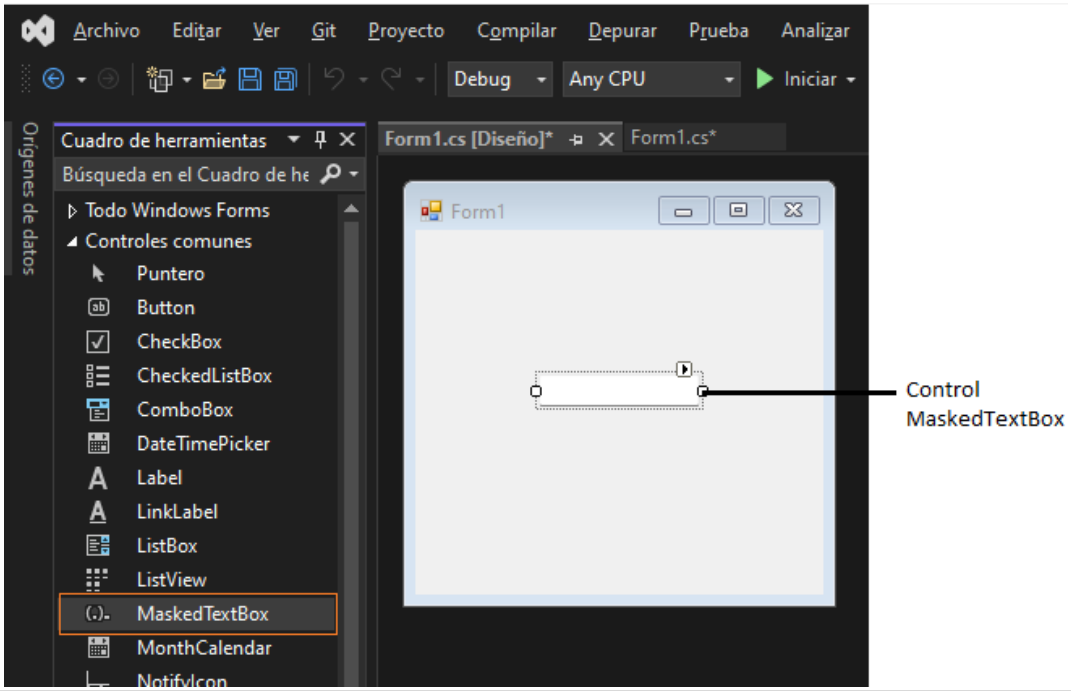
Eventos del Control de TextBox

Los siguientes son algunos de los eventos comúnmente utilizados del control TextBox:

- **Click**
- **KeyDown**
- **KeyPress**
- **KeyUp**
- **TextChanged (Evento por defecto)**

MaskedTextBox

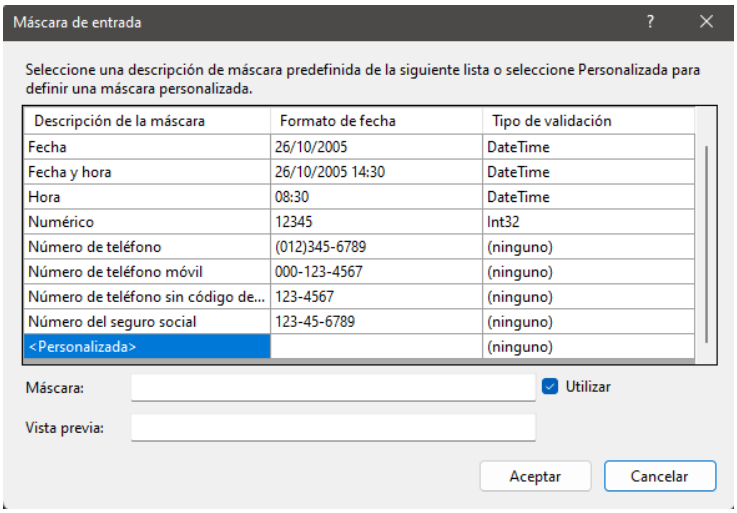
Un control MaskedTextBox proporciona un mecanismo de validación para la entrada del usuario en un formulario. Por ejemplo, si desea que un cuadro de texto acepte una fecha en formato mm/dd/aaaa, puede configurar el enmascaramiento en el cuadro de texto enmascarado.



Las propiedades del control MaskedTextBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control MaskedTextBox:

- **Name**
- **Text**
- **BorderStyle**
- **Enabled**
- **Font**
- **ForeColor**
- **TabIndex**
- **Mask:** Establece la cadena que controla la entrada permitida para este control.



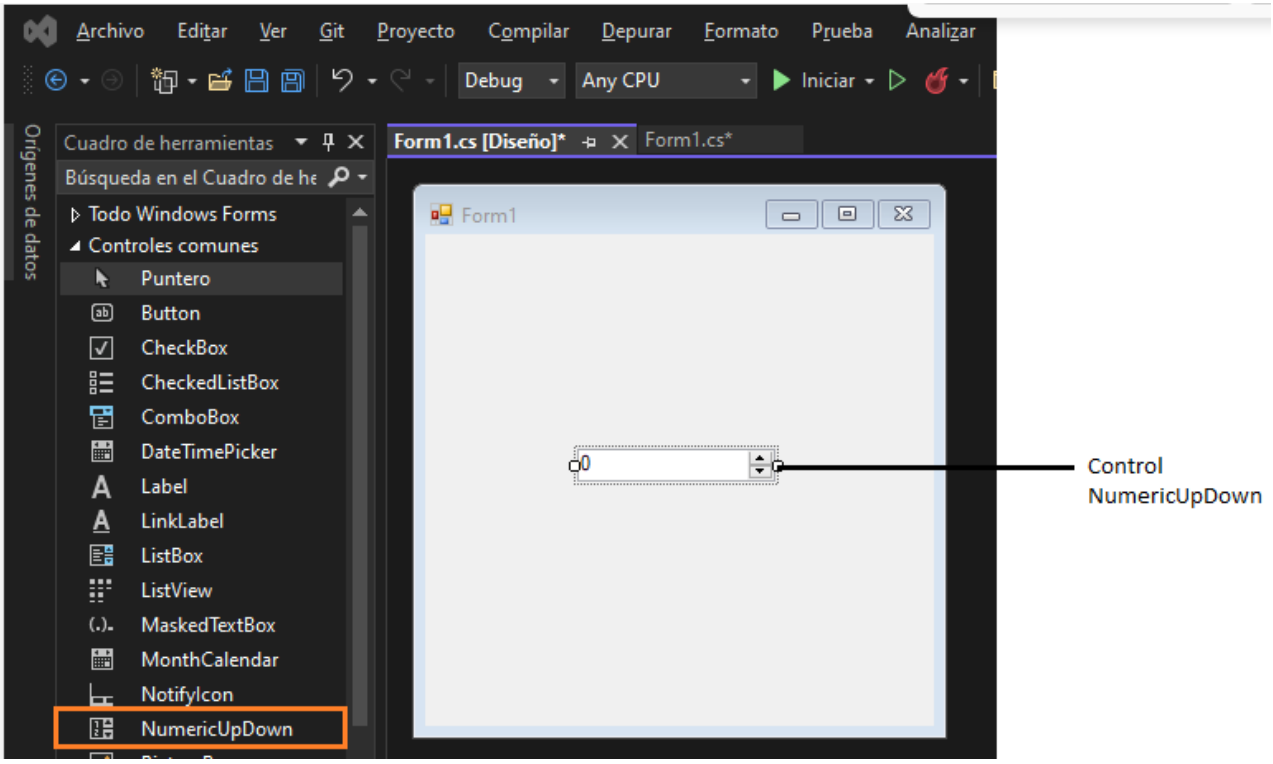
Eventos del Control de MaskedTextBox

Los siguientes son algunos de los eventos comúnmente utilizados del control TextBox:

- **Click**
- **KeyDown**
- **KeyPress**
- **KeyUp**
- **TextChanged**
- **MaskInputRejected (Evento por defecto):** Tiene lugar cuando el texto o el carácter de entrada no cumplen la especificación de la máscara.

NumericUpDown

Un control NumericUpDown permite a los usuarios proporcionar una interfaz giratoria (arriba/abajo) para moverse a través de números predefinidos usando flechas hacia arriba y hacia abajo.



Las propiedades del control NumericUpDown

Las siguientes son algunas de las propiedades comúnmente utilizadas del control NumericUpDown:

- **Name**
- **BackColor**
- **BorderStyle**
- **DecimalPlaces:** Indica el numero de posiciones decimales que se muestran.
- **Enabled**
- **Font**
- **ForeColor**
- **Increment:** Indica la cantidad que se va a aumentar o disminuir cada vez que hace click en el boton.
- **Minimun:** Indica el valor mínimo para el control numérico de flechas.
- **Maximun:** Indica el valor máximo para el control numérico de flechas.
- **TabIndex**
- **TextAlign**
- **UpDownAlign:** Indica la forma en que el control de flechas ubica los botones de flecha arriba y abajo en relación con su cuadro de edición.
- **Value:** Valor Actual del control numérico de flechas.
- **Visible**

Eventos del Control de NumericUpDown

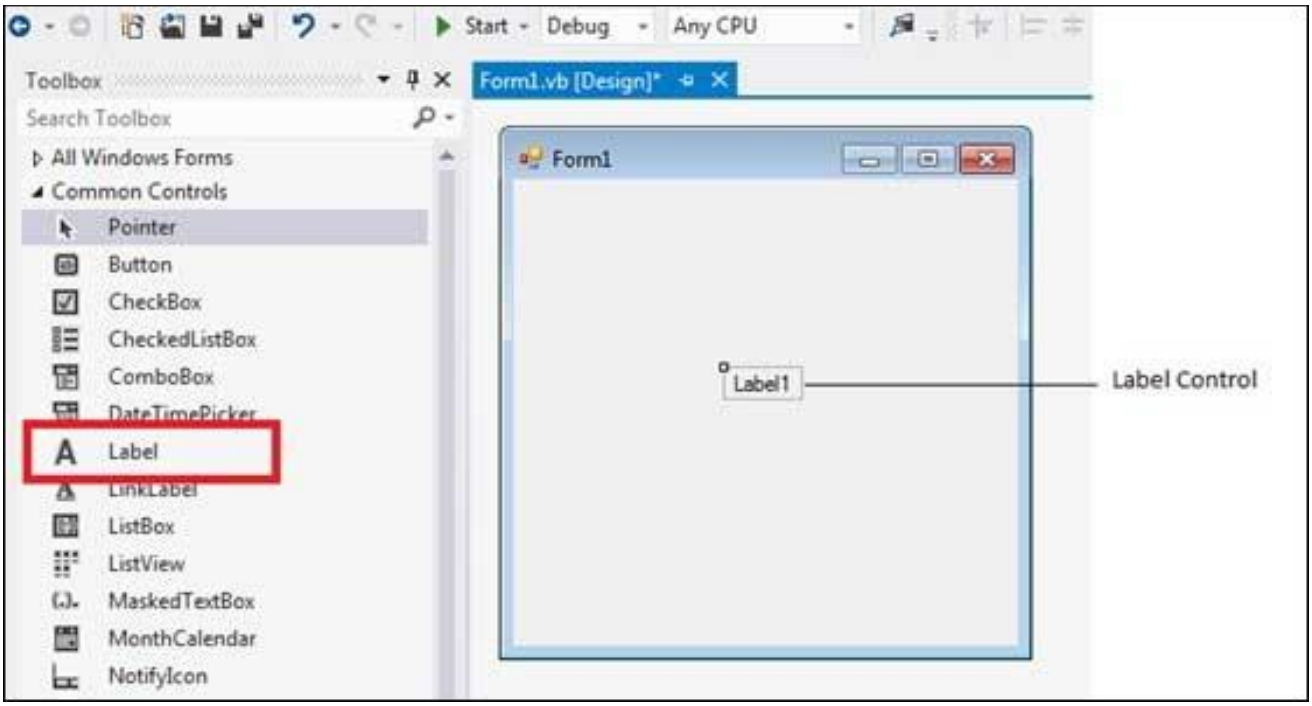
Los siguientes son algunos de los eventos comúnmente utilizados del control NumericUpDown:

- **Click**
- **KeyDown**
- **KeyPress**
- **KeyUp**
- **ValueChanged (Evento por defecto):** Tiene lugar cuando cambia el valor del control numérico de flechas.

Label

El control Etiqueta representa una etiqueta estándar de Windows. Generalmente se usa para mostrar algún texto informativo en la GUI que no se cambia durante el tiempo de ejecución.

Vamos a crear una etiqueta arrastrando un control Etiqueta desde la Caja de herramientas y soltándolo en el formulario.



Propiedades del control de etiquetas

Las siguientes son algunas de las propiedades comúnmente utilizadas del control Label:

- **AutoSize:** Obtiene o establece un valor que especifica si se debe cambiar el tamaño del control automáticamente para mostrar todo su contenido.
- **Font**
- **ForeColor**
- **Text**
- **TextAlign**

Métodos del control de etiquetas

Los siguientes son algunos de los métodos comúnmente utilizados del control Etiqueta:

- **GetPreferredSize:** Recupera el tamaño de un área rectangular en la que se puede colocar un control.
- **Refresh:** Obliga al control a invalidar su área de cliente e inmediatamente se vuelve a dibujar a sí mismo y a cualquier control secundario.
- **Select:** Activa el mando.
- **Show:** Muestra el control al usuario.
- **ToString:** Devuelve un String que contiene el nombre del control.

