Universidad Adventista de Centro América
Escuela de Ingeniería

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

Tema:

Controles, Eventos, Métodos y Propiedades de Visual Basic. Cristal Report

Trabajo Presentado a:
Lic. Jair del Valle

En cumplimiento parcial de la materia:
Lenguajes de IV Generación

38.749

Por:
Carlos A. Aquino López
Elmer Jonathan López

BIBLIOTECA UNADECA ALAJUELA, COSTA RICA

Alajuela, Costa Rica, 25 de octubre de 2000

INDICE

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

CONTROLES

CheckBox 02
ComboBox 04
CommandButton 06
Contenedor OLE 08
Control de Datos ADO 11
DataRepeater 13
DirListBox 16
DriveListBox 18
Extender 20
FileListBox 24
Form 26
HScrollBar, VScrollBar 30
ListBox 32
MSFlexGrid 35
MSHFlexGrid 39
Masked Edit 44
Option Button 47
PictureBox 50
Property Page 53
RichTextBox 55
TextBox 59
UserControl 62
UserDocument 65

EVENTOS

Activate, Deactivate	69
Change	70
Click	75
DbClick	79
Compare	83
DragDrop	84
DragOver	86
DragDown	89
EnterCell	90
ItemCheck	92
KeyDown, KeyUp	93
KeyPress	99
LinkClose	102
LinckError	103
LinckNotify	104
LinckOpen	105
LostFocus	106
MouseDown, MouseUp	108
MouseMove	111
ObjectEvent	114
ObjetctMove	115
OLECompleteDrag	116
OLEDragDrop	119
OLEDragOver	121
OLEGiveFeedback	124
OLESetData	126
OLEStarDrag	127
PathChange	128
PatternChange	129
Paint	131
Scroll	132
Resize	134
Updated	136
Validate	137

METODOS

AddItem 141
Circle 142
Clear 144
CLS 146
Drag 147
LinkExecute 151
LinkPoke 152
LinkRequest 154
LinkSend 156
Move 156
OLEDrag 158
PSet 159
PaintPicture 161
Point 163
PopupMenu 164
Refresh 166
RemoveItem 168
Scale 169
ScaleX, ScaleY 172
SetFocus 173
ShowWhatsThis 174
TextHeight 176
TextWidth 177
ZOrder 179

PROPIEDADES

Alignment	182
Align	183
Appearance	185
Archjive, Hidden, Normal, System	186
AutoRedraw	188
BackColor, ForeColor	191
BorderStyle	194
ClipControls	196
Columns	197
Container	199
CurrentX, CurrentY	200
DataChanged	201
DataField	203
DataFormat	204
DataMember	205
DataSource	210
DragIcon	211
DragMode	213
DrawMode	215
DrawStyle	217
DrawWidth	218
Drive	219
Enabled	221
FillColor	223
FileName	224
FileListBox	225
FillStyle	227
Font	228
FontName	231
FontSize	233
FontTransparent	234
Height, Width	235
HelpContextID	238
Image	240
Index	242
LargeChange, SmallChange	244

HSCrollBar, VSCrollBar 245
Left, Top 247
List 249
ListIndex 252
Locked 254
TextBox 255
LinkItem 258
LinkMode 259
LinkTimeOut 261
LinkTopic 262
Max, Min 265
MouseIcon 266
MousePointer 268
MultiSelect 270
Name 272
Negotiate 274
OLEDragMode 275
OLEDropMode 276
Parent 277
Path 279
Pattern 281
ReadOnly 282
Selected 283
Picture 284
RighToLeft 286
ScaleHeight, SacleWidth 287
ScaleLeft, ScaleTop 289
ScaleMode 291
TabIndex 292
TabStop 295
Tag 296
ToolTipText 298
TopIndex 299
Value 300
Visible 304
WhatsThisHelpID 306
hDC 307
hWnd 309

REFERENCIAS

Archivo Ejecutable 313
API de Windows 313
Aplicación 313
Comando Pegar Vinculo 313
Contexto de Dispositivo 313
Controlador 313
Cuadro de Lista 313
Destino 313
Evento 314
Evento en Cascada 314
Forculario Primario 314
Formulario MDI 314
Gráfico Persistente 314
Indicador 314
Indice 314
Información Sobre Herramientas 314
Instancia 315
Mapa de Bits 315
Mascara de Bits 315
Metarchivo 315
Objeto Vinculado 315
Objeto Incrustado 315
Origen 315
Orden de Tabulación 316
Pixel 316
Poke 316
Punto 316
Sintaxis 316
Submenú 316
Tecla de Acceso 316
Tecla de Edición 316
Tecla de Función 316
Twip 316
Ventana Activa 317
Ventana Secundaria MDI 317

