

Tema 7: Controles II

- ☐ La clase ListBox.
- ☐ La clase CheckedListBox.
- ☐ La clase ComboBox.
- ☐ Menus.
- ☐ Formularios MDI.

La clase ListBox

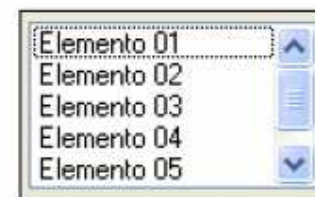
- ❑ Muestra una serie de elementos de los que el usuario puede seleccionar uno o más.
- ❑ Los elementos incluidos en el control se guardan en la propiedad `Items`.
 - `Items` es una colección del tipo `ListBox.ObjectCollection` que puede incluir cualquier tipo de objeto utilizado en .NET.
- ❑ Los elementos seleccionados se guardan en la propiedad `SelectedItems`, una colección del tipo `ListBox.SelectedObjectCollection`.
- ❑ Los índices de los elementos seleccionados se guardan en la propiedad `SelectedIndices`, una colección del tipo `ListBox.SelectedIndexCollection`

La clase ListBox (II)

❑ La colección Items.

- Representa a los objetos incluidos en la lista.
- Agregar elementos a la colección.
 - ✓ Se pueden agregar en tiempo de diseño mediante el editor de la propiedad.
 - En tiempo de diseño sólo es posible añadir cadenas.
 - ✓ En tiempo de ejecución se pueden agregar mediante el método Add.

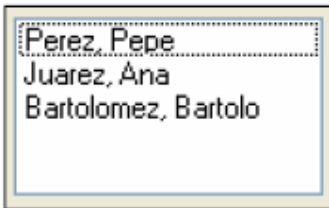
*Objeto*ListBox.Items.Add(*objeto*)



```
ListBox1.Items.Add("Elemento 01")
ListBox1.Items.Add("Elemento 02")
ListBox1.Items.Add("Elemento 03")
ListBox1.Items.Add("Elemento 04")
ListBox1.Items.Add("Elemento 05")
ListBox1.Items.Add("Elemento 06")
```

La clase ListBox (III)

- ❑ La colección Items.
 - Agregar elementos a la colección (continuación).
 - ✓ Mediante el método Add es posible añadir cualquier tipo de objetos.



```
Structure persona
    Dim id As Integer
    Dim nombre As String
    Dim apellidos As String
    Sub New(ByVal id As Integer, ByVal ape As String, ByVal nom As String)
        Me.id = id
        nombre = nom
        apellidos = ape
    End Sub

    'El método toString permite convertir un objeto en una cadena
    Overrides Function toString() As String
        Return apellidos & ", " & nombre
    End Function
End Structure

...
lstPersonas.Items.Add(New persona(123, "Perez", "Pepe"))
lstPersonas.Items.Add(New persona(323, "Juarez", "Ana"))
lstPersonas.Items.Add(New persona(333, "Bartolomez", "Bartolo"))
```

La clase ListBox (IV)

❑ La colección Items.

- Agregar elementos a la colección (continuación).

- ✓ El método Insert permite añadir un elemento en una posición específica \geq que 0 y \leq que el número de elementos.

*Objeto*ListBox.Items.Insert(*índice*,*objeto*)
ListBox1.Items.Insert(3,"Nuevo elemento")

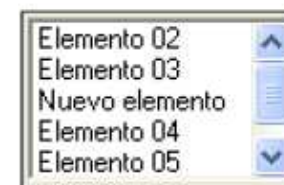


- ✓ Se puede insertar cualquier objeto de cualquier tipo en la lista,
lstPersonas.Items.Insert(0, New Persona(456, "Estévez", "Esteban"))

- Eliminar elementos de la colección.

- ✓ El método Remove permite eliminar un elemento de la colección a partir de su valor.

*Objeto*ListBox.Items.Remove(*objeto*)
ListBox1.Items.Remove("Elemento 01")



- ✓ El método RemoveAt permite eliminar un elemento de la colección a partir de su índice.

*Objeto*ListBox.Items.Remove(*índice*)
ListBox1.Items.RemoveAt(1)

- ✓ El método Clear permite eliminar todos los elementos.



La clase ListBox (V)

❑ Eliminar objetos de la lista...

- Se puede pasar como argumento una referencia a un objeto de la lista.

```
'p es una referencia a un elemento de la lista  
'No valdría si Dim p as persona = new Persona(323, "Juarez", "Ana")  
Dim p As Persona = lstPersonas.Items(1)  
lstPersonas.Items.Remove(p) 'Elimina el segundo elemento de la lista
```

- Si queremos buscar y borrar un elemento concreto habrá que realizar una búsqueda.

```
Dim p As persona = New persona(323, "Juarez", "Ana")  
'Elimina el objeto p (Ana Juarez) de la lista a partir de la búsqueda de su identificador  
'Es necesario codificar la función Equal en la estructura Persona  
For Each obj As persona In lstPersonas.Items  
    If obj.Equals(p) Then  
        lstPersonas.Items.Remove(obj)  
    Exit For  
End If  
Next  
...  
'En la estructura Persona  
Overloads Function Equals (ByVal o As persona) As Boolean  
    Return o.id = id  
End Function
```

La clase ListBox (VI)

❑ La colección `Items`.

- La propiedad `Count` devuelve el número de elementos de la colección.
- Buscar elementos en la colección.
 - ✓ La propiedad `Contains` devuelve un valor lógico `True` si el elemento que se pasa como argumento está incluido en la colección.
`ObjetoListBox.Items.Contains(objeto)`
 - ✓ La propiedad `IndexOf` devuelve el índice del objeto que se pasa como argumento. Devuelve `-1` si el objeto no se encuentra.
`ObjetoListBox.Items.IndexOf(objeto)`
 - ✓ En el código anterior se podría haber puesto...
`IstPersonas.Items.RemoveAt(IstPersonas.Items.IndexOf(obj))`

La clase ListBox (VII)

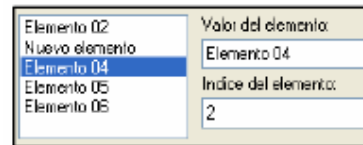
☐ Trabajar con elementos de la lista.