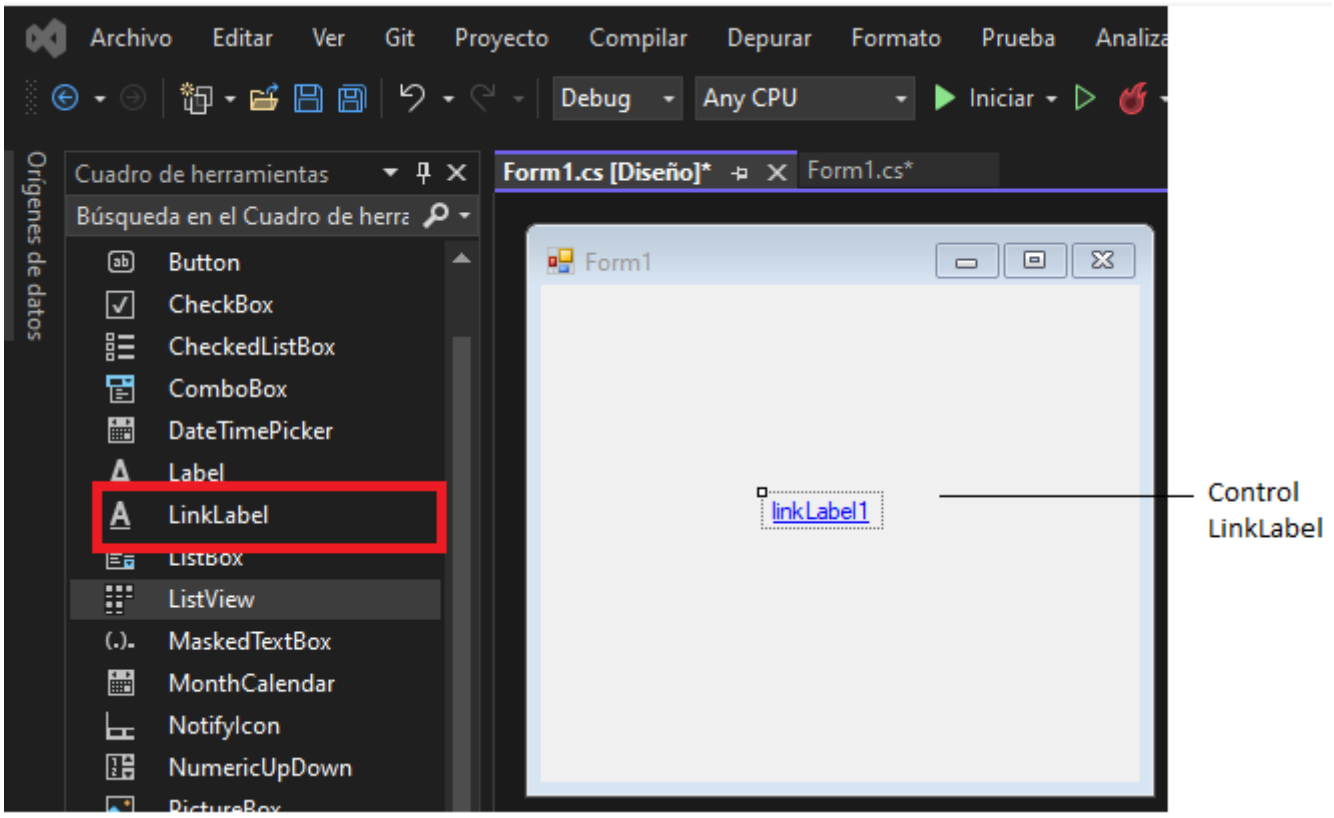
Eventos del Control de Label

Los siguientes son algunos de los eventos comúnmente utilizados del control Label:

- **Click (Evento por defecto)**
- **DoubleClick**
- **Leave**
- **TextChanged**

LinkLabel

Un control LinkLabel es un control de etiqueta que puede mostrar un hipervínculo. Un control LinkLabel se hereda de la clase Label, por lo que tiene toda la funcionalidad proporcionada por el control Label de Windows Forms. El control LinkLabel no participa en la entrada del usuario ni captura eventos del mouse o del teclado.



Propiedades del control de etiquetas de Links

Las siguientes son algunas de las propiedades comúnmente utilizadas del control LinkLabel:

- **Name**
- **BorderStyle**
- **Font**
- **ForeColor**
- **Text**
- **TextAlign**
- **TabIndex**
- **Visible**
- **Image:** Imagen que se mostrara en el control.
- **LinkArea:** Determina parte del texto de la etiqueta que se presenta como un hipervínculo.
- **LinkBehavior:** Determina el comportamiento del subrayado de un hipervínculo.
- **LinkColor:** Determina el color del hipervínculo en el estado predeterminado.
- **LinkVisited:** Determina si el hipervínculo debe presentarse como visitado.

Eventos del Control de LinkLabel

Los siguientes son algunos de los eventos comúnmente utilizados del control LinkLabel:

- **LinkClicked (Evento por defecto):** Tiene lugar cuando se hace click en el vínculo.

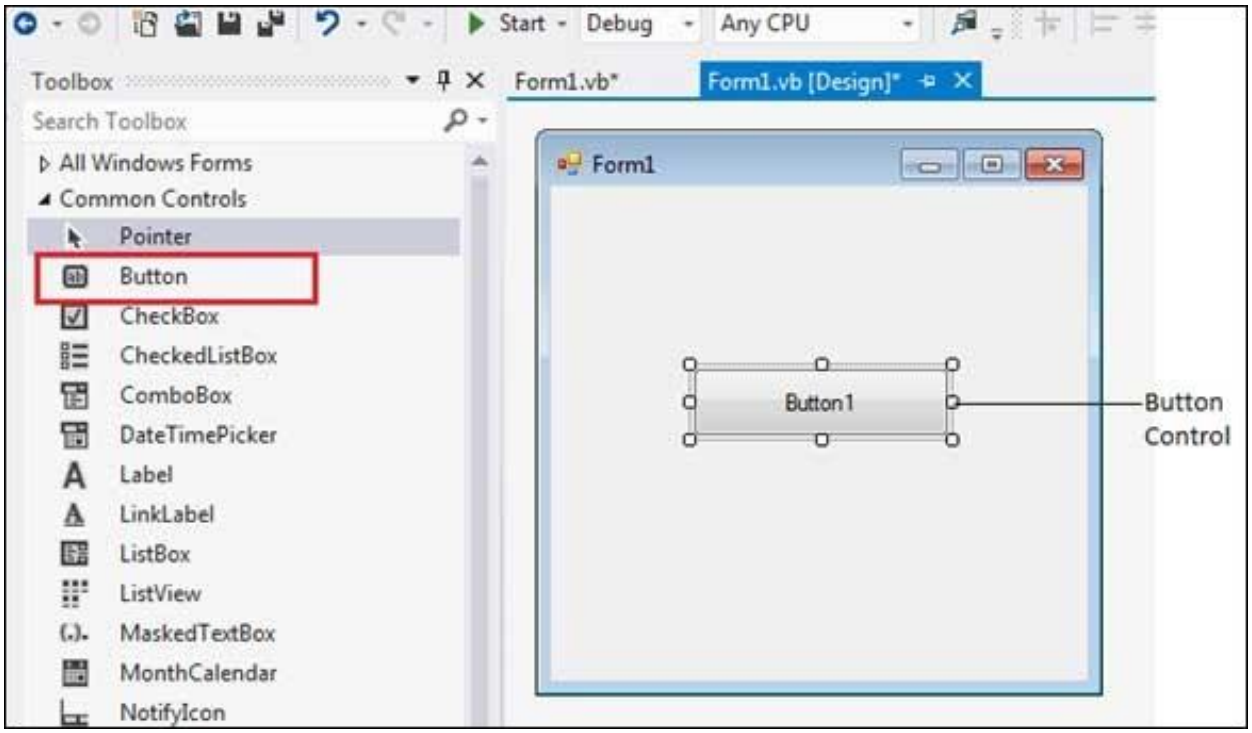
Ejemplo

```
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            linkLabel1.LinkVisited = true; // Cambio el color del link
            System.Diagnostics.Process.Start("https://www.youtube.com/"); // Abro la
dirección web en el navegador
        }
```

Button

El control de Windows Forms Button permite al usuario hacer clic en él para realizar una acción. El control Button puede mostrar tanto texto como imágenes. Cuando se hace clic en el botón, parece como si se estuviera presionando y soltando. Cada vez que el usuario hace clic en un botón, se invoca el controlador de eventos Click. Usted coloca el código en el controlador de eventos Click para realizar cualquier acción que elija. En los botones el evento por defecto será el Click.

Vamos a crear una etiqueta arrastrando un control button desde el cuadro de herramientas y soltándolo en el formulario.



Propiedades del control de botón

Las siguientes son algunas de las propiedades comúnmente utilizadas del control Botón:

- **Name**
- **BackGroundImage:** Obtiene o establece la imagen de fondo que se muestra en el control.
- **DialogResult:** Obtiene o establece un valor que se devuelve al formulario principal cuando se hace clic en el botón. Esto se utiliza al crear cuadros de diálogo.
- **Image:** Obtiene o establece la imagen que se muestra en un control de botón.
- **BackColor**
- **ForeColor**
- **Font**
- **AutoSizeMode**
- **Location**
- **TabIndex**
- **Text**

Métodos del Control de Botón

Los siguientes son algunos de los métodos comúnmente utilizados del control Botón:

- **GetPreferredSize:** Recupera el tamaño de un área rectangular en la que se puede colocar un control.
- **NotifyDefault:** Notifica al botón si es el botón predeterminado para que pueda ajustar su apariencia en consecuencia.
- **Select:** Activa el mando.
- **ToString:** Devuelve un String que contiene el nombre del control.

Eventos del Control Botón

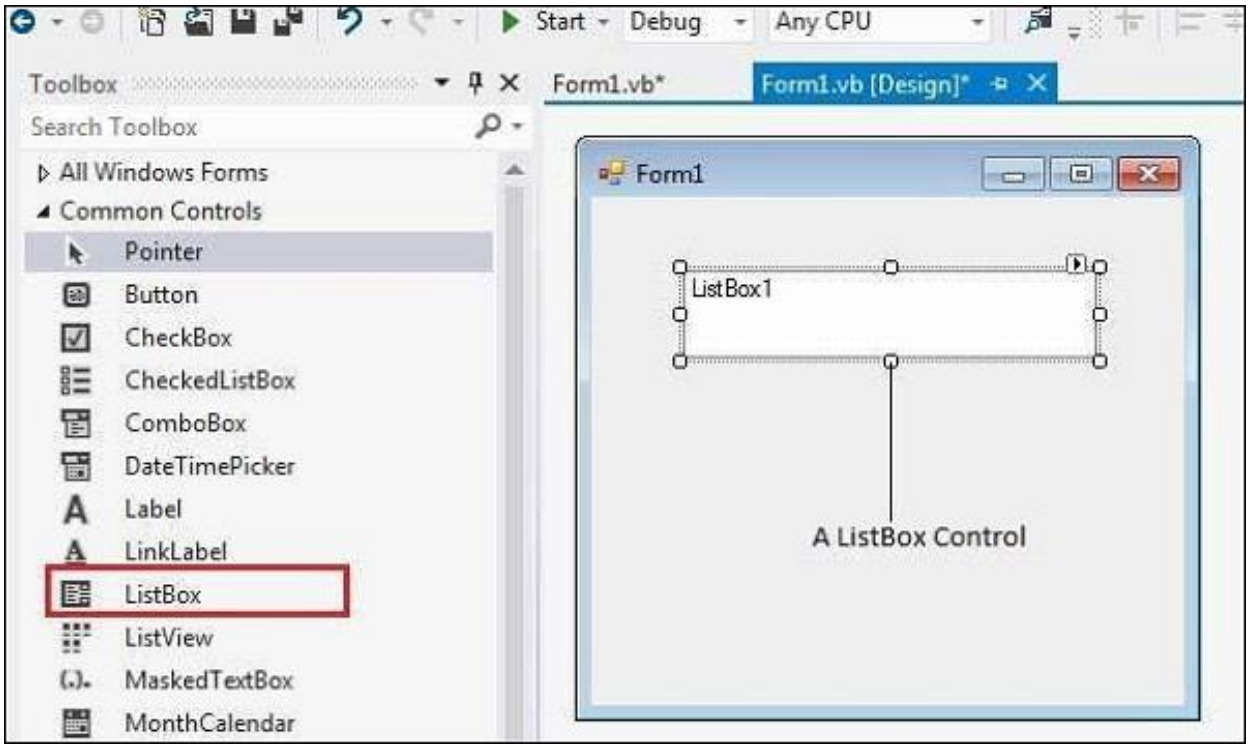
Los siguientes son algunos de los eventos de uso común del control Botón:

- **Click (Evento por defecto)**
- **DoubleClick**
- **Validated:** Ocurre cuando el control termina de validarse.

ListBox

ListBox representa un control de Windows para mostrar una lista de elementos a un usuario. Un usuario puede seleccionar un elemento de la lista. Le permite al programador agregar elementos en tiempo de diseño usando la ventana de propiedades o en tiempo de ejecución.

Vamos a crear un cuadro de lista arrastrando un control ListBox desde el cuadro de herramientas y soltándolo en el formulario.



Puede completar los elementos del cuadro de lista desde la ventana de propiedades o en tiempo de ejecución. Para agregar elementos a un ListBox, seleccione el control ListBox y acceda a la ventana de propiedades, para ver las propiedades de este control. Haga clic en el botón de puntos suspensivos (...) junto a la propiedad Elementos. Esto abre el cuadro de diálogo Editor de colección de cadenas, donde puede ingresar los valores uno por uno.

Propiedades del control ListBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control ListBox:

- **Name**
- **BorderStyle**
- **ColumnWidth:** Obtiene o establece el ancho de las columnas en un cuadro de lista de varias columnas.
- **Enabled**
- **Font**
- **ForeColor**
- **HorizontalExtent:** Obtiene o establece el área de desplazamiento horizontal de un cuadro de lista.
- **HorizontalScrollBar:** Obtiene o establece el valor que indica si se muestra una barra de desplazamiento horizontal en el cuadro de lista.
- **ItemHeight:** Obtiene o establece el alto de un elemento en el cuadro de lista.
- **Items:** Obtiene los elementos del cuadro de lista.
- **MultiColumn:** Obtiene o establece un valor que indica si el cuadro de lista admite varias columnas.
- **ScrollAlwaysVisible:** Obtiene o establece un valor que indica si la barra de desplazamiento vertical se muestra en todo momento.
- **SelectedIndex:** Obtiene o establece el índice de base cero del elemento seleccionado actualmente en un cuadro de lista.
- **SelectedItems:** Obtiene o establece el elemento seleccionado actualmente en el cuadro de lista.
- **SelectedValue:** Obtiene una colección que contiene los elementos seleccionados actualmente en el cuadro de lista.
- **SelectionMode:** Obtiene o establece el método en el que se seleccionan los elementos en el cuadro de lista. Esta propiedad tiene valores:
  - Ninguna
  - Una
  - MultiSimple
  - MultiExtendido
- **Sorted:** Obtiene o establece un valor que indica si los elementos del cuadro de lista están ordenados alfabéticamente.
- **TabIndex**
- **Visible**

Métodos del Control ListBox

Los siguientes son algunos de los métodos comúnmente utilizados del control ListBox:

- **BeginUpdate:** Evita que el control se dibuje hasta que se llame al método EndUpdate, mientras que los elementos se agregan al ListBox uno a la vez.
- **ClearSelected:** Anula la selección de todos los elementos del ListBox.
- **EndUpdate:** Reanuda el dibujo de un cuadro de lista después de que el método BeginUpdate lo haya desactivado.
- **FindString:** Encuentra el primer elemento en ListBox que comienza con la cadena especificada como argumento.
- **FindStringExact:** Encuentra el primer elemento en ListBox que coincida exactamente con la cadena especificada.
- **GetSelected:** Devuelve un valor que indica si el elemento especificado está seleccionado.
- **SetSelected:** Selecciona o borra la selección del elemento especificado en un ListBox.