CRYSTAL REPORTS

Crystal Reports Paso a Paso	319
Artículos, Trucos y comentarios sobre Crystal Reports 5.0	325
Generacion de Informes con Crystal Reports 8.0	326
Crystal Reports, Creación de Informes Por Impresora	328
Tablas de Referencia Cruzada	335
Modificación del Diseño del Informe	337
Herramientad de Diseño	338

Sac D'quino

INTRODUCCION

Hola, la siguiente información ha sido recopilada accesando algunas paginas de Internet, cuyas direcciones se publican al final de este pequeño manual de ayuda para el programador de Visual Basic.

El propósito fundamental de este manual, es poder explicar el funcionamiento y la utilidad que el programador de Visual Basic puede hacer con las opciones que este lenguaje de Programación le presenta.

Encontraras algunas palabras las cuales estarán subrayadas, esto quiere decir que puedes encontrar una mejor explicación en las secciones en las cuales se ha dividido este manual; estas secciones son CONTROLES, EVENTOS, METODOS, PROPIEDADES, REFERENCIAS y una pequeña recopilación del programa CRYSTAL REPORT.

Presentamos también un índice, para poder encontrar la información que necesitas con una mayor rapidez. Este índice lo hemos divido en las partes en que se divide el manual las cuales mencionamos en el párrafo anterior. Si tienes alguna sugerencia o pregunta, escríbenos a los mail que encontraras en la sección de direcciones.

Esperamos que pueda, serte de mucha ayuda este manual y si necesitas saber sobre otro tópico de la ayuda de Visual Basic, recuerda que al final están las direcciones de Internet, para que puedas accesarlas.

Atentamente.

Ing. Carlos A. Aquino L Ing. Elmer J. López D.

CONTROLES

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

CheckBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **CheckBox** muestra una X cuando está activado; la X desaparece cuando el control **CheckBox** se desactiva. Utilice este control para ofrecer al usuario una opción de tipo Verdadero o Falso o Sí o No. Puede usar controles **CheckBox** en grupos para mostrar múltiples opciones entre las cuales el usuario puede seleccionar una o más. También puede establecer el valor de **CheckBox** mediante programación con la propiedad **Value**.

Sintaxis
CheckBox

Comentarios
Los controles **CheckBox** y **OptionButton** funcionan de manera similar, pero con una diferencia importante: pueden seleccionarse varios controles **CheckBox** de un formulario al mismo tiempo. Por el contrario, sólo puede seleccionarse un control **OptionButton** en un grupo en un momento determinado.
Para mostrar el texto que aparece junto al **CheckBox**, establezca la propiedad **Caption**. Utilice la propiedad **Value** para determinar el estado del control: activado, desactivado o no disponible.

PROPIEDADES DE CheckBox (Control)

Alignment (Propiedad)
Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Caption (Propiedad)
Container (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DisabledPicture (Propiedad)
DownPicture (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)

Left, Top (Propiedades)
MaskColor (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Picture (Propiedad)
RightToLeft (Propiedad)
Style (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
UseMaskColor (Propiedad)
Value (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE CheckBox (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE CheckBox (Control)

Click (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)

OLESetData (Evento)
OLEStartDrag (Evento)
Validate (Evento)

ComboBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **ComboBox** combina las características de un control **TextBox** y un control **ListBox**; los usuarios pueden introducir información en la parte del cuadro de texto o seleccionar un elemento en la parte de cuadro de lista del control.

Sintaxis
ComboBox

Comentarios
Para agregar o eliminar elementos en un control **ComboBox**, use el método **AddItem** o **RemoveItem**. Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir a un usuario tener acceso a los elementos de un control **ComboBox**. Como alternativa, puede agregar elementos a la lista mediante la propiedad **List** en tiempo de diseño.