- La propiedad `SelectedIndex` devuelve el índice del elemento seleccionado de la lista.
 - ✓ Devuelve -1 si no se ha seleccionado ninguno.
- La propiedad `SelectedItem` devuelve el elemento seleccionado de la lista.
 - ✓ Devuelve el literal `Nothing` si no se ha seleccionado ninguno.
- La propiedad `Text` devuelve el contenido del elemento seleccionado convertido a cadena.

☐ Eventos.

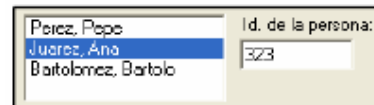
- Eventos `Click` y `DoubleClick`.
- Evento `SelectedItemChanged`.
 - ✓ Se produce cuando cambia el valor de la propiedad `SelectedItem`.
- Evento `SelectedIndexChanged`.
 - ✓ Se produce cuando cambia el valor de la propiedad `SelectedIndex`.

La clase ListBox (VIII)



Al seleccionar un elemento,
aparece su contenido y su posición

```
Private Sub ListBox1_SelectedIndexChanged (ByVal sender As System.Object,  
                                         ByVal e As System.EventArgs) _  
                                         Handles ListBox1.SelectedIndexChanged  
    TextBox1.Text = (ListBox1.SelectedItem)  
    TextBox2.Text = (ListBox1.SelectedIndex)  
End Sub
```



Al seleccionar una persona, aparece su
identificador

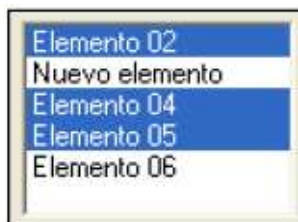
```
IstPersonas System Object  
Private Sub IstPersonas_SelectedIndexChanged (ByVal sender As System.Object, _  
                                             ByVal e As System.EventArgs) _  
                                             Handles IstPersonas.SelectedIndexChanged  
    Dim p As persona = IstPersonas.SelectedItem  
    TextBox3.Text = p.id  
End Sub
```

La clase ListBox (XI)

☐ Selección múltiple.

- La propiedad `SelectionMode` permite seleccionar varios elementos al mismo tiempo. Puede tomar alguno de los siguientes valores:
 - ✓ `None`. No se puede seleccionar ningún elemento.
 - ✓ `One`. Sólo es posible seleccionar un valor (valor predeterminado).
 - ✓ `MultiSimple`. Permite seleccionar varios elementos.
 - La selección se realiza marcando cada elemento con el ratón o la barra espaciadora.
 - ✓ `MultiExtended`. Permite seleccionar varios elementos.
 - La selección se puede realizar marcando cada elemento y utilizando las teclas `CTRL`, `SHIFT` o las teclas del cursor.
- La colección `SelectedItems` guarda los objetos seleccionados.
- La colección `SelectedIndices` guarda los índices de los elementos seleccionados.
 - El método `GetSelected(indice)` permite saber si un elemento ha sido seleccionado.
 - El método `SetSelected(indice,valor)` permite modificar el estado de un elemento determinado.

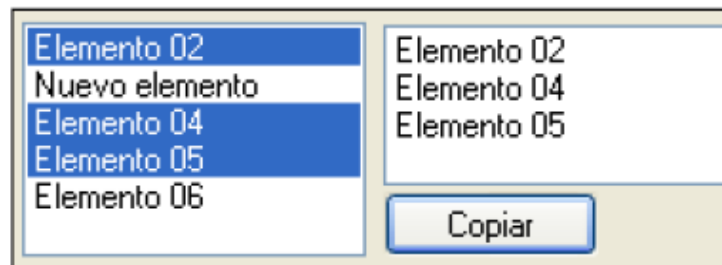
La clase ListBox (X)



Colección Items		
Índice	Objeto	Estado de la selección
0	Elemento 02	Seleccionado
1	Nuevo elemento	No seleccionado
2	Elemento 04	Seleccionado
3	Elemento 05	Seleccionado
4	Elemento 06	No seleccionado
Colección SelectedItems		
Índice	Objeto	
0	Elemento 02	
1	Elemento 04	
2	Elemento 05	
Colección SelectedIndices		
Índice	Objeto	
0	0	
1	2	
2	3	

La clase ListBox (XI)

- ❑ Ejemplo: copiar los elementos seleccionados de un ListBox a otro al pulsar el botón Copiar:



```
Private Sub Button1_Click (ByVal sender As System.Object, _  
                           ByVal e As System.EventArgs) _  
    Handles Button1.Click  
    For Each elem As Object In ListBox1.SelectedItems  
        ListBox2.Items.Add(elem)  
    Next  
End Sub
```

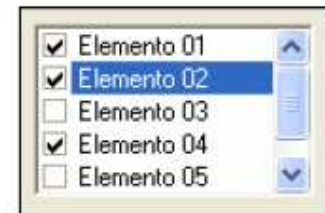
La clase ListBox (XII)

❑ Otras propiedades.

- Propiedad Sorted. Un valor True permite ordenar los elementos.
 - ✓ Cuando la propiedad está a True, el método Add e Insert añaden los elementos ordenados.
- Barras de desplazamiento.
 - ✓ La propiedad ScrollAlwaysVisible determina si se verá siempre la barra de desplazamiento.
 - ✓ La propiedad HorizontalScrollbar permite visualizar una barra de desplazamiento horizontal.
- Propiedad IntegralHeight.
 - ✓ Indica si la altura de la lista sólo puede visualizar elementos completos.
 - Un valor a True (predeterminado) impide que se visualicen elementos parcialmente.