Eventos del Control ListBox

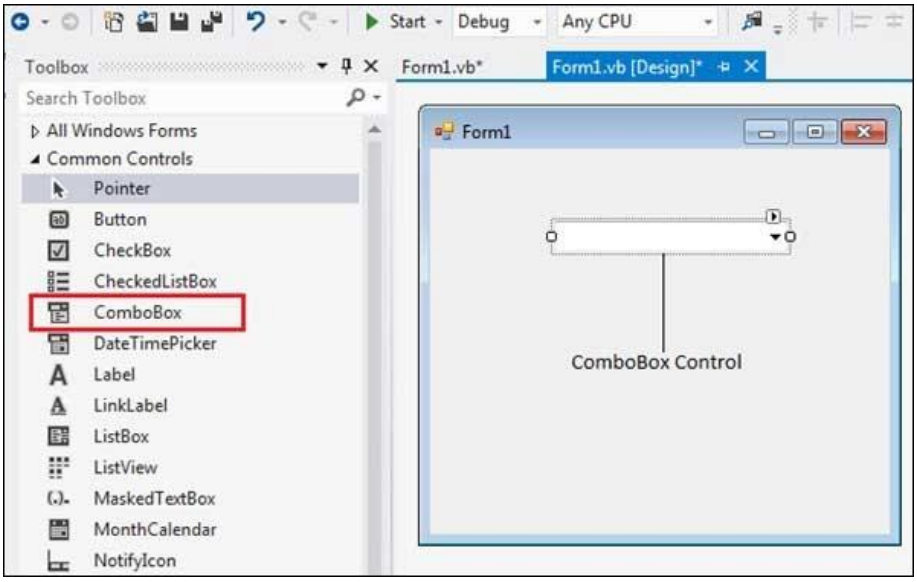
Los siguientes son algunos de los eventos de uso común del control ListBox:

- **Click**
- **SelectedIndexChanged:** Se produce cuando se cambia la propiedad SelectedIndex de un cuadro de lista. (Evento por defecto)

ComboBox

El control ComboBox se utiliza para mostrar una lista desplegable de varios elementos. Es una combinación de un cuadro de texto en el que el usuario ingresa un elemento y una lista desplegable desde la cual el usuario selecciona un elemento.

Vamos a crear un cuadro combinado arrastrando un control ComboBox desde el cuadro de herramientas y soltándolo en el formulario.



Puede completar los elementos del cuadro de lista desde la ventana de propiedades o en tiempo de ejecución. Para agregar elementos a un ComboBox, seleccione el control ComboBox y vaya a la ventana de propiedades para ver las propiedades de este control. Haga clic en el botón de puntos suspensivos (...) junto a la propiedad Elementos. Esto abre el cuadro de diálogo Editor de colección de cadenas, donde puede ingresar los valores uno por uno.

Propiedades del Control ComboBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control ComboBox:

- **Name**
- **AutoCompleteCustomSource:** Obtiene o establece una System.Collections.Specialized.StringCollection personalizada para usar cuando la propiedad AutoCompleteSource se establece en CustomSource.
- **AutoCompleteMode:** Obtiene o establece una opción que controla cómo funciona la finalización automática para ComboBox.
- **AutoCompleteSource:** Obtiene o establece un valor que especifica el origen de las cadenas completas utilizadas para la finalización automática.
- **DataSource:** Obtiene o establece el origen de datos para este ComboBox.
- **DropDownHeight:** Obtiene o establece el alto en píxeles de la parte desplegable de ComboBox.
- **DropDownStyle:** Obtiene o establece un valor que especifica el estilo del cuadro combinado.
- **DropDownWidth:** Obtiene o establece el ancho de la parte desplegable de un cuadro combinado.
- **FlatStyle:** Obtiene o establece la apariencia de ComboBox.
- **ItemHeight:** Obtiene o establece el alto de un elemento en el cuadro combinado.
- **Items:** Obtiene un objeto que representa la colección de elementos contenidos en este ComboBox.
- **MaxDropDownItems:** Obtiene o establece el número máximo de elementos que se mostrarán en la parte desplegable del cuadro combinado.
- **MaxLength:** Obtiene o establece el número máximo de caracteres que un usuario puede ingresar en el área editable del cuadro combinado.
- **SelectedIndex:** Obtiene o establece el índice que especifica el elemento seleccionado actualmente.
- **SelectedItem:** Obtiene o establece el elemento actualmente seleccionado en ComboBox.
- **SelectedText:** Obtiene o establece el texto que se selecciona en la parte editable de un ComboBox.
- **SelectedValue:** Obtiene o establece el valor de la propiedad de miembro especificada por la propiedad ValueMember.
- **SelectionLength:** Obtiene o establece el número de caracteres seleccionados en la parte editable del cuadro combinado.
- **SelectionStart:** Obtiene o establece el índice inicial del texto seleccionado en el cuadro combinado.
- **Sorted:** Obtiene o establece un valor que indica si los elementos del cuadro combinado están ordenados.
- **TabIndex**

- **Text**

Métodos del Control ComboBox

Los siguientes son algunos de los métodos comúnmente utilizados del control ComboBox:

- **BeginUpdate:** Impide que el control se dibuje hasta que se llame al método EndUpdate, mientras que los elementos se agregan al cuadro combinado de uno en uno.
- **EndUpdate:** Reanuda el dibujo de un cuadro combinado, después de que el método BeginUpdate lo haya desactivado.
- **FindString:** Busca el primer elemento en el cuadro combinado que comienza con la cadena especificada como argumento.
- **FindStringExact:** Encuentra el primer elemento en el cuadro combinado que coincide exactamente con la cadena especificada.
- **SelectAll:** Selecciona todo el texto en el área editable del cuadro combinado.

Eventos del Control ComboBox

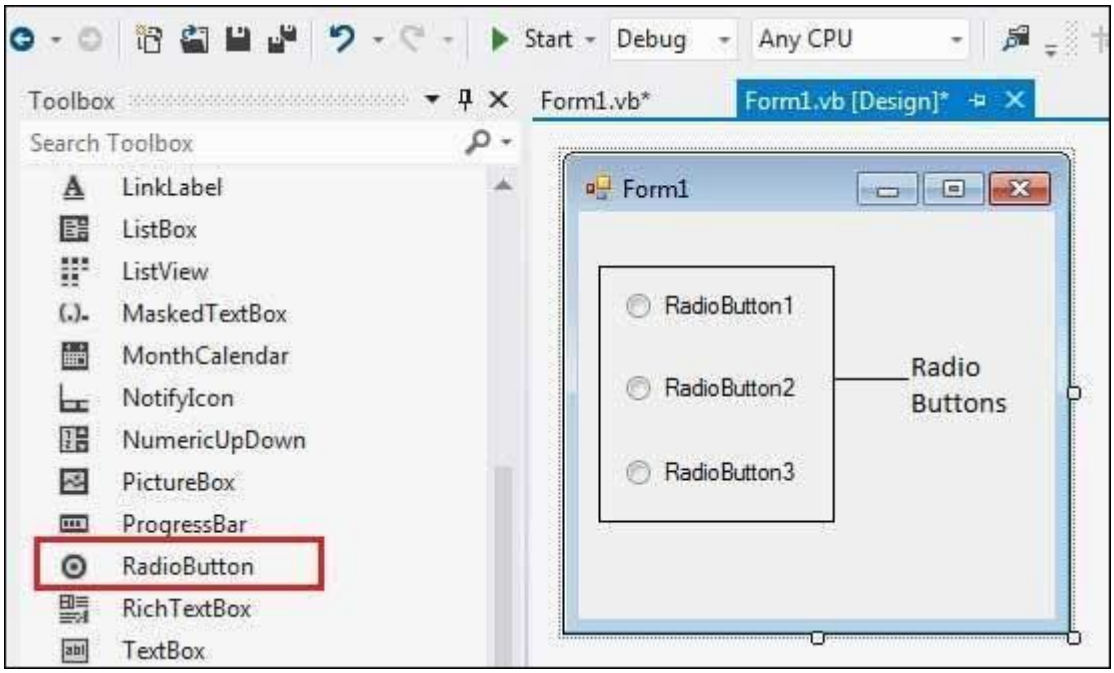
Los siguientes son algunos de los eventos de uso común del control ComboBox:

- **DropDown:** Se produce cuando se muestra la parte desplegable de un cuadro combinado.
- **DropDownClosed:** Se produce cuando la parte desplegable de un cuadro combinado ya no está visible.
- **DropDownStyleChanged:** Ocurre cuando la propiedad DropDownStyle de ComboBox ha cambiado.
- **SelectedIndexChanged:** Se produce cuando cambia la propiedad SelectedIndex de un control ComboBox. **(Evento por defecto)**
- **SlectionChangeCommitted:** Ocurre cuando el elemento seleccionado ha cambiado y el cambio aparece en el cuadro combinado.

RadioButton

El control RadioButton se utiliza para proporcionar un conjunto de opciones mutuamente excluyentes. El usuario puede seleccionar un botón de radio en un grupo. Si necesita colocar más de un grupo de botones de radio en el mismo formulario, debe colocarlos en diferentes controles de contenedor como un control GroupBox.

Vamos a crear tres botones de radio arrastrando los controles RadioButton desde la caja de herramientas y colocándolos en el formulario.



La propiedad *Checked* del botón de radio se usa para establecer el estado de un botón de radio. Puede mostrar texto, imagen o ambos en el control del botón de opción. También puede cambiar la apariencia del control de botón de opción mediante la propiedad *Apariencia*.

Propiedades del control RadioButton

Las siguientes son algunas de las propiedades comúnmente utilizadas del control RadioButton:

- **Name**
- **Appearance:** Obtiene o establece un valor que determina la apariencia del botón de opción.
- **AutoCheck:** Obtiene o establece un valor que indica si el valor Checked y la apariencia del control cambian automáticamente cuando se hace clic en el control.
- **CheckAlign:** Obtiene o establece la ubicación de la parte de la casilla de verificación del botón de opción
- **Checked**: Obtiene o establece un valor que indica si el control está activado.
- **TabIndex**
- **Text**

Métodos del control RadioButton

Los siguientes son algunos de los métodos comúnmente utilizados del control RadioButton:

- **PerformClick:** Genera un evento Click para el control, simulando un clic de un usuario.

Eventos del control RadioButton

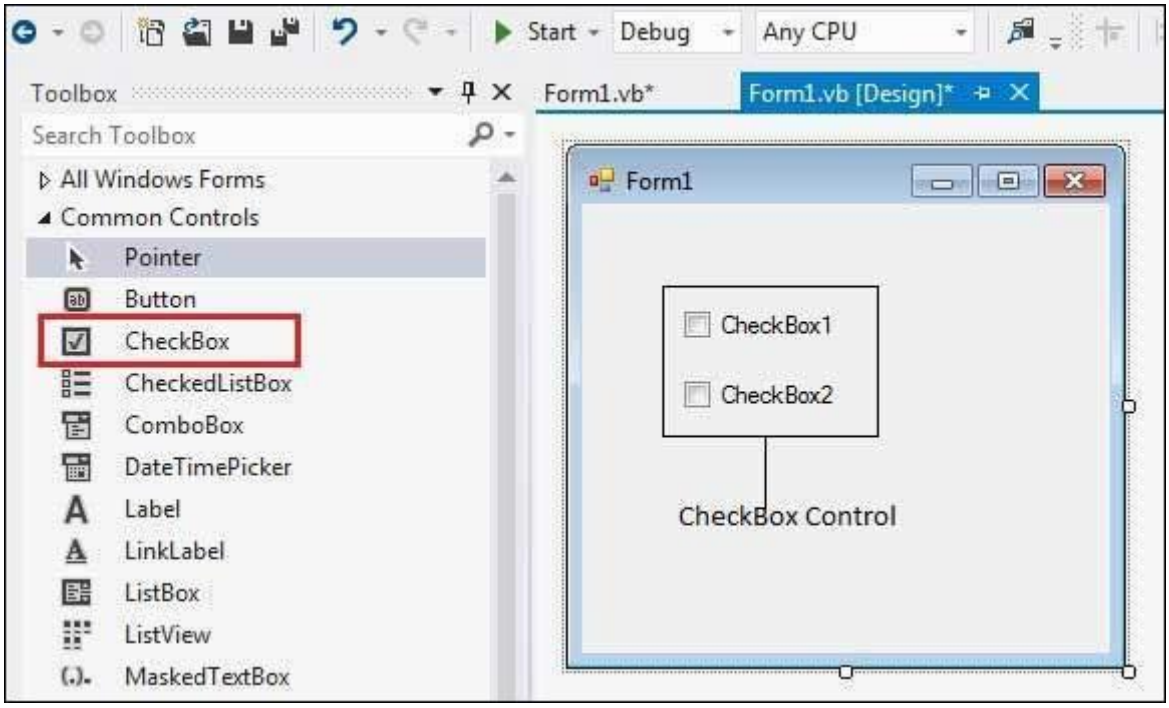
Los siguientes son algunos de los eventos de uso común del control RadioButton:

- **AppearanceChanged:** Se produce cuando se cambia el valor de la propiedad Appearance del control RadioButton.
- **CheckedChanged:** Se produce cuando se cambia el valor de la propiedad Checked del control RadioButton. **(Evento por defecto)**

CheckBox

El control CheckBox permite al usuario establecer opciones de tipo verdadero/falso o sí/no. El usuario puede seleccionarlo o deseleccionarlo. Cuando se selecciona una casilla de verificación, tiene el valor Verdadero, y cuando se desactiva, mantiene el valor Falso.

Vamos a crear dos casillas de verificación arrastrando los controles CheckBox desde la caja de herramientas y soltándolos en el formulario.



El control CheckBox tiene tres estados, **marcado**, **no marcado** e **indeterminado**. En el estado indeterminado, la casilla de verificación está atenuada. Para habilitar el estado indeterminado, la propiedad *ThreeState* de la casilla de verificación se establece en **True**.