Nota Un evento Scroll ocurrirá en un control **ComboBox** sólo cuando se desplace el contenido de la parte desplegable del **ComboBox**, no cada vez que cambie el contenido del **ComboBox**. Por ejemplo, si la parte desplegable de un **ComboBox** contiene cinco elementos y el elemento superior está resaltado, no ocurrirá un evento Scroll hasta que presione seis veces la flecha hacia abajo (o una vez la tecla AV PÁG). Después de eso, ocurrirá un evento Scroll por cada pulsación de la tecla de flecha hacia abajo. Sin embargo, si después presiona la tecla de flecha hacia arriba, no ocurrirá un evento Scroll hasta que presione seis veces la tecla de flecha hacia arriba (o una vez la tecla RE PÁG). Después de eso, cada vez que presione la tecla de flecha hacia arriba se producirá un evento Scroll.

PROPIEDADES DE ComboBox (Control)

Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)

Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
IntegralHeight (Propiedad)
ItemData (Propiedad)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
Locked (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
NewIndex (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
RightToLeft (Propiedad)
SelLength, SelStart, SelText (Propiedades)
SelLength, SelStart, SelText (Propiedades, controles ActiveX)
Sorted (Propiedad)
Style (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
Text (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

BIBLIOTECA UNADECA ALAJUELA COSTA RICA

METODOS DE ComboBox (Control)

AddItem (Método)
Clear (Método para Clipboard, ComboBox y ListBox)
Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)

RemoveItem (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE ComboBox (Control)
Change (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
DropDown (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Scroll (Evento)
Validate (Evento)

CommandButton (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Utilice un control **CommandButton** para comenzar, interrumpir o terminar un proceso. Cuando está activado, un **CommandButton** aparece presionado y a veces se denomina botón de comando.

Sintaxis
CommandButton

Comentarios
Para mostrar texto en un control **CommandButton**, establezca su propiedad **Caption**. Un usuario siempre puede elegir un **CommandButton** si hace clic en él. Para permitir que el usuario lo elija presionando la tecla ENTRAR, establezca la propiedad **Default** a **True**. Para permitir que el usuario elija el botón presionando la tecla ESC, establezca la propiedad **Cancel** del **CommandButton** a **True**.

PROPIEDADES DE CommandButton (Control)

Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Cancel (Propiedad)
Caption (Propiedad)
Container (Propiedad)
Default (Propiedad)
DisabledPicture (Propiedad)
DownPicture (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
MaskColor (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Picture (Propiedad)
Style (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
UseMaskColor (Propiedad)
Value (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE CommandButton (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)

SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE CommandButton (Control)
Click (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Validate (Evento)

Contenedor OLE (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control contenedor **OLE** le permite agregar objetos insertables a los formularios de las aplicaciones de Visual Basic. Con el control contenedor **OLE** es posible:

- Crear un marcador de posición para un objeto insertable en la aplicación. En tiempo de ejecución puede crear el objeto que se presenta dentro del control contenedor **OLE** o cambiar un objeto que ha colocado dentro del control contenedor **OLE** en tiempo de diseño.

- Crear un objeto vinculado en la aplicación.

- Enlazar el control contenedor **OLE** con una base de datos mediante el control **Data**.

Puede crear el objeto en tiempo de diseño si usa el cuadro de diálogo **Insertar objeto** (que contiene los comandos **Insertar objeto, Pegado especial**, etc.) o en tiempo de ejecución si establece las propiedades adecuadas.

Al mover el control contenedor **OLE** en un formulario con el método **ObjectMove**, los valores de las propiedades **Height** y **Width** del objeto pueden ser ligeramente diferentes después del movimiento. Esto se debe a que los parámetros del método **ObjectMove** son píxeles convertidos al modo actual de la escala del formulario. El proceso de conversión de píxeles a twips y viceversa no produce siempre valores idénticos.

Usar los menús emergentes del control contenedor OLE

Cada vez que se dibuja un control contenedor **OLE** en un formulario aparece el cuadro de diálogo **Insertar objeto**. Use este cuadro de diálogo para crear un objeto vinculado o incrustado. Si hace clic en **Cancelar**, no se creará ningún objeto.
En tiempo de diseño, haga clic en el control contenedor **OLE** con el botón secundario del *mouse* (ratón) para presentar un menú emergente. Los comandos mostrados en dicho menú emergente dependen del estado del control contenedor **OLE** como se muestra en la siguiente tabla:

Comando	**Habilitado en el menú emergente cuando**
Insertar objeto	Siempre habilitado.
Pegado especial	El objeto **Clipboard** contiene un objeto válido.
Eliminar objeto incrustado	El control contenedor **OLE** contiene un objeto incrustado.
Eliminar objeto vinculado	El control contenedor **OLE** contiene un objeto vinculado.
Crear vínculo	La propiedad **SourceDoc** está establecida.
Crear objeto incrustado	La propiedad **Class** o la propiedad **SourceDoc** está establecida.