Clase `CheckedListBox`

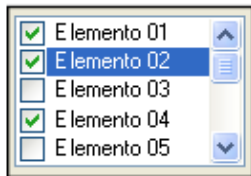
- ❑ Hereda de `ListBox` y utiliza sus mismos miembros.
 - Permite visualizar un cuadro de lista con casillas de verificación a la izquierda de sus elementos.
- ❑ No permite la selección de varios objetos, aunque si permite marcar las casillas de varios de ellos.
- ❑ Las colecciones `SelectedItems` y `SelectedIndices` se sustituyen por `CheckedItems` y `CheckedIndices`.
- ❑ El evento `ItemCheck` se produce cuando cambia el estado de alguno de sus elementos.
 - Utiliza un argumento del tipo `System.Windows.Forms.ItemCheckEventArgs` con los siguientes miembros:
 - ✓ `Index`. Índice del elemento que va a cambiar.
 - ✓ `CurrentValue`. Estado actual del elemento (`Checked`, `Unchecked`, `Indeterminate`).
 - ✓ `NewValue`. Nuevo estado del elemento.



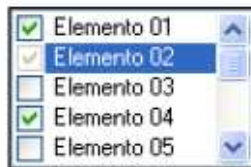
Clase **CheckedListBox** (II)

- ❑ **Método** `GetItemChecked(índice)`.
 - Devuelve `True` si el elemento está activado (estado `Checked` o `Indeterminate`) o `False` en caso contrario.
- ❑ **Método** `SetItemChecked(índice, estado)`.
 - Permite establecer el elemento a los estados `Checked` o `Unchecked`.
- ❑ **Método** `GetItemCheckState(índice)`.
 - Permite obtener el estado del elemento.
 - ✓ Devuelve `CheckedState.Checked`, `CheckedState.Unchecked` o `CheckedState.Indeterminate`.
- ❑ **Método** `SetItemCheckState(índice, estado)`.
 - Permite establecer el estado del elemento a `CheckedState.Checked`, `CheckedState.Unchecked` o `CheckedState.Indeterminate`.

Clase CheckedListBox (III)



```
Debug.WriteLine(CheckedListBox1.GetItemChecked(0))    'Escribe True
Debug.WriteLine(CheckedListBox1.GetItemChecked(2))    'Escribe False
```



```
CheckedListBox1.SetItemCheckState(1, CheckState.Indeterminate)
Debug.WriteLine(CheckedListBox1.GetItemCheckState(0)) 'Escribe Checked
Debug.WriteLine(CheckedListBox1.GetItemCheckState(1)) 'Escribe Indeterminate
Debug.WriteLine(CheckedListBox1.GetItemCheckState(2)) 'Escribe Unchecked
```

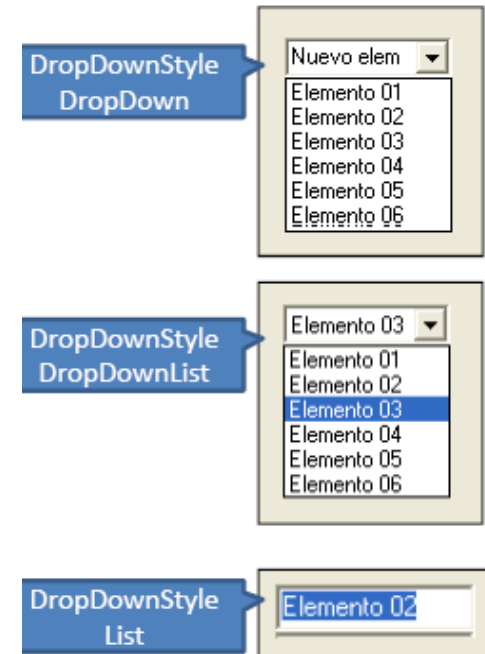

Clase ComboBox

❑ Combina un cuadro de lista con un cuadro de texto.

- Presenta la mayoría de las propiedades, métodos y eventos de ambos controles.
 - ✓ No permite multiselección.
 - ✓ No captura el evento DoubleClick.
 - ✓ La propiedad SelectedIndex también vale -1 si el usuario está editando el texto.

❑ Propiedad DropDownStyle.

- DropDown. Un cuadro de lista desplegable en el que usuario puede el editar el texto.
- DropDownList. Un cuadro de lista desplegable en el que el usuario no puede editar texto. Se puede acceder a los elementos a partir de la inicial.
- Simple. Una lista no desplegable en la que sólo se ve el elemento seleccionado o el que edita el usuario.



Clase ComboBox (II)

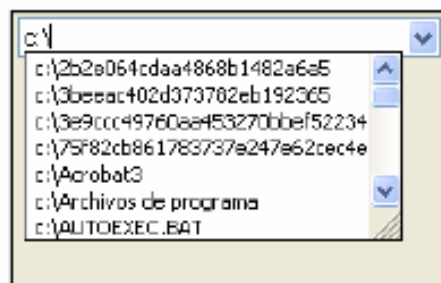
- ❑ Autocompletar el contenido de un ComboBox.
 - La propiedad AutoCompleteMode permite indicar si queremos que se autocomplete el contenido de lo escrito en un ComboBox:
 - ✓ None, no se autocomplete.
 - ✓ Append, al teclear los primeros caracteres añaden los que faltan.
 - ✓ Suggest, despliega una lista con las posibles opciones a completar.
 - ✓ SuggestAppend, añade los caracteres que faltan y despliega la lista.