Propiedades del control CheckBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control CheckBox:

- **Name**
- **Appearance:** Obtiene o establece un valor que determina la apariencia del botón de opción.
- **AutoCheck:** Obtiene o establece un valor que indica si el valor Checked y la apariencia del control cambian automáticamente cuando se hace clic en el control.
- **CheckAlign:** Obtiene o establece la ubicación de la parte de la casilla de verificación del botón de opción
- **Checked:** Obtiene o establece un valor que indica si el control está activado.
- **CheckState:** Obtiene o establece el estado de una casilla de verificación.
- **TabIndex**
- **Text**
- **ThreeState:** Obtiene o establece un valor que indica si una casilla de verificación debe permitir tres estados de verificación en lugar de dos.

Eventos del control CheckBox

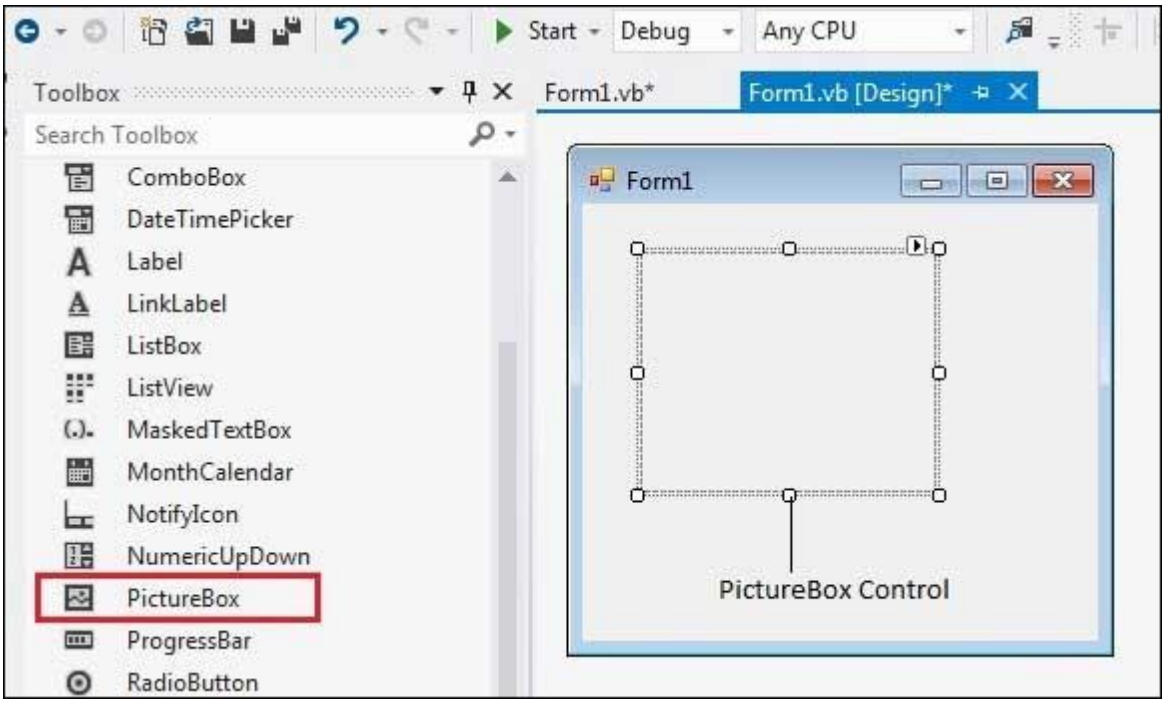
Los siguientes son algunos de los eventos de uso común del control CheckBox:

- **AppearanceChanged:** Se produce cuando se cambia el valor de la propiedad Appearance del control RadioButton.
- **CheckedChanged:** Se produce cuando se cambia el valor de la propiedad Checked del control RadioButton. **(Evento por defecto)**
- **CheckStateChanged:** Se produce cuando se cambia el valor de la propiedad CheckState del control CheckBox.

PictureBox

El control PictureBox se utiliza para mostrar imágenes en el formulario. La propiedad Imagen del control le permite establecer una imagen tanto en tiempo de diseño como en tiempo de ejecución.

Vamos a crear un cuadro de imagen arrastrando un control PictureBox desde el cuadro de herramientas y soltándolo en el formulario.



Propiedades del control PictureBox

Las siguientes son algunas de las propiedades comúnmente utilizadas del control PictureBox:

- **Name**
- **ErrorImage:** Obtiene o especifica una imagen que se mostrará cuando se produzca un error durante el proceso de carga de la imagen o si se cancela la carga de la imagen.
- **Image:** Obtiene o establece la imagen que se muestra en el control.
- **ImageLocation:** Obtiene o establece la ruta de acceso o la dirección URL de la imagen que se muestra en el control
- **InitialImage:** Obtiene o establece la imagen que se muestra en el control cuando se carga la imagen principal
- **SizeMode:** Determina el tamaño de la imagen que se mostrará en el control. Esta propiedad toma su valor de la enumeración PictureBoxSizeMode, que tiene valores:
  - **Normal**: la esquina superior izquierda de la imagen se coloca en la parte superior izquierda del cuadro de imagen
  - **StrechImage**: permite estirar la imagen
  - **Tamaño** automático: permite cambiar el tamaño del cuadro de imagen al tamaño de la imagen.

- **CenterImage**: permite centrar la imagen en el cuadro de imagen
- **Zoom**: permite aumentar o disminuir el tamaño de la imagen para mantener la relación de tamaño.

❖ **TabIndex**

❖ **WaitOnLoad:** Especifica si una imagen se carga sincrónicamente o no.

Métodos del PictureBox Control

Los siguientes son algunos de los métodos comúnmente utilizados del control PictureBox:

❖ **CancelAsync:** Cancela una carga de imagen asíncrona.

❖ **Load:** Muestra una imagen en el cuadro de imagen

❖ **LoadAsync:** Carga la imagen de forma asíncrona.

❖ **ToString:** Devuelve la cadena que representa el cuadro de imagen actual.

Eventos del PictureBox Control

Los siguientes son algunos de los eventos de uso común del control PictureBox:

❖ **Click (Evento por Defecto)**

❖ **Resize:** Se produce cuando se cambia el tamaño del control.

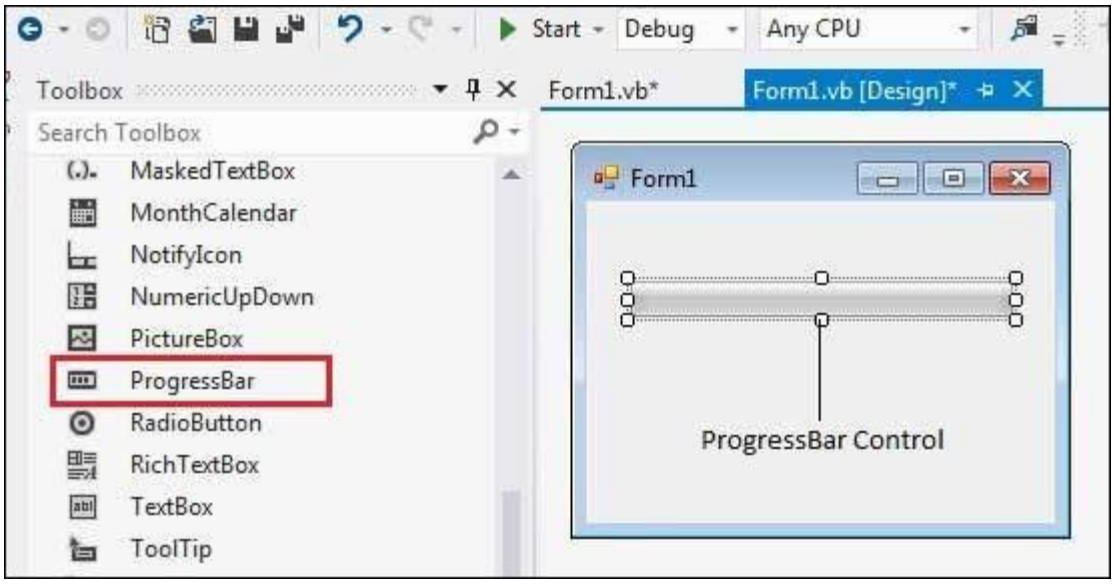
❖ **SizeChanged:** Se produce cuando cambia el valor de la propiedad Tamaño.

❖ **SizeModeChanged:** Ocurre cuando cambia SizeMode.

ProgressBar

Representa un control de la barra de progreso de Windows. Se utiliza para proporcionar información visual a sus usuarios sobre el estado de alguna tarea. Muestra una barra que se va rellenando de izquierda a derecha a medida que avanza la operación.

Hagamos clic en un control ProgressBar del cuadro de herramientas y colóquelo en el formulario.



Las principales propiedades de una barra de progreso son *Valor*, *Máximo* y *Mínimo*. Las propiedades Mínimo y Máximo se utilizan para establecer los valores mínimo y máximo que puede mostrar la barra de progreso. La propiedad Value especifica la posición actual de la barra de progreso.

El control ProgressBar generalmente se usa cuando una aplicación realiza tareas como copiar archivos o imprimir documentos. Para un usuario, la aplicación puede parecer que no responde si no hay una señal visual. En tales casos, el uso de ProgressBar permite al programador proporcionar un estado visual del progreso.

Propiedades del control ProgressBar

Las siguientes son algunas de las propiedades comúnmente utilizadas del control ProgressBar:

- **Name**
- **MarqueeAnimationSpeed:** Obtiene o establece el período de tiempo, en milisegundos, que tarda el bloque de progreso en desplazarse por la barra de progreso.
- **Maximun:** Obtiene o establece el valor máximo del rango del control.
- **Minimun:** Obtiene o establece el valor mínimo del rango del control.
- **Step:** Obtiene o establece la cantidad por la que una llamada al método PerformStep aumenta la posición actual de la barra de progreso.
- **Style:** Obtiene o establece la forma en que se debe indicar el progreso en la barra de progreso.
- **TabIndex**
- **Value:** Obtiene o establece la posición actual de la barra de progreso.

Métodos del control ProgressBar

Los siguientes son algunos de los métodos comúnmente utilizados del control ProgressBar:

- **Increment:** Incrementa la posición actual del control ProgressBar en la cantidad especificada.
- **PerformStep:** Incrementa el valor por el paso especificado.
- **ResetText:** Restablece la propiedad Texto a su valor predeterminado.
- **ToString:** Devuelve una cadena que representa el control de la barra de progreso.

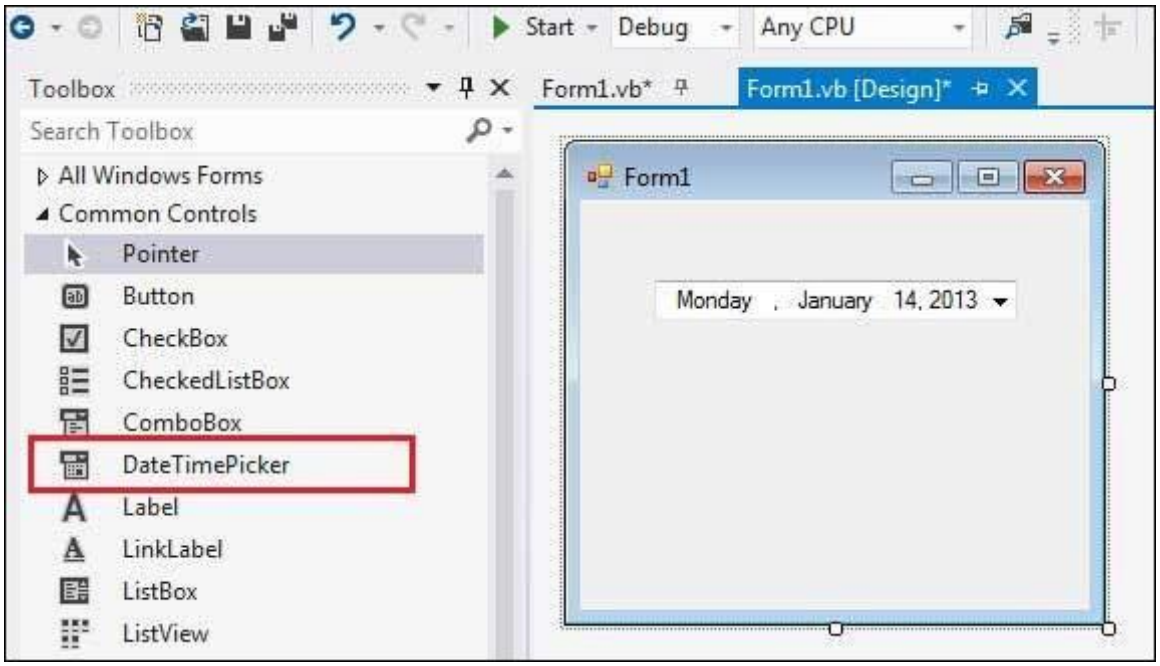
Eventos del control ProgressBar

Los siguientes son algunos de los eventos comúnmente utilizados del control ProgressBar:

- **Click (Evento Por Defecto)**
- **MouseClick:** Ocurre cuando se hace clic en el control con el mouse.
- **RightToLeftLayoutChanged:** Se produce cuando cambia la propiedad RightToLeftLayout.

DateTimePicker

El control DateTimePicker permite seleccionar una fecha y hora editando los valores mostrados en el control. Si hace clic en la flecha del control DateTimePicker, muestra un calendario mensual, como un control de cuadro combinado. El usuario puede hacer la selección haciendo clic en la fecha requerida. El nuevo valor seleccionado aparece en la parte del cuadro de texto del control.



Las **propiedades MinDate** y **MaxDate** le permiten poner límites en el rango de fechas.

Propiedades del control DateTimePicker

Las siguientes son algunas de las propiedades comúnmente utilizadas del control DateTimePicker:

- **Name**
- **CalendarFont:** Obtiene o establece el estilo de fuente aplicado al calendario.
- **CalendarForeColor:** Obtiene o establece el color de primer plano del calendario.
- **CalendarMonthBackGround:** Obtiene o establece el color de fondo del mes del calendario.
- **CalendarTitleBackColor:** Obtiene o establece el color de fondo del título del calendario.
- **CalendarTitleForeColor:** Obtiene o establece el color de primer plano del título del calendario.