Un control contenedor **OLE** sólo puede contener un objeto cada vez. Puede crear un objeto vinculado o un objeto incrustado de varias formas:

- Use los cuadros de diálogo **Insertar objeto** o **Pegado especial** (en tiempo de ejecución o en tiempo de diseño).
-
- Establezca la propiedad **Class** en la ventana Propiedades, haga clic en el control contenedor **OLE** con el botón secundario del *mouse* y después seleccione el comando apropiado (sólo en tiempo de diseño).
- Use el método apropiado del control contenedor **OLE**.

Buscar nombres de clase

Puede obtener una lista de los nombres de clase disponibles en una aplicación si selecciona la propiedad **Class** en la ventana Propiedades y hace clic en el botón **Propiedades**.
Nota El cuadro de diálogo **Insertar objeto** no presenta una lista de nombres de clases. Este cuadro de diálogo presenta nombres descriptivos de cada clase de objeto, que suelen ser más largos y más comprensibles.

PROPIEDADES DE Contenedor OLE (Control)

Action (Propiedad, contenedor OLE)
AppIsRunning (Propiedad)
Appearance (Propiedad)
AutoActivate (Propiedad)
AutoVerbMenu (Propiedad)
BackColor, ForeColor (Propiedades)
BackStyle (Propiedad)
BorderStyle (Propiedad)
Class (Propiedad)
Container (Propiedad)
Data (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataText (Propiedad)
DisplayType (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
FileNumber (Propiedad)
Format (Propiedad, control contenedor OLE)
Height, Width (Propiedades)
HelpContextID (Propiedad)
HostName (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
MiscFlags (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropAllowed (Propiedad)
OLEType (Propiedad)
OLETypeAllowed (Propiedad)
Object (Propiedad, contenedor OLE)
ObjectAcceptFormats (Propiedad)
ObjectAcceptFormatsCount (Propiedad)
ObjectGetFormats (Propiedad)
ObjectGetFormatsCount (Propiedad)
ObjectVerbFlags (Propiedad)
ObjectVerbs (Propiedad)
ObjectVerbsCount (Propiedad)
Parent (Propiedad)
PasteOK (Propiedad)
Picture (Propiedad)

SizeMode (Propiedad)
SourceDoc (Propiedad)
SourceItem (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
UpdateOptions (Propiedad)
Verb (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)
lpOleObject (Propiedad)

BIBLIOTECA UNADECA ALAJUELA COSTA RICA

METODOS DE Contenedor OLE (Control)
Close (Método, contenedor OLE)
Copy (Método, control contenedor OLE)
CreateEmbed (Método)
CreateLink (Método)
Delete (Método, contenedor OLE)
DoVerb (Método)
Drag (Método)
Fetch

EVENTOS DE Contenedor OLE (Control)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
ObjectMove (Evento)
Resize (Evento)
Updated (Evento)

Control de datos ADO

Vea también Ejemplo Propiedades Métodos Eventos

El **Control de datos ADO** es similar al control intrínseco **Data** y al **Control de datos remotos** (RDC). El **Control de datos ADO** permite crear rápidamente una conexión con una base de datos mediante **Objetos de datos ActiveX de Microsoft** (ADO).

Comentarios

Es posible crear en tiempo de diseño una conexión al establecer la propiedad **ConnectionString** con una cadena de conexión válida y, a continuación, la propiedad **RecordSource** con una instrucción apropiada para el administrador de base de datos. Puede establecer también la propiedad **ConnectionString** con el nombre de un archivo que defina una conexión; el archivo se genera mediante el cuadro de diálogo **Vínculo de datos** que aparece cuando hace clic en **ConnectionString** en la ventana Propiedades y, después, en **Generar** o en **Seleccionar**.