Clase ComboBox (III)

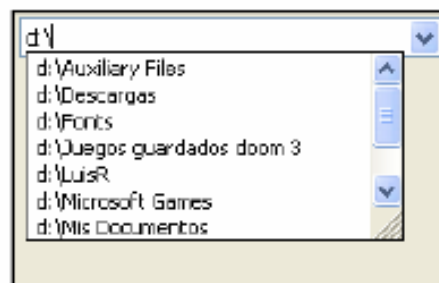
- La propiedad `AutoCompleteSource`, indica el origen de los datos a autocompletar.
 - ✓ `FileSystem` Especifica el sistema de archivos como origen.
 - ✓ `HistoryList` Incluye los URL en la lista de historial.
 - ✓ `RecentlyUsedList` Incluye los URL de la lista de las direcciones usadas recientemente.
 - ✓ `AllUrl` Especifica el equivalente de `HistoryList` y `RecentlyUsedList` como el origen.
 - ✓ `AllSystemSources` Especifica el equivalente de `FileSystem` y `AllUrl` como el origen.
 - ✓ `FileSystemDirectories` Especifica que sólo los nombres de directorio y no los nombres de archivo se finalizarán automáticamente.
 - ✓ `ListItems`. Especifica que los elementos de la lista son el origen.
 - ✓ `CustomSource` Especifica que se utilizarán las cadenas que formen la propiedad `AutoCompleteCustomSource`

Clase ComboBox (IV)

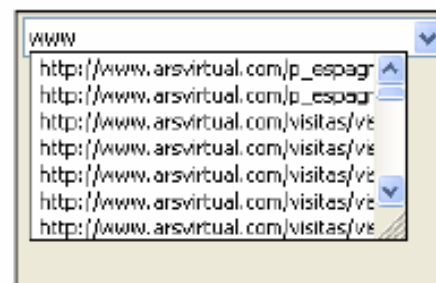
☐ AutoCompleteSource



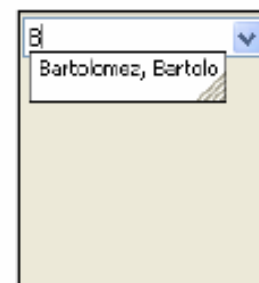
FileSystem



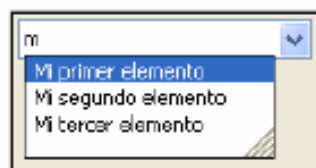
FileSystemDirectories



HistoryList, AllUrl
y RecentlyUsedList



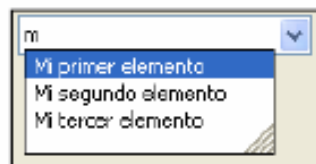
ListItems



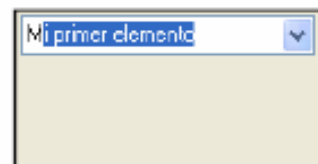
CustomSource

Con AutoCompleteSource a CustomSource es necesario rellenar los elementos de la colección personalizada en la propiedad AutoCompleteCustomSource

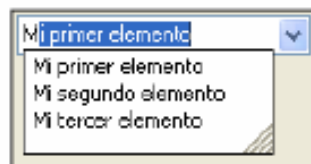
☐ AutoCompleteMode.



Suggest



Append



SuggestAppend

Menús

❑ Los menús se construyen a partir de la clase MenuStrip.

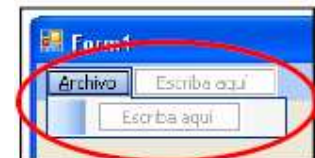
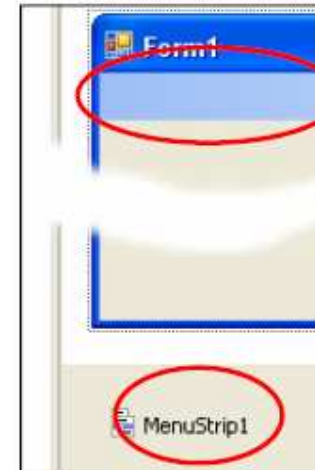
- MenuStrip representa un contenedor de la estructura de menús de un formulario.

❑ Crear un menú.

- Al arrastrar un menú en el formulario, aparecerá en la bandeja de componentes y el área donde aparecerá el menú acoplada en la parte superior del formulario.

❑ Elementos del menú.

- Son objetos de la clase ToolStripMenuItem.
- Al seleccionar la barra de menús o el control MenuStrip en la bandeja de componentes, el entorno permitirá escribir el título del menú.
- A medida que se dan nombres a los elementos ToolStripMenuItem del menú, aparecen posiciones para un nuevo elemento de menú del mismo nivel o un menú desplegable



Menús (II)

❑ Títulos de los menús.

- El carácter ampersand (&) hace que el carácter siguiente se convierta en la tecla de acceso rápido.
 - ✓ Las recomendaciones de diseño de la interfaz indican que **todos** los elementos de un menú deben tener tecla de acceso rápido.

❑ Nombres de los objetos MenuStrip y ToolStripMenuItem.

- Puesto que un formulario sólo tendrá normalmente un elemento MenuStrip la mayoría de las veces no será necesario dar un nombre distinto.
- En los elementos ToolStripMenuItem Visual Studio pone por omisión un nombre formado por el título y el sufijo ToolStripMenuItem (por ejemplo, ArchivoToolStripMenuItem).
 - ✓ Para los submenús, se recomienda utilizar para el nombre, el título del menú de jerarquía superior y el nombre del actual.
 - Por ejemplo una opción Nuevo dentro del menú Archivo podría tener el nombre ArchivoNuevoToolStripMenuItem.

Menús (III)

❑ Tipos de elementos de menú.