- **CalendarTrailingForeColor:** Obtiene o establece el color de primer plano de las fechas finales del calendario.
- **Checked:** Obtiene o establece un valor que indica si la propiedad Value se ha establecido con un valor de fecha/hora válido y si el valor mostrado se puede actualizar.
- **CustomFormat:** Obtiene o establece la cadena de formato de fecha/hora personalizada.
- **DropDownAlign:** Obtiene o establece la alineación del calendario desplegable en el control DateTimePicker.
- **Font**
- **Format:** Obtiene o establece el formato de la fecha y hora que se muestra en el control.
- **MaxDate:** Obtiene o establece la fecha y la hora máximas que se pueden seleccionar en el control.
- **MinDate:** Obtiene o establece la fecha y la hora mínimas que se pueden seleccionar en el control.
- **RightToLeftLayout:** Obtiene o establece si el contenido de DateTimePicker se presenta de derecha a izquierda.
- **ShowCheckBox:** Obtiene o establece un valor que indica si se muestra una casilla de verificación a la izquierda de la fecha seleccionada.
- **ShowUpDown:** Obtiene o establece un valor que indica si se usa un control de botón giratorio (también conocido como control arriba-abajo) para ajustar el valor de fecha y hora.
- **TabIndex**
- **Text**
- **Value:** Obtiene o establece el valor de fecha y hora asignado al control.

Métodos del control DateTimePicker

Los siguientes son algunos de los métodos comúnmente utilizados del control DateTimePicker:

- **ToString:** Devuelve la cadena que representa el control.

Eventos del control DateTimePicker

Los siguientes son algunos de los eventos de uso común del control DateTimePicker:

- **CloseUp:** Ocurre cuando el calendario desplegable se descarta y desaparece.
- **DragDrop:** Se produce cuando se completa una operación de arrastrar y colocar.
- **FormatChanged:** Se produce cuando el valor de la propiedad Formato ha cambiado.
- **RightToLeftLayoutChanged:** Se produce cuando cambia la propiedad RightToLeftLayout.
- **ValueChanged:** Se produce cuando cambia la propiedad Value. (Evento por defecto)

DataGridView

El DataGridView es un objeto de visualización, una evolución del objeto DataGrid. Es una cuadricula en la que se pueden visualizar datos enlazados a una tabla, o provenientes de cualquier otra estructura de datos.

El enlace de las tablas a la cuadricula se realiza mediante la propiedad DataSource. Cuando se realiza este enlace no es posible añadir filas de forma manual al DataGrid.

Este control dispone de un sin fin de opciones de configuración para mejorar el formato de visualización. Por ejemplo, existe la clase DataGridViewCellStyle, que gestioina los estilos de formato de dicho objeto, si bien se recomienda no utilizarla cuando el volumen de datos es muy elevado.

Cuando empleamos un control DataGridView con su configuración de propiedades por defecto, el aspecto de los datos ofrecidos es correcto. No obstante, de la austeridad en su presentación se desprende la sensación de que con ciertos retoques estéticos, aunque los datos sigan siendo los mismos, el control ganaría enormemente en atractivo.

Para paliar esta carencia de vistosidad, es momento de que entren en el terreno de juego unos elementos que resultarán una inestimable ayuda en el trabajo cotidiano con el control DataGridView: los estilos.

Un estilo es un objeto de la clase DataGridViewCellStyle, que contiene una serie de propiedades relacionadas con los aspectos visuales de una celda, tales como colores, alineación, tipo de letra, etc. Actuando a todos los efectos como una plantilla de presentación de datos, que permite al programador aplicar uniformemente un conjunto de características visuales sobre la totalidad del control DataGridView, o sólo en determinadas regiones del mismo. En la siguiente tabla podemos ver las propiedades más importantes de esta clase.

Propiedad	Descripción
BackColor	Color de fondo para la celda
ForeColor	Color de primer plano para la celda
SelectionBackColor	Color de fondo para la celda en estado seleccionado
SelectionForeColor	Color de primer plano para la celda en estado seleccionado
Format	Cadena de formato para personalizar la presentación de datos numéricos, fechas, etc
Alignment	Valor de la enumeración DataGridViewContentAligment, que utilizaremos para establecer la alineación del contenido de la celda
Font	Tipo de letra con el que se mostrará el texto de la celda
WrapMode	En celdas con texto extenso, esta propiedad permite dividir dicho texto en varias líneas o truncarlo
NullValue	Tipo Object que utilizaremos para mostrar el valor de la celda cuando el campo original del origen de datos sea nulo
Padding	Establece el espacio de relleno para los márgenes de la celda

Propiedades del control DataGridView

- AllowDrop: Obtiene o establece un valor que indica si el control puede aceptar los datos que el usuario arrastra al mismo.
- AllowUserToAddRows: Permitir al usuario añadir filas.
- AllowUserToDeleteRows: Permitir al usuario eliminar las filas.
- AllowUserToOrderColumns: Permitir al usuario Ordenar las Columnas.
- AllowUserToResizeColumns: Permitir al usuario cambiar el tamaño de las columnas.
- AllowUserToResizeRows: Permitir al usuario cambiar el tamaño de las filas.
- AlternatingRowsDefaultCellStyle: Estilo de celdas al alternar filas.
- Anchor: Obtiene o establece los bordes del contenedor al que está enlazado un control y determina cómo se cambia el tamaño de un control con su elemento primario.
- AutoSizeColumnsMode: Tamaño de columnas automático.
- AutoSizeRowsMode: Tamaño de filas automático.
- BackGroundColor: Color de Fondo.
- BorderStyle: Estilo de los bordes.
- CausesValidation: Obtiene o establece un valor que indica si el control hace que se realice una validación de todos los controles que requieren validación cuando reciben el foco.
- CellBorderStyle: Estilo de los bordes de las celdas.
- ClipboardCopyMode: Modo de copiado del portapapeles.
- ColumnsHeaderBorderStyle: Estilo de los bordes del Encabezado de las columnas.
- ColumnsHeaderDefaultCellStyle: Estilo de las celdas de encabezado de las columnas.
- ColumnsHeaderHeight: Alto del encabezado de columnas.
- ColumnsHeaderHeightSizeMode: Modo de alto en encabezado de columnas.
- ColumnsHeaderVisible: Establece la visibilidad del encabezado de las columnas.
- Columns: Columnas.
- ContextMenuStrip: Obtiene o establece el ContextMenuStrip (menu contextual) asociado a este control.
- Cursor: Obtiene o establece el cursor que se muestra cuando el puntero del mouse se sitúa sobre el control.
- DataMember: Indica una sublista de DataSource para mostrar en el control DataGridView
- DataSource: Recurso de datos del datagridview
- DefaultCellStyle: Estilo predeterminado de las celdas
- Dock: Obtiene o establece que los bordes del control se acoplarán a su control principal y determina cómo se cambia el tamaño de un control con su elemento primario.
- EdiMode: Identifica el modo que determina como se inicia la edición de celdas
- Enabled: Obtiene o establece un valor que indica si el control puede responder a la interacción del usuario.
- GridColor: Color de la Grilla

- Locked: Habilita o deshabilita el uso de la grilla
- MultiSelect: Define la forma de seleccion de las filas y/o columnas del datagridview
- ReadOnly: Define si el control solamente es de lectura
- RowHeadersBorderStyle: El estilo de borde de las celdas de encabezado de las filas
- RowHeadersDefaultCellStyle: El estilo predeterminado que se aplicara a las celdas de encabezado de las filas
- RowHeadersVisible: Indica si se muestra la columna que contiene los encabezados de las filas
- RowHeadersWidth: El ancho, en pixeles, de la columna que contiene los encabezados de las filas
- RowHeadersWidthSizeMode: Determina el comportamiento para ajustar el ancho de los encabezados de fila
- RowsDefaultCellStyle: El estilo predeterminado que se aplica a las celdas de la fila de DataGridView
- RowTemplate: Identifica la fila de plantilla cuyas características se utilizan como base para todas las nuevas filas implícitamente agregadas.
- ScrollBars: Establece que barras de desplazamiento se muestran
- SelectionMode: Modo de selección de los datos del DataGridView (fila completa, etc.)
- ShowCellErrors: Indica si se mostraran los errores de celda
- ShowCellToolTips: Indica si se mostrara la información para herramientas cuando el puntero del mouse se detenga en una celda
- ShowEditingIcon: Indica si el glifo de edición esta visible en el encabezado de la fila de la celda que se esta editando
- ShowRowErrors: Indica si los encabezados de las filas mostraran glifos de error para cada fila que contenga errores de entrada de datos.
- StandardTab: Indica si la tecla tabulador mueve el foco al siguiente control en el orden de tabulación en lugar de moverlo a la siguiente celda en el control
- TabIndex: Obtiene o establece el orden de tabulación del control en su contenedor.
- TabStop: Obtiene o establece un valor que indica si el usuario puede dar el foco a este
- control mediante la tecla TAB.
- Tag: Obtiene o establece el objeto que contiene datos sobre el control.
- UseWaitCursor: Obtiene o establece un valor que indica si se utiliza el cursor de espera para
- el control actual y todos los controles secundarios.
- VirtualMode: Indica si se ha proporcionado sus propias operaciones de administración de datos para el control DataGridView
- Visible: Obtiene o establece un valor que indica si se muestran el control y todos sus controles primarios.

Eventos del control DataGridView

- AllowUserToAddRowsChanged: Se produce cuando cambia el valor de la propiedad AllowUserToAddRow.
- AllowUserToDeleteRowsChanged: Se produce cuando cambia el valor de la propiedad AllowUserToDeleteRowsChanged.
- AllowUserToOrderColumnsChanged: Se produce cuando cambia el valor de la propiedad AllowUserToOrderColumns.
- AllowUserToResizeColumnsChanged: Se produce cuando cambia el valor de la propiedad AllowUserToResizeColums
- AlternatingRowsDefaultCellStyleChanged: Se produce cuando cambia el valor de la propiedad AlternatingRowsDefaultCellStyle.
- AutoGenerateColumnsChanged: Se produce cuando cambia el valor de la propiedad AutoGenerateColumnsChanged
- AutoSizeColumnModeChanged: Se produce cuando cambia el valor de la propiedad AutoSizeMode de una columna.
- AutoSizeColumnsModeChanged: Se produce cuando cambia el valor de la propiedad AutoSizeColumnsMode.
- AutoSizeRowsModeChanged: Se produce cuando cambia el valor de la propiedad DataGridViewAutoSizeRowsMode
- BackColorChanged: Este evento admite la infraestructura de .NET Framework y no se debe usar directamente a partir del código. Se produce cuando cambia el valor de la propiedad BackColor
- BackgroundColorChanged: Se produce cuando cambia el valor de la propiedad BackgroundColor
- BackgroundImageChanged: Este evento admite la infraestructura de .NET Framework y no se debe usar directamente a partir del código. Se produce cuando cambia el valor de la propiedad BackgroundImage.
- BackgroundImageLayoutChanged: Este evento admite la infraestructura de .NET framework y no debe usarse directamente a partir del código. Se produce cuando cambia la propiedad BackgroundImageLayout.
- BorderStyleChanged: Se produce cuando cambia el valor de la propiedad BorderStyle.
- CancelRowEdit: Se produce cuando la propiedad VirtualMode de Un Control DataGridView es true y se cancela la edición de una fila.
- CellBeginEdit: Se produce cuando se inicia el modo de edición para la celda seleccionada.
- CellBorderStyleChanged: Se produce cuando cambia el estilo de borde de una celda.
- CellClick: Se produce cuando se hace clic en cualquier parte de una celda.
- CellContentClick: Se produce cuando se hace clic en el contenido de una celda.
- CellContentDoubleClick: Se produce cuando el usuario hace doble clic en el contenido de una celda.
- CellContextMenuStripChanged: Se produce cuando cambia el valor de la propiedad ContextMenuStrip.
- CellContextMenuStripNeeded: Se produce cuando se necesita el menú contextual de una celda.
- CellDoubleClick: Se produce cuando el usuario hace doble clic en cualquier parte de una celda.

- CellEndEdit: Se produce cuando se detiene el modo de edición para la celda seleccionada actualmente.
- CellEnter: Se produce cuando la celda actual cambia en el control DataGridView o cuando el control recibe el foco de entrada.
- CellErrorTextChanged: Se produce cuando cambia el valor de la propiedad ErrorText de una celda.
- CellErrorTextNeeded: Se produce cuando es necesario el texto de error de una celda.
- CellFormatting: Se produce cuando hay que aplicar formato al contenido de una celda para su presentación.
- CellLeave: Se produce cuando una celda pierde el foco de entrada y deja de ser la celda actual.
- CellMouseClick: Aparece siempre que el usuario hace clic en cualquier parte de una celda con el mouse.
- CellMouseDoubleClick: Se produce cuando se hace doble clic en una celda del control DataGridView
- CellMouseDown: Se produce cuando el usuario presiona un botón del mouse mientras el puntero del mouse está dentro de los límites de una celda.
- CellMouseEnter: Se produce cuando el puntero del mouse entra en una celda.
- CellMouseLeave: Se produce cuando el puntero del mouse sale de una celda.
- CellMouseMove: Se produce cuando el puntero del mouse se mueve sobre el control DataGridView
- CellMouseUp: Se produce cuando el usuario libera un botón del mouse mientras esta sobre una celda.
- CellPainting: Se produce cuando hay que dibujar una celda.
- CellParsing: Se produce cuando una celda cuyo valor ha sido modificado deja de estar en modo de edición.
- CellStateChanged: Se produce cuando cambia el estado de una celda, como cuando la celda pierde o recibe el foco.
- CellStyleChanged: Se produce cuando cambia la propiedad Style de un control DataGridViewCell
- CellStyleContentChanged: Se produce cuando uno de los valores de un estilo de celda cambia.
- CellToolTipTextChanged: Se produce cuando cambia el valor de la propiedad ToolTipText para una celda del control DataGridView.
- CellToolTipTextNeeded: Se produce cuando es necesario el texto de información sobre herramientas de una celda.
- CellValidated: Se produce cuando la celda termina de validar.
- CellValidating: Se produce cuando una celda pierde el foco de entrada. Habilita la validación de contenido.
- CellValueChanged: Se produce cuando cambia el valor de una celda
- CellValueNeeded: Se produce cuando el valor de la propiedad VirtualMode del control DataGridView es true y el control DataGridView requiere un valor para una celda a fin de aplicar formato a la celda y mostrarla.
- CellValuePushed: Se produce cuando la propiedad VirtualMode del control DataGridView es true y un valor de celda ha cambiado y requiere almacenamiento en el origen de datos subyacente.
- ColumnAdded: Se produce cuando se agrega una columna al control.
- ColumnContextMenuStripChanged: Se produce cuando cambia la propiedad ContextMenuStrip de una columna.