Conecte el **Control de datos ADO** a un control enlazado a datos como **DataGrid**, **DataCombo** o **DataList**; para esto, establezca la propiedad **DataSource** con el **Control de datos ADO**.

En tiempo de ejecución, establezca dinámicamente las propiedades **ConnectionString** y **RecordSource** para cambiar la base de datos. Opcionalmente, puede establecer la propiedad **Recordset** directamente con un conjunto de registros abierto previamente.

PROPIEDADES DE Control de datos ADO

Align (Propiedad)
Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Caption (Propiedad)
CausesValidation (Propiedad)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Font (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
Name (Propiedad)
Object (Propiedad)
Orientation (Propiedad)
Parent (Propiedad)
Password (Propiedad, control de datos ADO)
RecordSource (Propiedad, control de datos ADO)
Recordset (Propiedad, control de datos ADO)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)

UserName (Propiedad, control de datos ADO)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)

METODOS DE Control de datos ADO
Drag (Método)
Move (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
UpdateControls (Método, objeto BindingCollection)
ZOrder (Método)

EVENTOS DE Control de datos ADO
DragDrop (Evento)
DragOver (Evento)
Error (Evento, control de datos ADO)
GotFocus (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
Validate (Evento)

DataRepeater (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control **DataRepeater** funciona como un contenedor desplazable de controles de usuario enlazados a datos. Cada control se muestra en su propia fila como un control "repetido", de forma que el usuario puede ver varios controles enlazados a datos a la vez.

Sintaxis
DataRepeater

Comentarios
Para usar el control **DataRepeater**, debe crear primero un control de usuario enlazado a datos y compilarlo en un archivo .ocx. Después de crear el control de usuario, es necesario efectuar los siguientes pasos básicos:

- Agregue al proyecto el control de usuario mediante el cuadro de diálogo **Componentes**. Esto garantiza que el Asistente de empaquetado y distribución de Visual Basic incluirá los archivos apropiados. Asimismo, permite tener acceso a las propiedades públicas, los eventos y los métodos del control.

- En la ventana **Propiedades**, haga clic en **RepeatedControlName** y seleccione el control de usuario en la lista desplegable.

- Agregue al formulario un origen de datos, como el control de datos de ADO, y conéctelo con un proveedor de datos.

- Establezca la propiedad **DataSource** del control **DataRepeater** en el origen de datos.

- Haga clic con el botón secundario del *mouse* (ratón) en el control **DataRepeater** y, a continuación, haga clic en **Propiedades** del **DataRepeater**.

- Haga clic en la ficha **RepeaterBindings**.

- Establezca la propiedad **PropertyName** con un objeto **DataField** apropiado y haga clic en el botón **Agregar**.

El control **DataRepeater** ahorra los recursos del equipo al mostrar solamente un único control de usuario (el control activo) a la vez. Los demás controles son simples imágenes que no mantienen una conexión individual con el origen de datos, como ocurriría si el formulario contuviera varios controles de usuario.

Nota para la distribución Cuando cree o distribuya aplicaciones que empleen el control **DataRepeater**, debe instalar MSDatRep.ocx en el subdirectorio System o System32 de Microsoft Windows del equipo del cliente. El kit de instalación incluido con Visual Basic ofrece herramientas que le ayudan a escribir programas para instalar correctamente sus aplicaciones.

PROPIEDADES DE DataRepeater (Control)

ActiveRow (Propiedad)
Appearance (Propiedad, controles ActiveX)
BackColor, ForeColor (Propiedades, controles ActiveX)
BorderStyle (Propiedad, controles ActiveX)
Caption (Propiedad, controles ActiveX)
CaptionStyle (Propiedad)
CausesValidation (Propiedad)
Container (Propiedad)
CurrentRecord (Propiedad)

DataBindings (Propiedad)
DataFields (Propiedad)
DataMember (Propiedad, controles ActiveX)
DataSource (Propiedad, controles ActiveX)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad, controles ActiveX)
Height, Width (Propiedades)
HelpContextID (Propiedad)
hWnd (Propiedad, controles ActiveX)
Index (Propiedad, matriz de controles)
IntegralHeight (Propiedad, control DataRepeater)
Left, Top (Propiedades)
MouseIcon (Propiedad, controles ActiveX)
MousePointer (Propiedad, controles ActiveX)
Name (Propiedad)
Object (Propiedad)
Parent (Propiedad)
PropertyNames (Propiedad)
RecordCount (Propiedad)
RepeatedControl (Propiedad)
RepeatedControlName (Propiedad)
RepeaterBindings (Propiedad)
RightToLeft (Propiedad, controles ActiveX)
RowDividerStyle (Propiedad, control DataRepeater)
RowIndicator (Propiedad)
Scrollbars (Propiedad, control DataRepeater)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
Visible (Propiedad)
VisibleRecords (Propiedad)
VisibleRows (Propiedad, control DataRepeater)
WhatsThisHelpID (Propiedad)