- Por omisión el aspecto del elemento de menú es una etiqueta con texto estático.
- Es posible cambiar ese aspecto para mostrar un ComboBox o un TextBox.
 - ✓ Al pulsar con el botón secundario en un elemento de menú, seleccionar la opción “Convertir en”.
 - MenuItem. El aspecto por omisión.
 - ComboBox. Aparece una lista desplegable. La propiedad Items del elemento de menú permite añadir elementos.
 - Se pueden añadir elementos al ComboBox de forma dinámica con el método Add de la propiedad Item del objeto ToolStripMenuItem.
 - Se puede acceder al texto seleccionado por la propiedad Text del objeto.
 - TextBox. Aparece como un cuadro de texto editable.
 - Se puede acceder al texto mediante la propiedad Text del objeto ToolStripMenuItem.
 - Los elementos de tipo ComboBox y TextBox **no pueden** tener submenús.

Menús (IV)

❑ Aspecto del menú.

- Para agregar un separador entre dos elementos de menú, en el menú contextual del elemento, seleccionar la opción “Insertar” y en el submenú “Separator”.
- Marcas de verificación.
 - ✓ La propiedad Checked permite añadir una marca de verificación al menú.
 - ✓ En tiempo de ejecución, mediante código, es posible modificar la marca mediante la propiedad CheckState.
 - Puede tomar los valores Checked, Unchecked o Indeterminate.
 - ✓ Las propiedades Checked y CheckState también permiten obtener el estado de verificación.
 - ✓ La propiedad CheckOnClick, permiten modificar el estado de la verificación al hacer clic.

Menús (V)

☐ Aspecto del menú (continuación).

- Imágenes.

- ✓ Si se trata de un elemento de menú de tipo MenuItem es posible añadir una imagen al margen con la propiedad Image.
- ✓ Si la casilla de verificación está activada, aparecerá un recuadro rodeando la imagen.

- Habilitar y deshabilitar elementos de un menú.

- ✓ La propiedad Enabled, permite deshabilitar las opciones no disponibles en un momento dado.
 - No debería ser posible acceder a aquellas opciones no disponibles: la interfaz debe mostrar pistas visuales.
 - También es posible que no se muestren las opciones mediante la propiedad Visible.

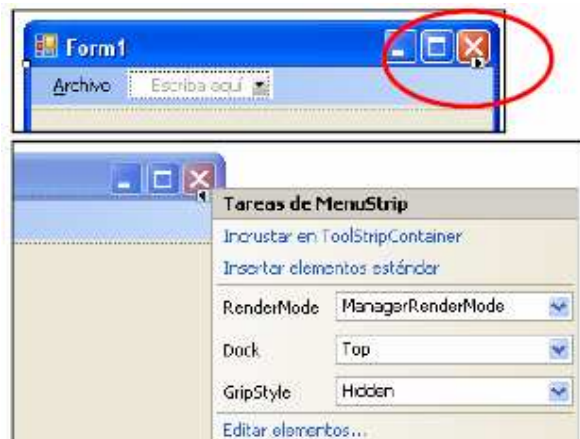
Menús (VI)

❑ Teclas de método abreviado.

- La propiedad ShortcutKeys permite asociar una tecla de método abreviado al elemento de menú.
 - ✓ Sólo deben tener teclas de método abreviado las opciones finales de menú.
 - ✓ Si la propiedad ShowShortcutKeys está a True, aparecerá la combinación de teclas a la derecha.

❑ Añadir opciones estándar de menú.

- En el glifo () de etiqueta inteligente del control MenuStrip y seleccionar “Insertar elementos estándar”.
- Se añaden los elementos estándar de un menú Windows.



Menús (VII)

☐ Controlar los eventos.

- Para asociar una acción a cualquier elemento de un menú se utilizará el evento Click.

```
Private Sub ArchivoNuevoToolStripMenuItem_Click (ByVal sender As System.Object, _  
                                                    ByVal e As System.EventArgs) _  
                                                    Handles NuevoToolStripMenuItem.Click  
    'Introducir el código correspondiente a la opción Nuevo del menú Archivo  
End Sub
```

- Si se utilizan elementos de menú con casillas de verificación, los eventos `CheckedChanged` y `CheckStateChanged` permite verificar si se ha modificado su estado.
 - ✓ Funcionan de la misma forma que sus equivalentes de la clase `CheckBox`.

Menús (VIII)

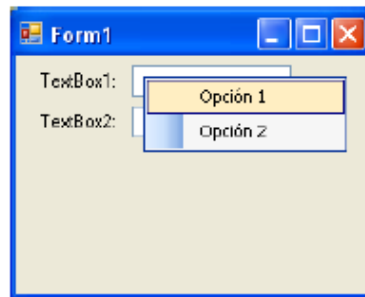
❑ Menús emergentes (menús contextuales o Popup).

- Se activan al hacer clic con el botón secundario en un control.
- El contenedor será en este caso un objeto de la clase ContextMenuStrip.
 - ✓ También hay que arrastrarlo a la bandeja de componentes.
 - ✓ Aunque en un formulario normalmente sólo hay un menú principal (objeto de la clase MenuStrip), puede haber tantos menús emergentes cómo se desee.
 - ✓ El objeto ContextMenuStrip contendrá los elementos de menú (ToolStripMenuItem).
 - ✓ Para asociar el menú emergente a un control o formulario, será necesario indicarlo en la propiedad ContextMenuStrip del control o formulario.

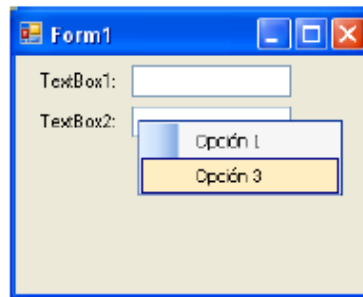
Menús (IX)

❑ El evento `Opening` se produce antes de que se abra el menú contextual.