- ColumnDataPropertyNameChanged: Se produce cuando cambia el valor de la propiedad DataPropertyName para una columna.
- ColumnDefaultCellStyleChanged: Se produce cuando cambia el valor de la propiedad DefaultCellStyle para una columna.
- ColumnDisplayIndexChanged: Se produce cuando cambia el valor de la propiedad DisplayIndex para una columna.
- ColumnDividerDoubleClick: Se produce cuando el usuario hace doble clic en un divisor entre dos columnas.
- ColumnDividerWidthChanged: Se produce cuando cambia la propiedad DividerWidth.
- ColumnHeaderCellChanged: Se produce cuando cambia el contenido de una celda de encabezado de columna.
- ColumnHeaderMouseClick: Se produce cuando el usuario hace clic en un encabezado de columna.
- ColumnHeaderMouseDoubleClick: Se produce cuando se hace doble clic en un encabezado de columna.
- ColumnHeadersBorderStyleChanged: Se produce cuando cambia la propiedad ColumnHeaderBorderStyle.
- ColumnHeadersDefaultCellStyleChanged: Se produce cuando cambia el valor de la propiedad ColumnHeadersDefaultCellStyle.
- ColumnHeadersHeightChanged: Se produce cuando cambia el valor de la propiedad ColumnHeadersHeight.
- ColumnHeadersHeightSizeModeChanged: Se produce cuando cambia el valor de la propiedad ColumnHeadersHeightSizeMode.
- ColumnMinimumWidthChanged: Se produce cuando cambia el valor de la propiedad MinimumWidth para una columna.
- ColumnNameChanged: Se produce cuando cambia el valor de la propiedad Name para una columna.
- ColumnRemoved: Se produce cuando se quita una columna del control.
- ColumnSortModeChanged: Se produce cuando cambia el valor de la propiedad SortMode para una columna.
- ColumnStateChanged: Se produce cuando una columna cambia de estado (p.ej., recibe o pierde el foco).
- ColumnToolTipTextChanged: Se produce cuando cambia el valor de la propiedad ToolTipText para una columna del control DataGridView
- ColumnWidthChanged: Se produce cuando cambia el valor de la propiedad Widht para una columna.
- CurrentCellChanged: Se produce cuando cambia la propiedad CurrentCell.
- CurrentCellDirtyStateChanged: Se produce cuando el estado de una celda cambia en relación con un cambio de su contenido.
- DataBindingComplete: Se produce después que haya finalizados una operación de enlace de datos.
- DataError: Se produce cuando una operación de análisis o validación de datos externos produce una excepción o cuando se produce un error al intentar confirmar datos de un origen de datos.
- DataMemberChanged: Se produce cuando cambia el valor de la propiedad DataMember.
- DataSourceChanged: Se produce cuando cambia el valor de la propiedad DataSource.
- DefaultCellStyleChanged: se produce cuando cambia el valor de la propiedad DefaultCellStyle.

- DefaultValuesNeeded: Se produce cuando el usuario especifica la fila para nuevos registros de forma que se pueda rellenar con valores predeterminados.
- EditingControlShowing: Se produce cuando se muestra un control para editar una celda.
- EditModeChanged: Se produce cuando cambia el valor de la propiedad EditMode.
- FontChanged: Se produce cuando cambia el valor de la propiedad Font.
- ForeColorChanged: Se produce cuando cambia el valor de la propiedad ForeColor.
- GridColorChanged: Se produce cuando cambia el valor de la propiedad GridColor.
- MultiSelectChanged: Se produce cuando cambia el valor de la propiedad MultiSelect
- NewRowNeeded: Se produce cuando la propiedad VirtualMode de DataGridView es true y el usuario se desplaza a la nueva fila en la parte inferior del control DataGridView.
- PaddingChanged: Este evento admite la infraestructura de .NET Framework y no se debe usar directamente a partir del código. Se produce cuando cambia el valor de la propiedad Padding
- ReadOnlyChanged: Se produce cuando cambia la propiedad ReadOnly.
- RowContextMenuStripChanged: Se produce cuando cambia el valor de la propiedad ContextMenuStrip.
- RowContextMenuStripNeeded: Se produce cuando se necesita el menú contextual de una fila.
- RowDefaultStyleChanged: Se produce cuando cambia el valor de la propiedad DefaultCellStyle para una fila.
- RowDirtyStateNeeded: Se produce cuando la propiedad del control DataGridView es true y el control DataGridView necesita determinar si se han confirmado los cambios de la fila actual.
- RowDividerDoubleClick: Se produce cuando el usuario hace doble clic en un divisor entre dos filas
- RowDividerHeightChanged: Se produce cuando cambia la propiedad DividerHeight.
- RowEnter: Se produce cuando una fila recibe el foco de entrada, pero antes de que se convierta en la fila actual.
- RowErrorTextChanged: Se produce cuando cambia la propiedad ErrorText de una fila.
- RowErrorTextNeeded: Se produce cuando es necesario el texto de error de una fila.
- RowHeaderCellChanged: Se produce cuando el usuario cambia el contenido de una celda de encabezado de fila.
- RowHeaderMouseClick: Se produce cuando el usuario hace clic dentro los límites de un encabezado de fila.
- RowHeaderMouseDoubleClick: Se produce cuando se hace doble clic en un encabezado de fila.
- RowHeadersBorderStyleChanged: Se produce cuando cambia la propiedad RowHeadersBorderStyle.
- RowHeadersDefaultCellStyleChanged: Se produce cuando cambia la propiedad RowHeadersDefautlCellStyle.
- RowHeadersWidthChanged: Se produce cuando cambia el valor de la propiedad RowHeadersWidth.
- RowHeadersWidthSizeModeChanged: Se produce cuando cambia el valor de la propiedad RowHeadersWidthSizeMode.
- RowHeightChanged: Se produce cuando cambia el valor de la propiedad Height para una fila.
- RowHeightInfoNeeded: Se produce cuando se solicita información sobre el alto de fila.

- RowHeightInfoPushed: Se produce cuando el usuario camba el alto de una fila.
- RowLeave: Se produce cuando una fila pierde el foco de entrada y deja de ser la fila actual.
- RowMinimumHeightChanged: Se produce cuando cambia el valor de la propiedad MinimumHeight para una fila.
- RowPostPaint: Se produce después de que se dibuje un objeto DataGridViewRow.
- RowPrePaint: Se produce antes de que se dibuje un objeto DataGridView.
- RowsAdded: Se produce después de agregar una nueva fila a DataGridView.
- RowsDefaultCellStyleChanged: Se produce cuando cambia el valor de la propiedad RowsDefaultCellStyle.
- RowsRemoved: Se produce cuando se elimina una o mas filas del control DataGridView
- RowStateChanged: Se produce cuando cambia el estado de una fila, como cuando pierde o recibe el foco de entrada.
- RowUnshared: Se produce cuando el estado de una fila cambia de compartido a no compartido.
- RowValidated: Se produce cuando finaliza la validación de una fila.
- RowValidating: Se produce cuando se esta validando una fila.
- Scroll: Se produce cuando el usuario se desplaza por el contenido del control.
- SelectionChanged: Se produce cuando cambia la selección actual.
- SortCompare: Se produce cuando DataGridView compara dos valores de celda para realizar una operación de comparación.
- Sorted: Se produce cuando el control DataGridView finaliza una operación de ordenación.
- StyleChanged: Este evento admite la infraestructura de .NET Framework y no se debe usar directamente a partir del código. Se produce cuando cambia el estilo del control.
- TextChanged: Este evento admite la infraestructura de .NET Framework y no se debe usar directamente a partir del código. Se produce cuando cambia el valor de la propiedad Text.
- UserAddedRow: Se produce cuando el usuario acaba de agregar una fila al control DataGridView.
- UserDeletedRow: Se produce cuando el usuario acaba de eliminar una fila del control DataGridView.
- UserDeletingRow: Se produce cuando el usuario elimina una fila del control DataGridView.

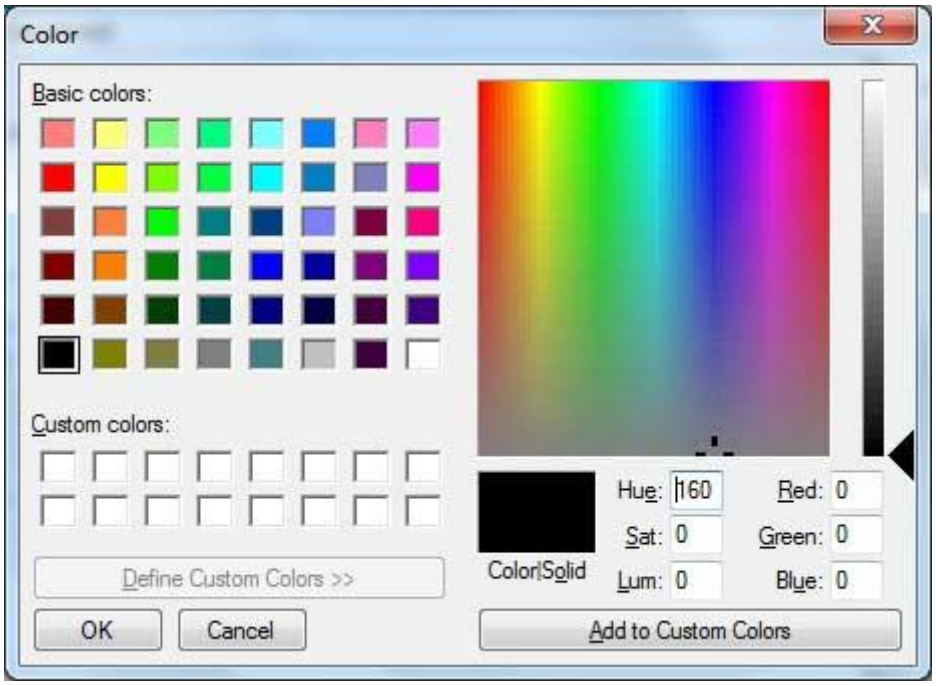
Cuadros de dialogo

ColorDialog

La clase de control ColorDialog representa un cuadro de diálogo común que muestra los colores disponibles junto con controles que permiten al usuario definir colores personalizados. Permite al usuario seleccionar un color.

La propiedad principal del control ColorDialog es *Color*, que devuelve un objeto **Color**.

A continuación, se muestra el cuadro de diálogo Color:



Propiedades del control ColorDialog

Las siguientes son algunas de las propiedades comúnmente utilizadas del control ColorDialog:

- **AllowFullOpen:** Obtiene o establece un valor que indica si el usuario puede usar el cuadro de diálogo para definir colores personalizados.
- **AnyColor:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra todos los colores disponibles en el conjunto de colores básicos.
- **Color:** Obtiene o establece el color seleccionado por el usuario.
- **FullOpen:** Obtiene o establece un valor que indica si los controles usados para crear colores personalizados están visibles cuando se abre el cuadro de diálogo.
- **ShowHelp:** Obtiene o establece un valor que indica si aparece un botón de Ayuda en el cuadro de diálogo de color.
- **SolidColorOnly:** Obtiene o establece un valor que indica si el cuadro de diálogo restringirá a los usuarios a seleccionar solo colores sólidos.

Métodos del control ColorDialog

Los siguientes son algunos de los métodos comúnmente utilizados del control ColorDialog:

- **Reset:** Restablece todas las opciones a sus valores predeterminados, el último color seleccionado a negro y los colores personalizados a sus valores predeterminados.
- **ShowDialog:** Ejecuta un cuadro de diálogo común con un propietario predeterminado.

Eventos del Control ColorDialog

Los siguientes son algunos de los eventos de uso común del control ColorDialog:

- **HelpRequest:** Ocurre cuando el usuario hace clic en el botón Ayuda en un cuadro de diálogo común.

FontDialog

Solicita al usuario que elija una fuente entre las instaladas en la computadora local y le permite seleccionar la fuente, el tamaño de fuente y el color. Devuelve los objetos Font y Color.

A continuación, se muestra el cuadro de diálogo Fuente:

De forma predeterminada, el cuadro combinado de colores no se muestra en el cuadro de diálogo Fuente. Debe establecer la propiedad **ShowColor** del control FontDialog en **True**.

Propiedades del Control FontDialog

Las siguientes son algunas de las propiedades comúnmente utilizadas del control FontDialog:

- **AllowSimulations:** Obtiene o establece un valor que indica si el cuadro de diálogo permite simulaciones de fuentes de interfaz de dispositivo gráfico (GDI).
- **AllowVectorFonts:** Obtiene o establece un valor que indica si el cuadro de diálogo permite selecciones de fuentes vectoriales.

- **AllowVerticalFonts:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra fuentes verticales y horizontales, o solo fuentes horizontales.
- **Color:** Obtiene o establece el color de fuente seleccionado.
- **FixedPitchOnly:** Obtiene o establece un valor que indica si el cuadro de diálogo solo permite la selección de fuentes de paso fijo.
- **Font:** Obtiene o establece la fuente seleccionada.
- **FontMustExist:** Obtiene o establece un valor que indica si el cuadro de diálogo especifica una condición de error si el usuario intenta seleccionar una fuente o estilo que no existe.
- **MaxSize:** Obtiene o establece el tamaño máximo en puntos que puede seleccionar un usuario.
- **MinSize:** Obtiene o establece el tamaño mínimo en puntos que un usuario puede seleccionar.
- **ScriptsOnly:** Obtiene o establece un valor que indica si el cuadro de diálogo permite la selección de fuentes para todos los conjuntos de caracteres que no sean OEM y símbolos, así como el conjunto de caracteres ANSI.
- **ShowApply:** Obtiene o establece un valor que indica si el cuadro de diálogo contiene el botón **Aplicar**
- **ShowColor:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra la opción de color.
- **ShowEffects:** Obtiene o establece un valor que indica si el cuadro de diálogo contiene controles que permiten al usuario especificar las opciones de tachado, subrayado y color del texto.
- **ShowHelp:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra un botón de Ayuda.

Métodos del Control FontDialog

Los siguientes son algunos de los métodos comúnmente utilizados del control FontDialog:

- **Reset:** Restablece todas las opciones a sus valores predeterminados.
- **ShowDialog:** Ejecuta un cuadro de diálogo común con un propietario predeterminado.