METODOS DE DataRepeater (Control)

Clear (Método, controles ActiveX)
Drag (Método)
Move (Método)
Refresh (Método, controles ActiveX)
SetFocus (Método)

ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE DataRepeater (Control)
ActiveRowChanged (Evento)
Click (Evento, controles ActiveX)
CurrentRecordChanged (Evento)
DataUpdate (Evento)
DblClick (Evento, controles ActiveX)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos, controles ActiveX)
KeyPress (Evento, controles ActiveX)
LostFocus (Evento)
MouseDown, MouseUp (Eventos, controles ActiveX)
MouseMove (Evento, controles ActiveX)
RepeatedControlLoaded, RepeatedControlUnloaded (Eventos)
Validate (Evento)
VisibleRecordChanged (Evento)

DirListBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **DirListBox** muestra directorios y rutas de acceso en tiempo de ejecución. Utilice este control para mostrar una lista jerárquica de directorios. Puede crear cuadros de diálogo que, por ejemplo, permitan a un usuario abrir un archivo desde una lista de archivos de todos los directorios disponibles.

Sintaxis
DirListBox

Comentarios
Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir al usuario tener acceso a los elementos de una lista. Si también muestra los controles **DriveListBox** y **FileListBox**, puede escribir código para sincronizarlos con el control **DirListBox** y entre sí.

PROPIEDADES DE DirListBox (Control)

Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Path (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE DirListBox (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE DirListBox (Control)

Change (Evento)
Click (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Scroll (Evento)
Validate (Evento)

DriveListBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos

Un control **DriveListBox** permite al usuario seleccionar una unidad de disco válida en tiempo de ejecución. Utilice este control para mostrar una lista de todas las unidades válidas del sistema de un usuario. Puede crear cuadros de diálogo que permitan al usuario abrir un archivo de una lista de un disco en cualquier unidad disponible.

Sintaxis

DriveListBox

Comentarios

Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir al usuario tener acceso a los elementos de la lista. Si además muestra los controles **DirListBox** y **FileListBox**, podrá escribir código para sincronizarlos con el control **DriveListBox** y entre sí.

PROPIEDADES DE DriveListBox (Control)

Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Drive (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE DriveListBox (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE DriveListBox (Control)

Change (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Scroll (Evento)
Validate (Evento)

Extender (Objeto)

Vea también Ejemplo Propiedades Métodos Eventos

Un objeto **Extender** almacena las propiedades del control tratadas por el contenedor del control y no por el control propiamente dicho.

Sintaxis

Extender

Comentarios

El contenedor, y no el control, proporciona algunas propiedades del control; éstas son propiedades extendidas. Ejemplos de propiedades extendidas son **Name**, **Tag** y **Left**. El control todavía necesita conocer el valor de dichas propiedades extendidas y, algunas veces, ser capaz de modificar una propiedad extendida; el objeto **Extender** proporciona al control el acceso a esas propiedades.

Algunas propiedades extendidas son estándar, mientras que otras son específicas de ciertos contenedores. Un control puede tener acceso a propiedades extendidas no estándar, pero esto hará que el control sea específico del contenedor. Si el control hace uso de una propiedad extendidas, debe tratar los casos donde el contenedor actual no admita la propiedad extendidas.

Cuando se compila el control, Visual Basic no tiene ninguna manera de saber las propiedades extendidas que pueden estar disponibles cuando se ejecute el control; por tanto, las referencias a las propiedades extendidas siempre se enlazan en tiempo de ejecución.

El objeto **Extender** no está disponible cuando se produce el evento Initialize, pero está disponible cuando se produce el evento InitProperties o el evento ReadProperties.