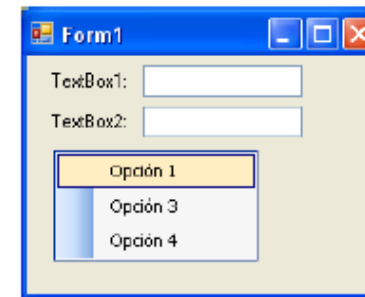
- Se puede utilizar para añadir distintas opciones a un menú contextual dependiendo del control que se ha abierto.
 - ✓ La propiedad `SourceControl` de la clase `ContextMenuStrip`, permite guardar una referencia al objeto sobre el que se abrió el menú emergente.
- En el ejemplo, el mismo objeto `ContextMenuStrip` muestra tres menús emergentes distintos, aunque con opciones compartidas.



El menú emergente del control `TextBox1` muestra los elementos `Opción 1` y `Opción 2`



El menú emergente del control `TextBox2` muestra los elementos `Opción 1` y `Opción 3`



El menú emergente del formulario muestra los elementos `Opción 1`, `Opción 3` y `Opción 4`

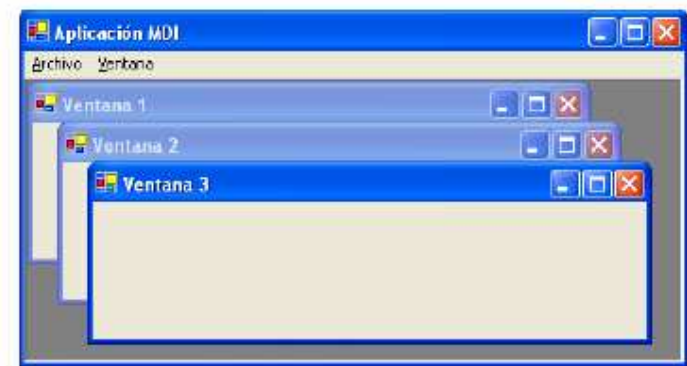
Menús (X)

'Se supone que TextBox1, Textbox2 y Form1 tienen la propiedad ContextMenuStrip a ContextMenuStrip1.
'También existen los elementos de menú Opción1, Opción2, Opción3 y Opción4

```
Private Sub ContextMenuStrip1_Opening (ByVal sender As System.Object, _  
                                     ByVal e As System.ComponentModel.CancelEventArgs) _  
                                     Handles ContextMenuStrip1.Opening  
    If ContextMenuStrip1.SourceControl Is TextBox1 Then  
        'Limpia el contenido anterior del menú  
        ContextMenuStrip1.Items.Clear()  
        ContextMenuStrip1.Items.Add(Opción1ToolStripMenuItem)  
        ContextMenuStrip1.Items.Add(Opción2ToolStripMenuItem)  
    ElseIf ContextMenuStrip1.SourceControl Is TextBox2 Then  
        ContextMenuStrip1.Items.Clear()  
        ContextMenuStrip1.Items.Add(Opción1ToolStripMenuItem)  
        ContextMenuStrip1.Items.Add(Opción3ToolStripMenuItem)  
    ElseIf ContextMenuStrip1.SourceControl Is Me Then  
        ContextMenuStrip1.Items.Clear()  
        ContextMenuStrip1.Items.Add(Opción1ToolStripMenuItem)  
        ContextMenuStrip1.Items.Add(Opción3ToolStripMenuItem)  
        ContextMenuStrip1.Items.Add(Opción4ToolStripMenuItem)  
    End If  
End Sub
```

Formularios MDI

- ☐ Aplicaciones SDI (Single Document Interface).
 - La aplicación sólo permite tener abierta una única ventana al mismo tiempo (por ejemplo la aplicación WordPad).
- ☐ Aplicaciones MDI (Multiple Document Interface).
 - Existe una ventana MDI primaria (padre) que actúa como contenedor de ventanas MDI secundarias (hijas).
 - ✓ Es útil cuando una aplicación requiere de varias ventanas de características generales o para navegar entre las distintas ventanas de una aplicación.
 - En algunas aplicaciones actuales (cómo Office 2007) se sigue un modelo similar al MDI:
 - ✓ La aplicación mantiene varias ventanas de documento abiertas, aunque no existe una ventana primaria contenedora.



Formularios MDI (II)

☐ Formulario MDI primario.

- Un objeto de la clase Form con la propiedad IsMdiContainer a True.

☐ Formulario MDI secundario.

- Un objeto de la clase Form cuya propiedad MdiParent apunta al formulario MDI primario.

☐ Abrir una ventana MDI secundaria.

```
Private Sub AbrirFormularioHijo()  
    Dim frm As New FormularioHijo  
    frm.MdiParent = Me  
    Static Dim numHijos As Integer  
    numHijos += 1  
    frm.Text = "Ventana " & numHijos  
    frm.Show()  
End Sub
```

'La clase FormularioHijo ya está creada
'MdiParent apunta al formulario actual
'Esta variable sirve para el título
'del formulario hijo. Se incrementa en 1
'Nuevo título de la ventana
'Por último se muestra el formulario

Formularios MDI (III)

❑ Acceso a los formularios hijo.