Eventos del control FontDialog

Los siguientes son algunos de los eventos comúnmente utilizados del control FontDialog:

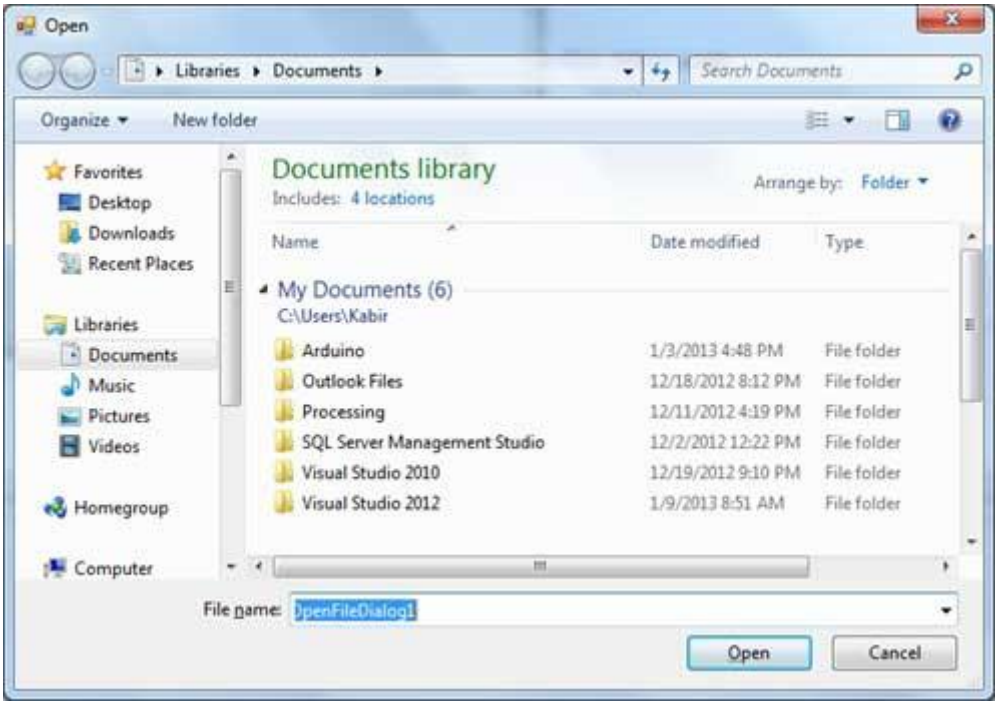
- **Apply:** Ocurre cuando se hace clic en el botón Aplicar en el cuadro de diálogo de la fuente. (Evento por defecto)
- **HelpRequest:** Ocurre cuando el usuario hace clic en el botón Ayuda en un cuadro de diálogo común.

OpenFileDialog

El control **OpenFileDialog** solicita al usuario que abra un archivo y le permite seleccionar un archivo para abrir. El usuario puede verificar si el archivo existe y luego abrirlo. La clase de control OpenFileDialog hereda de la clase abstracta **FileDialog** .

Si la propiedad ShowReadOnly se establece en True, aparece una casilla de verificación de solo lectura en el cuadro de diálogo. También puede establecer la propiedad ReadOnlyChecked en True, de modo que la casilla de verificación de solo lectura aparezca marcada.

A continuación, se muestra el cuadro de diálogo Abrir archivo:



Propiedades del control OpenFileDialog

Las siguientes son algunas de las propiedades comúnmente utilizadas del control OpenFileDialog:

- **AddExtension:** Obtiene o establece un valor que indica si el cuadro de diálogo agrega automáticamente una extensión a un nombre de archivo si el usuario omite la extensión.
- **AutoUpgradeEnabled:** Obtiene o establece un valor que indica si esta instancia de FileDialog debe actualizar automáticamente la apariencia y el comportamiento cuando se ejecuta en Windows Vista.
- **CheckFileExists:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra una advertencia si el usuario especifica un nombre de archivo que no existe.
- **CheckPathExists:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra una advertencia si el usuario especifica una ruta que no existe.
- **DefaultExt:** Obtiene o establece la extensión de nombre de archivo predeterminada.
- **DereferenceLinks:** Obtiene o establece un valor que indica si el cuadro de diálogo devuelve la ubicación del archivo al que hace referencia el acceso directo o si devuelve la ubicación del acceso directo (.lnk).
- **FileName:** Obtiene o establece una cadena que contiene el nombre del archivo seleccionado en el cuadro de diálogo del archivo.
- **Filter:** Obtiene o establece la cadena de filtro de nombre de archivo actual, que determina las opciones que aparecen en el cuadro "Guardar como tipo de archivo" o "Archivos de tipo" en el cuadro de diálogo.
- **FilterIndex:** Obtiene o establece el índice del filtro actualmente seleccionado en el cuadro de diálogo del archivo.
- **InitialDirectory:** Obtiene o establece el directorio inicial que se muestra en el cuadro de diálogo del archivo.

- **MultiSelect:** Obtiene o establece un valor que indica si el cuadro de diálogo permite seleccionar varios archivos.
- **ReadOnlyChecked:** Obtiene o establece un valor que indica si la casilla de verificación de solo lectura está seleccionada.
- **RestoreDirectory:** Obtiene o establece un valor que indica si el cuadro de diálogo restaura el directorio actual antes de cerrarse.
- **ShowHelp:** Obtiene o establece un valor que indica si el botón Ayuda se muestra en el cuadro de diálogo del archivo.
- **ShowReadOnly:** Obtiene o establece un valor que indica si el cuadro de diálogo contiene una casilla de verificación de solo lectura.
- **SupportMultiDottedExtensions:** Obtiene o establece si el cuadro de diálogo admite mostrar y guardar archivos que tienen varias extensiones de nombre de archivo.
- **Title:** Obtiene o establece el título del cuadro de diálogo del archivo.
- **ValidateNames:** Obtiene o establece un valor que indica si el cuadro de diálogo solo acepta nombres de archivo Win32 válidos.

Métodos del control OpenFileDialog

Los siguientes son algunos de los métodos comúnmente utilizados del control OpenFileDialog:

- **OpenFile:** Abre el archivo seleccionado por el usuario, con permiso de solo lectura. El archivo se especifica mediante la propiedad FileName.
- **Reset:** Restablece todas las opciones a su valor predeterminado.

Eventos del control OpenFileDialog

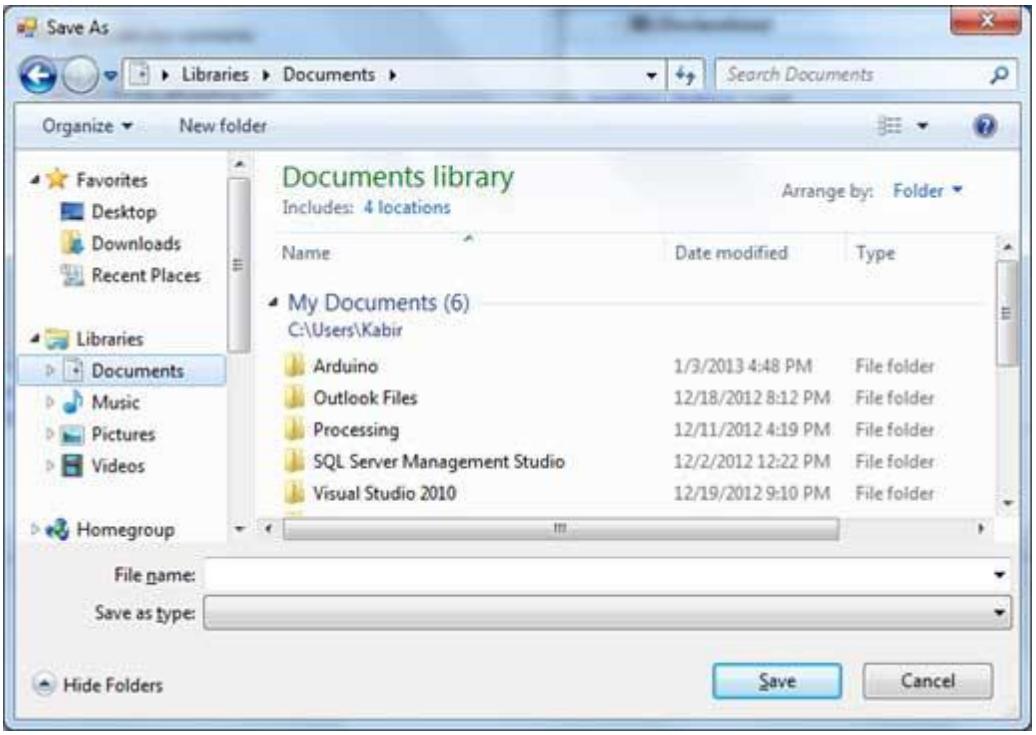
Los siguientes son algunos de los eventos comúnmente utilizados del control OpenFileDialog:

- **FileOk:** Tiene lugar cuando el usuario hace click en el botón Abrir o Guardar del cuadro de dialogo . (Evento por defecto)
- **HelpRequest:** Ocurre cuando el usuario hace clic en el botón Ayuda en un cuadro de diálogo común.

SaveFileDialog

El control **SaveFileDialog** solicita al usuario que seleccione una ubicación para guardar un archivo y le permite especificar el nombre del archivo para guardar los datos. La clase de control SaveFileDialog hereda de la clase abstracta FileDialog.

A continuación, se muestra el cuadro de diálogo Guardar archivo:



Propiedades del control SaveFileDialog

Las siguientes son algunas de las propiedades comúnmente utilizadas del control SaveFileDialog:

- **AddExtension:** Obtiene o establece un valor que indica si el cuadro de diálogo agrega automáticamente una extensión a un nombre de archivo si el usuario omite la extensión.
- **CheckFileExists:** Obtiene o establece un valor que indica si el cuadro de diálogo muestra una advertencia si el usuario especifica un nombre de archivo que no existe.
- **CheckPathExists**: Obtiene o establece un valor que indica si el cuadro de diálogo muestra una advertencia si el usuario especifica una ruta que no existe.
- **CreatePrompt:** Obtiene o establece un valor que indica si el cuadro de diálogo solicita permiso al usuario para crear un archivo si el usuario especifica un archivo que no existe.
- **DefaultExt:** Obtiene o establece la extensión de nombre de archivo predeterminada.
- **DereferenceLinks**: Obtiene o establece un valor que indica si el cuadro de diálogo devuelve la ubicación del archivo al que hace referencia el acceso directo o si devuelve la ubicación del acceso directo (.lnk).
- **FileName:** Obtiene o establece una cadena que contiene el nombre del archivo seleccionado en el cuadro de diálogo del archivo.

- **Filter:** Obtiene o establece la cadena de filtro de nombre de archivo actual, que determina las opciones que aparecen en el cuadro "Guardar como tipo de archivo" o "Archivos de tipo" en el cuadro de diálogo.
- **FilterIndex:** Obtiene o establece el índice del filtro actualmente seleccionado en el cuadro de diálogo del archivo.
- **InitialDirectory:** Obtiene o establece el directorio inicial que se muestra en el cuadro de diálogo del archivo.
- **OverwritePrompt:** Obtiene o establece un valor que indica si el cuadro de diálogo Guardar como muestra una advertencia si el usuario especifica un nombre de archivo que ya existe.
- **RestoreDirectory:** Obtiene o establece un valor que indica si el cuadro de diálogo restaura el directorio actual antes de cerrarse.
- **ShowHelp:** Obtiene o establece un valor que indica si el botón Ayuda se muestra en el cuadro de diálogo del archivo.
- **SupportMultiDottedExtensions:** Obtiene o establece si el cuadro de diálogo admite mostrar y guardar archivos que tienen varias extensiones de nombre de archivo.
- **Title:** Obtiene o establece el título del cuadro de diálogo del archivo.
- **ValidateNames:** Obtiene o establece un valor que indica si el cuadro de diálogo solo acepta nombres de archivo Win32 válidos.

Métodos del control SaveFileDialog

Los siguientes son algunos de los métodos comúnmente utilizados del control SaveFileDialog:

- **OpenFile:** Abre el archivo con permiso de lectura/escritura.
- **Reset:** Restablece todas las opciones del cuadro de diálogo a sus valores predeterminados.

Eventos del control SaveFileDialog

Los siguientes son algunos de los eventos comúnmente utilizados del control SaveFileDialog:

- **FileOk:** Tiene lugar cuando el usuario hace click en el botón Abrir o Guardar del cuadro de dialogo . (Evento por defecto)
- **HelpRequest:** Ocurre cuando el usuario hace clic en el botón Ayuda en un cuadro de diálogo común.

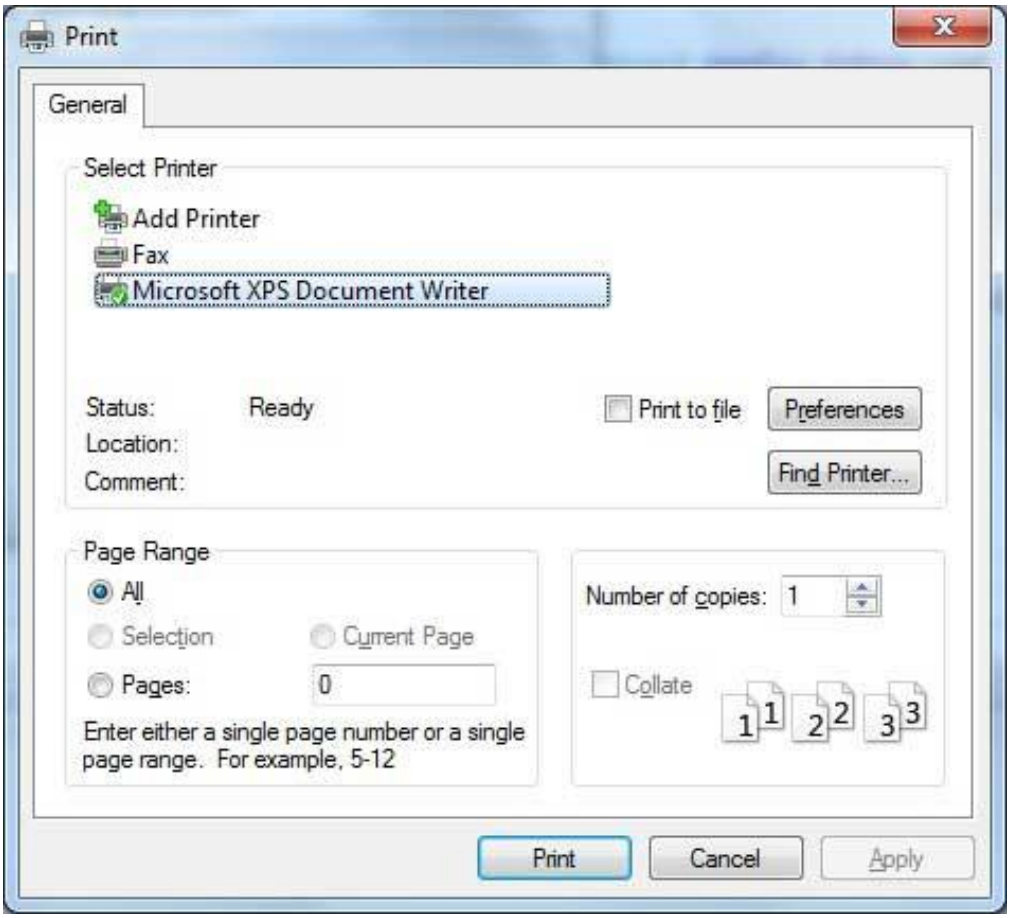
PrintDialog

El control PrintDialog permite al usuario imprimir documentos seleccionando una impresora y eligiendo qué secciones del documento imprimir desde una aplicación de Windows Forms.