El objeto **Extender** tiene varias propiedades estándar:

- La propiedad **Name**: una cadena de sólo lectura que contiene el nombre del control definido por el usuario.
-
- La propiedad **Visible**: un valor de tipo **Boolean** de lectura y escritura que especifica si el control es visible.
- La propiedad **Parent**: un objeto de sólo lectura que representa el contenedor del control, como un formulario de Visual Basic.
-
- La propiedad **Cancel**: un valor de tipo **Boolean** de sólo lectura que indica que el control es el botón para cancelar predeterminado del contenedor.
-
- La propiedad **Default**: un valor de tipo **Boolean** de sólo lectura que indica que el control es el botón predeterminado del contenedor.

Visual Basic proporciona más métodos, propiedades y eventos extendidos. No se garantiza que otros contenedores proporcionen estos métodos, propiedades y eventos extendidos. Los métodos, propiedades y eventos extendidos específicos de Visual Basic son:

- La propiedad **Container**: un objeto de sólo lectura que representa el contenedor visual del control.

- La propiedad **DragIcon**: una imagen de lectura y escritura que especifica el icono que se utiliza cuando se arrastra el control.

- La propiedad **DragMode**: un valor de tipo **Integer** de lectura y escritura que especifica si el control se arrastrará automáticamente o si el usuario del control tendrá que llamar al método **Drag**.

- La propiedad **Enabled**: un valor de tipo **Boolean** de sólo lectura que especifica si el control está habilitado. Esta propiedad extendida no se presenta a menos que el control tenga también una propiedad **Enabled** con el Id. del procedimiento correcto. Para obtener información adicional, vea el tema "Permitir que el control se habilite y se deshabilite" en el capítulo 9, "Generar controles ActiveX".

- La propiedad **Height**: un valor de tipo **Integer** de lectura y escritura que especifica el alto del control en las unidades de escala del contenedor.

- La propiedad **HelpContextID**: un valor de tipo **Integer** de lectura y escritura que especifica el Id. de contexto que se utiliza si se presiona la tecla F1 cuando el control tiene el enfoque.

- La propiedad **Index**: un valor de tipo **Integer** de sólo lectura que especifica la ubicación que ocupa esta instancia del control dentro de una matriz de controles.

- La propiedad **Left**: un valor de tipo **Integer** de lectura y escritura que especifica la ubicación del borde izquierdo del control respecto al borde izquierdo del contenedor, en las unidades de escala del contenedor.

- La propiedad **TabIndex**: un valor de tipo **Integer** de lectura y escritura que especifica la ubicación del control en el orden de tabulación de los controles del contenedor.

- La propiedad **TabStop**: un valor de tipo **Boolean** de lectura y escritura que especifica si el tabulador se detendrá en este control.

- La propiedad **Tag**: una cadena de lectura y escritura que contiene un valor definido por el usuario.

- La propiedad **ToolTipText**: una cadena de lectura y escritura que contiene el texto que se presenta cuando el cursor se detiene sobre el control durante más de un segundo.

- La propiedad **Top**: un valor de tipo **Integer** de lectura y escritura que especifica la ubicación del borde superior del control respecto al borde superior del contenedor, especificada en las unidades de escala del contenedor.

- La propiedad **WhatThisHelpID**: un valor de tipo **Integer** de lectura y escritura que especifica el Id. de contexto que se emplea cuando se utiliza el cuadro emergente **Qué es esto** en el control.

- La propiedad **Width**: un valor de tipo **Integer** de lectura y escritura que especifica el ancho del control en las unidades de escala del contenedor.

- El método **Drag**: un método para iniciar, terminar o cancelar una operación de arrastre del control.

- El método **Move**: un método para mover la ubicación del control.

- El método **SetFocus**: un método para establecer el enfoque en el control.

- El método **ShowWhatsThis**: un método para presentar un tema seleccionado de un archivo de Ayuda mediante el cuadro emergente **Qué es esto** proporcionado por la Ayuda.

- El método **ZOrder**: un método para colocar el control al principio o al final del orden Z en su nivel gráfico.

- El evento DragDrop: un evento que se desencadena cuando otro control del formulario se coloca en este control.

- El evento DragOver: un evento que se desencadena cuando otro control del formulario se arrastra sobre este control.

- El evento GotFocus: un evento que se desencadena cuando este control recibe el enfoque.

- El evento LostFocus: un evento que se desencadena cuando este control pierde el enfoque.