- Los formularios primarios guardan en la propiedad MdiChildren la colección de formularios secundarios.

```
'Da color rojo a todos los formularios secundarios desde un formulario hijo
For Each frm As Form In My.Forms.frmAplicaciónMDI.MdiChildren
    frm.BackColor = Color.Red
Next
```

- La propiedad ActiveMdiChild de la clase Form devuelve una referencia al formulario hijo activo o Nothing si no existe ningún formulario hijo activo.
- El método ActivateMdiChild(referenickFormularioHijo) activa una ventana secundaria concreta.
- El evento MdiChildActivate se desencadena cuando se abre un formulario secundario.

```
'Cambia el título del formulario primario cada vez que cambia el secundario
Private Sub Form1_MdiChildActivate( ByVal sender As Object, _
                                   ByVal e As System.EventArgs) Handles MyBase.MdiChildActivate
    If Me.ActiveMdiChild Is Nothing Then
        Me.Text = "Aplicación MDI"
    Else
        Me "Aplicación - " & Me.ActiveMdiChild Me.Text = MDI Me.ActiveMdiChild.Text
    End If
End Sub
```

Formularios MDI (IV)

☐ El menú Ventana.

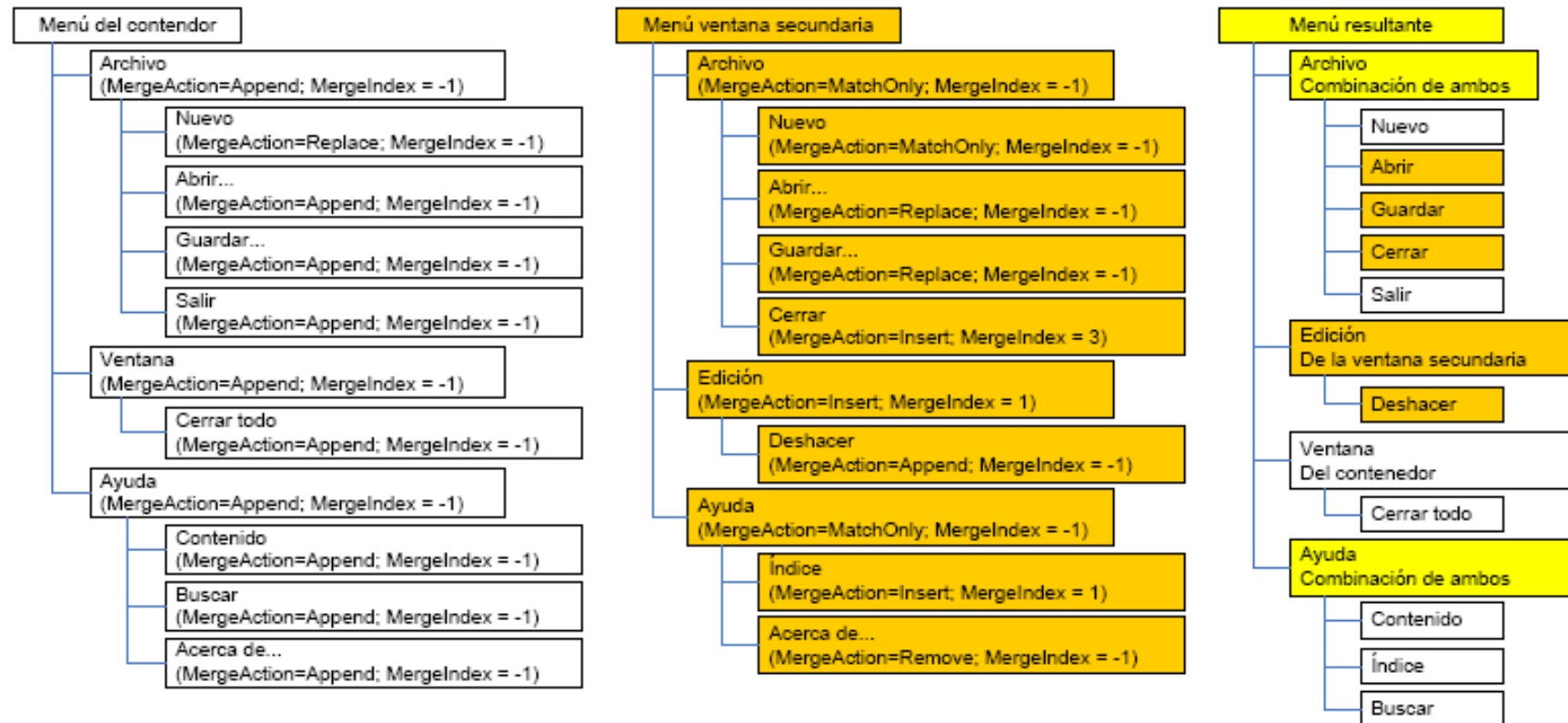
- En ocasiones las aplicaciones MDI presentan una opción de menú para administrar los formularios hijo.
- Para que en ese menú aparezca una lista con todas las ventanas secundarias, es necesario indicar al objeto MenuStrip de la ventana padre en que elemento aparecerá la lista de ventanas mediante la propiedad MdiWindowsListItem.
 - ✓ Normalmente se tratará de un menú de primer nivel (p.e. el menú Ventana).
- Para organizar las ventanas secundarias se utiliza el método LayoutMdi de la clase Form.
 - ✓ Organización en cascada.
 - Me.LayoutMdi(MdiLayout.Cascade)
 - ✓ Organización en mosaico horizontal.
 - Me.LayoutMdi(MdiLayout.TileHorizontal)
 - ✓ Organización en mosaico vertical.
 - Me.LayoutMdi(MdiLayout.TileVertical)
 - ✓ Organiza los iconos de las ventanas en el caso de que estén minimizadas.
 - Me.LayoutMdi(MdiLayout.ArrangeIcons)

Formularios MDI (V)

❑ Combinación de menús.