Hay varios otros controles relacionados con la impresión de documentos. Echemos un breve vistazo a estos controles y su propósito. Estos otros controles son:

- El control **PrintDocument**: brinda soporte para eventos y operaciones reales de impresión en Visual Basic y establece las propiedades para la impresión.
- El control **PrinterSettings**: se utiliza para configurar cómo se imprime un documento especificando la impresora.
- El control **PageSetUpDialog**: permite al usuario especificar la configuración de impresión relacionada con la página, incluida la orientación de la página, el tamaño del papel y el tamaño del margen.
- El control **PrintPreviewControl**: representa la parte de vista previa sin formato de la vista previa de impresión desde una aplicación de Windows Forms, sin cuadros de diálogo ni botones.
- El control **PrintPreviewDialog**: representa un formulario de cuadro de diálogo que contiene un PrintPreviewControl para imprimir desde una aplicación de Windows Forms.

A continuación, se muestra el cuadro de diálogo Imprimir:



Propiedades del control PrintDialog

Las siguientes son algunas de las propiedades comúnmente utilizadas del control PrintDialog:

- **AllowCurrentPage:** Obtiene o establece un valor que indica si se muestra el botón de opción **Página actual.**
- **AllowPrintToFile:** Obtiene o establece un valor que indica si la casilla de verificación **Imprimir en archivo** está habilitada.
- **AllowSelection:** Obtiene o establece un valor que indica si el botón de opción **Selección** está habilitado.
- **AllowSomePages**: Obtiene o establece un valor que indica si el botón de opción **Páginas** está habilitado.
- **Document:** Obtiene o establece un valor que indica el PrintDocument usado para obtener PrinterSettings.
- **PrintToFile:** Obtiene o establece un valor que indica si la casilla de verificación **Imprimir en archivo** está se leccionada.
- **ShowHelp:** Obtiene o establece un valor que indica si se muestra el botón **Ayuda.**
- **ShowNetwork:** Obtiene o establece un valor que indica si se muestra el botón **Red.**

Métodos del control PrintDialog

Los siguientes son algunos de los métodos comúnmente utilizados del control PrintDialog:

- **Reset:** Restablece todas las opciones a sus valores predeterminados.
- **ShowDialog:** Ejecuta un cuadro de diálogo común con un propietario predeterminado.

MessageBox

MessageBox en Windows Forms muestra un mensaje con el texto y los botones de acción dados. También puede usar el control MessageBox para agregar opciones adicionales, como un título, un icono o botones de ayuda.

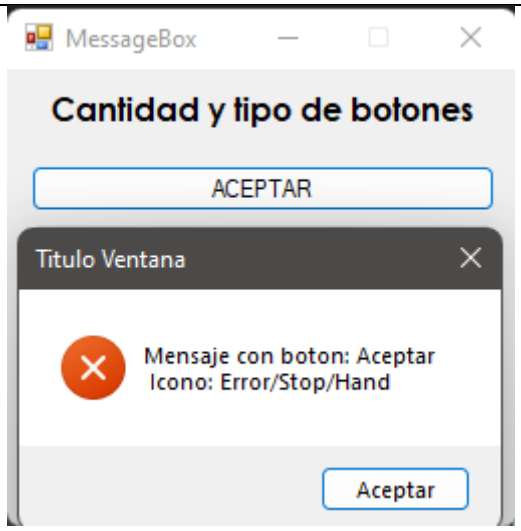
La clase MessageBox tiene un método Show estático sobrecargado que muestra un cuadro de mensaje con un mensaje y botones de acción. Los botones de acción pueden ser:

- Aceptar
- Aceptar – Cancelar
- Sí – No
- Si – No – Cancelar
- Reintentar – Cancelar
- Anular – Reintentar – Omitir

Aceptar

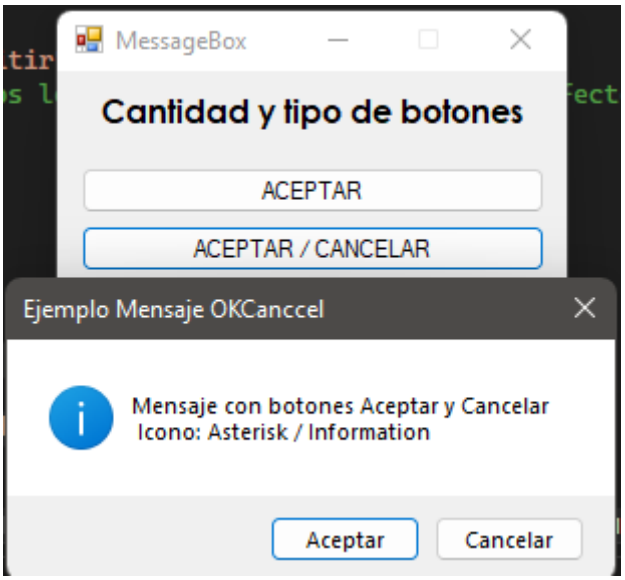
```
private void btnAceptar_Click(object sender, EventArgs e)
        {
            //Mensaje simple de un solo botón
            MessageBox.Show("Mensaje con boton: Aceptar \n Icono: Error/Stop/Hand", //
mensaje
                "Titulo Ventana",                    // titulo
                MessageBoxButtons.OK,                // tipo de boton
                MessageBoxIcon.Error);         // tipo icono
```

```
            //Mensaje simple donde podríamos obviar El titulo, el tipo de botón y el icono
            MessageBox.Show("Esto es un mensaje simple");
        }
```



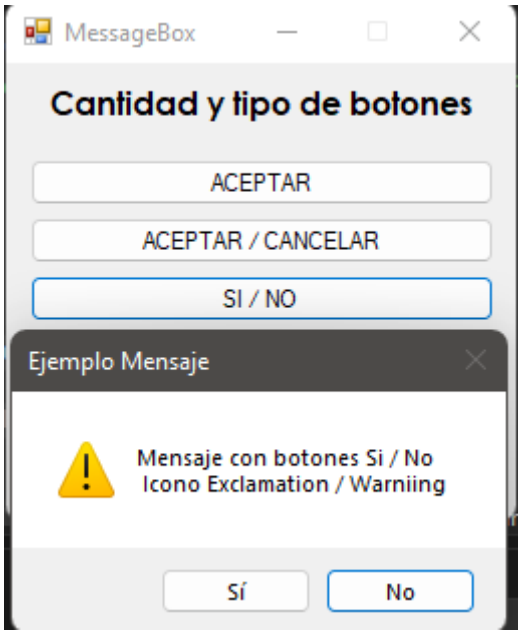
Aceptar – Cancelar

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void btnAceptarCancelar_Click(object sender, EventArgs e)
        {
            Resultado = MessageBox.Show("Mensaje con botones Aceptar y Cancelar \n Icono:
Asterisk / Information", // mensaje
                "Ejemplo Mensaje OKCanccel",        // titulo
                MessageBoxButtons.OKCancel,         // tipo boton
                MessageBoxIcon.Asterisk,            // tipo icono
                MessageBoxDefaultButton.Button1);   // Botón predeterminado

            if (Resultado == DialogResult.OK)
            {
                MessageBox.Show("Seleccionaste Aceptar");
            }
            else
            {
                MessageBox.Show("Seleccionaste Cancelar");
                return; //Con este comando, dejamos lo que estuvieramos haciendo sin efecto
            }
        }
```



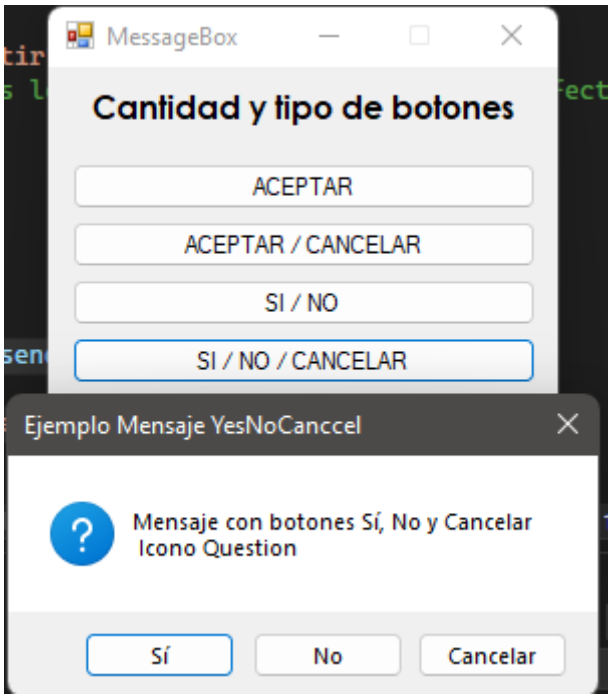
Si – No

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void btnSiNo_Click(object sender, EventArgs e)
        {
            Resultado = MessageBox.Show("Mensaje con botones Si / No \n Icono Exclamation /
Warniing",
                "Ejemplo Mensaje",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Exclamation,
                MessageBoxDefaultButton.Button2);

            if (Resultado == DialogResult.Yes)
            {
                MessageBox.Show("Seleccionaste Si");
            }
            else
            {
                MessageBox.Show("Seleccionaste No");
                return; //Con este comando, dejamos lo que estuvieramos haciendo sin efecto
            }
        }
```



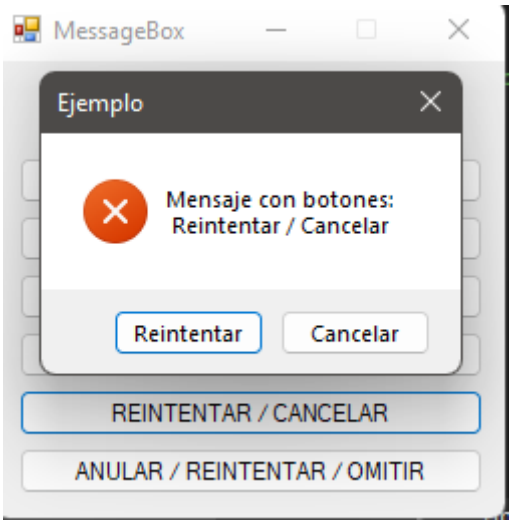
Si – No – Cancelar

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void btnSiNoCancelar_Click(object sender, EventArgs e)
        {
            Resultado = MessageBox.Show("Mensaje con botones Sí, No y Cancelar \n Icono
Question",
                "Ejemplo Mensaje YesNoCanccel",
                MessageBoxButtons.YesNoCancel,
                MessageBoxIcon.Question);

            switch (Resultado)
            {
                case DialogResult.Yes:
                    MessageBox.Show("Seleccionaste Si");
                    break;

                case DialogResult.No:
                    MessageBox.Show("Seleccionaste No");
                    return; //Con este comando, dejamos lo que estuviéramos haciendo sin
efecto

                case DialogResult.Cancel:
                    MessageBox.Show("Seleccionaste Cancelar");
                    return;//Con este comando, dejamos lo que estuviéramos haciendo sin
efecto

                default:
                    break;
            }
        }
```



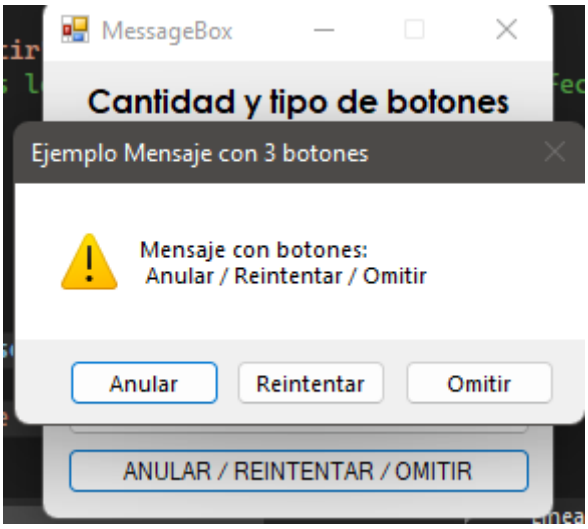
Reintentar – Cancelar

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void btnReintentarCancelar_Click(object sender, EventArgs e)
        {
            Resultado = MessageBox.Show("Mensaje con botones: \n Reintentar / Cancelar",
                "Ejemplo",
                MessageBoxButtons.RetryCancel,
                MessageBoxIcon.Stop);

            if (Resultado == DialogResult.Retry)
            {
                MessageBox.Show("Seleccionaste Reintentar");
            }
            else
            {
                MessageBox.Show("Seleccionaste Cancelar");
                return;
            }
        }
```



Anular – Reintentar – Omitir

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void btnAnularReintentarOmitir_Click(object sender, EventArgs e)
        {
            Resultado = MessageBox.Show("Mensaje con botones: \n Anular / Reintentar /
Omitir",
                "Ejemplo Mensaje con 3 botones",
                MessageBoxButtons.AbortRetryIgnore,
                MessageBoxIcon.Exclamation);

            switch (Resultado)
            {
               case DialogResult.Abort:
                    MessageBox.Show("Seleccionaste Anular");
                    break;

                case DialogResult.Retry:
                    MessageBox.Show("Seleccionaste Reintentar");
                    return; //Con este comando, dejamos lo que estuvieramos haciendo sin
efecto

                case DialogResult.Ignore:
                    MessageBox.Show("Seleccionaste Omitir");
                    return; //Con este comando, dejamos lo que estuvieramos haciendo sin
efecto

                default:
                    break;
            }
        }
```



Cerrar desde la barra de titulo

```
/* variable de ámbito de clase tipo DialogResult donde se guardarán
 * las respuestas de qué botón seleccionó el usuario en el cuadro de
 * mensajes (Ok, Cancel, Aceptar...), necesaria para codificar según
 * la respuesta.
 */
DialogResult Resultado;


private void frmMessageBox_FormClosing(object sender, FormClosingEventArgs e)
        {
            Resultado = MessageBox.Show("Esta seguro de CERRAR la aplicación?",
                    "SALIR DEL SISTEMA",
                    MessageBoxButtons.OKCancel,
                    MessageBoxIcon.Asterisk,
                    MessageBoxDefaultButton.Button2);

            if (Resultado == DialogResult.Cancel)
            {
                e.Cancel = true; //Cancela el cierre del formulario
            }
        }
```