- Los menús del formulario secundario activo se combinarán con el del formulario contenedor.
 - ✓ La propiedad AllowMerge de la clase Form posibilita o impide la combinación de menús.
 - ✓ La propiedad MergeAction especifica el tipo de combinación que se utilizará.
 - Su valor es un miembro del enumerado MenuAction:
 - Append. Los elemento del formulario secundario se añaden al final de los del formulario contenedor.
 - Insert. Inserta el elemento en el formulario contendor en la posición indicada por la propiedad MergeIndex.
 - Replace. Reemplaza el elemento coincidente (el que tenga el mismo texto en la etiqueta).
 - Remove. Elimina los elementos que tengan el mismo nombre.
 - MatchOnly. Realiza alguna acción si los elementos coinciden. La acción a realizar dependerá de los elementos de los submenús o del valor MergeAction del otro elemento coincidente.

Formularios MDI (VI)



Formularios MDI (VII)

❑ Aplicación de ejemplo.

- Sólo será capaz de crear nuevas ventanas, cerrarlas y organizarlas.
- El formulario principal sólo tendrá el menú archivo con dos opciones de menú:
 - ✓ Cada vez que se da la opción Nuevo del menú Archivo se abrirá una nueva ventana con el título “Ventana xxx”.
 - ✓ La opción Salir del menú Archivo terminará la aplicación.
- La ventana secundaria tendrá los menús Archivo y Ventana.
 - ✓ El menú archivo se combinará con el de la ventana principal y añadirá la opción Cerrar que cerrará la ventana.
 - ✓ El menú ventana tendrá como opciones:
 - Cerrar todo. Cierra todas las ventanas.
 - Una lista de las ventanas abiertas.
 - Una opción Organizar con submenús para organizar las ventanas en cascada, en mosaico horizontal, mosaico vertical u organizar iconos.



Formularios MDI (VIII)

'Código del formulario principal

'Este evento se produce al intentar cerrar el formulario principal

```
Private Sub frmAplicaciónMDI FormClosing (frmAplicaciónMDI_ByVal sender As Object, _  
                                           ByVal e As System.Windows.Forms.FormClosingEventArgs) _  
                                           Handles Me.FormClosing  
  
    'Si existen hijos abiertos  
    If Me.MdiChildren.Length <> 0 Then  
        'Se pregunta si se desea seguir cerrando  
  
        If MessageBox.Show ("Todavía quedan ventanas abiertas ¿Desea continuar", _  
                             "Aplicación MDI", MessageBoxButtons.YesNo, _  
                             MessageBoxIcon.Exclamation, _  
                             MessageBoxDefaultButton.Button2) = Windows.Forms.DialogResult.No Then  
  
            'Si no se quiere cerrar la aplicación la propiedad Cancel del evento se pone a True,  
            'por lo que se cancela la operación de cierre  
            e.Cancel = True  
  
        End If  
    End If  
End Sub
```

Formularios MDI (IX)

```
'Cuando se activa un formulario MDI hijo
Private Sub frmAplicaciónMDI_MdiChildActivate (ByVal sender As Object, ByVal e As System.EventArgs)
    Handles Me.MdiChildActivate

    'Si no hay ningún formulario secundario
    If Me.ActiveMdiChild Is Nothing Then
        Me.Text = "Aplicación MDI"
        'De esta forma no se muestra el menú ventana cuando no hay ventanas hijas
        VentanaToolStripMenuItem.Visible = False
    Else
        'Si no se pone el título del formulario seguido del nombre de la ventana
        Me.Text = "Aplicación MDI - " & Me.ActiveMdiChild.Text
        'De esta forma se muestra el menú ventana cuando hay ventanas hijas
        VentanaToolStripMenuItem.Visible = True
    End If
End Sub

Private Sub NuevoToolStripMenuItem_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles NuevoToolStripMenuItem.Click

    Dim frm As New frmFormularioHijo
    frm.MdiParent = Me
    Static numHijos As Integer 'numHijos lleva un contador permanente de ventanas secundarias
    numHijos += 1
    frm.Text = "Ventana " & numHijos 'El título de la ventana incluye el numHijos
    frm.Show()
End Sub
```

Formularios MDI (X)

'Código del formulario secundario

```
Private Sub MosaicohorizontalToolStripMenuItem_Click (ByVal sender As System.Object, _  
                                                    ByVal e As System.EventArgs) _  
                                                    Handles MosaicohorizontalToolStripMenuItem.Click  
    My.Forms.frmAplicaciónMDI.LayoutMdi(MdiLayout.TileHorizontal)  
End Sub
```

```
Private Sub MosaicoverticalToolStripMenuItem_Click (ByVal sender As Object, ByVal e As System.EventArgs)_  
                                                    Handles MosaicoverticalToolStripMenuItem.Click  
    My.Forms.frmAplicaciónMDI.LayoutMdi(MdiLayout.TileVertical)  
End Sub
```

```
Private Sub CascadaToolStripMenuItem_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) _  
                                                    Handles CascadaToolStripMenuItem.Click  
    My.Forms.frmAplicaciónMDI.LayoutMdi(MdiLayout.Cascade)  
End Sub
```

```
Private Sub OrganizarIconosToolStripMenuItem_Click (ByVal sender As System.Object, _  
                                                    ByVal e As System.EventArgs) _  
                                                    Handles OrganizarIconosToolStripMenuItem.Click  
    My.Forms.frmAplicaciónMDI.LayoutMdi(MdiLayout.ArrangeIcons)  
End Sub
```


Formularios MDI (XI)

```
Private Sub CerrarTodoolStripMenuItem_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) _  
                                         Handles CerrarTodoToolStripMenuItem.Click  
    'Recorre todos los formularios de la ventana contenedora y los cierra  
    For Each frm As Form In My.Forms.frmAplicaciónMDI.MdiChildren  
        frm.Close()  
    Next  
End Sub  
  
Private Sub CerrarToolStripMenuItem_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) _  
                                         Handles CerrarToolStripMenuItem.Click  
    'Cierra la ventana actual (es decir el formulario hijo activo)  
    Me.Close()  
End Sub
```