PROPIEDADES DE Extender (Objeto)

Align (Propiedad)
Cancel (Propiedad)
CausesValidation (Propiedad)
Container (Propiedad)
DataBindings (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
Default (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, controles ActiveX)
Left, Top (Propiedades)
Name (Propiedad)
Object (Propiedad)
Parent (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)

METODOS DE Extender (Objeto)
Drag (Método)
Move (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE Extender (Objeto)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
LostFocus (Evento)
ObjectEvent (Evento)
Validate (Evento)

FileListBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control **FileListBox** encuentra y muestra los archivos del directorio especificado por la propiedad **Path** en tiempo de ejecución. Utilice este control para mostrar una lista de los archivos seleccionados por tipo. Puede crear cuadros de diálogo en la aplicación que, por ejemplo, permitan al usuario seleccionar un archivo o un grupo de archivos.

Sintaxis
FileListBox

Comentarios
Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir al usuario tener acceso a los elementos de la lista. Si además muestra los controles **DirListBox** y **DriveListBox**, podrá escribir código para sincronizarlos con el control **FileListBox** y entre sí.

PROPIEDADES DE FileListBox (Control)
Appearance (Propiedad)
Archive, Hidden, Normal, System (Propiedades)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
FileName (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)

FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
Locked (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
MultiSelect (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Path (Propiedad)
Pattern (Propiedad)
ReadOnly (Propiedad)
Selected (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE FileListBox (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE FileListBox (Control)

Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
PathChange (Evento)
PatternChange (Evento)
Scroll (Evento)
Validate (Evento)

Form (Objeto), Forms (Colección)

Vea también Ejemplo Propiedades Métodos Eventos

Un objeto **Form** es una ventana o un cuadro de diálogo que forma parte de la interfaz de usuario de una aplicación.

Forms es una colección cuyos elementos representan cada formulario cargado en una aplicación. La colección incluye el formulario MDI, el formulario secundario MDI y formularios no MDI de la aplicación. La colección **Forms** tiene una única propiedad, **Count**, que especifica el número de elementos de la colección.

Sintaxis

Form

Forms(*índice*)

El marcador de posición *índice* representa un entero entre 0 y `Forms.Count - 1`.

Comentarios

Puede usar la colección **Forms** para iterar a través de todos los formularios cargados en una aplicación. Identifica una variable global intrínseca denominada **Forms**. Puede pasar **Forms(*índice*)** a una función cuyo argumento se especifica como una clase **Forms**.

Los formularios tienen propiedades que determinan aspectos de su apariencia (como posición, tamaño y color) y aspectos de su comportamiento (como si puede ajustar o no su tamaño).

Los formularios también pueden responder a eventos iniciados por un usuario o desencadenados por el sistema. Por ejemplo, podría escribir código en el procedimiento de evento Click de un formulario que permitiera al usuario cambiar el color de un formulario haciendo clic en él.

Además de propiedades y eventos, puede emplear métodos para manipular formularios mediante código. Por ejemplo, puede usar el método **Move** para cambiar la ubicación y el tamaño de un formulario.

Un tipo especial de formulario, el formulario MDI, puede contener otros formularios llamados formularios secundarios MDI. Un formulario MDI se crea con el comando **Agregar formulario MDI** del menú **Proyecto**; un formulario secundario MDI se crea eligiendo **Nuevo formulario** en el menú **Archivo** y, a continuación, estableciendo la propiedad **MDIChild** a **True**.

Puede crear múltiples instancias de formularios en código si utiliza la palabra clave **New** en instrucciones **Dim**, **Set** y **Static**.

Al diseñar formularios, establezca la propiedad **BorderStyle** para definir el borde del formulario y establezca la propiedad **Caption** para colocar texto en la barra de título. En el código puede usar los métodos **Hide** y **Show** para hacer que los formularios sean invisibles o visibles en tiempo de ejecución.

Nota Si establece **BorderStyle** a 0 se eliminará el borde. Si desea que el formulario tenga borde sin barra de título, cuadro del menú **Control**, botón **Maximizar** y botón **Minimizar**, elimine texto de la propiedad **Caption** y establezca a **False** las propiedades **ControlBox**, **MaxButton** y **MinButton**.

Form es un tipo de datos Object. Puede declarar variables como de tipo **Form** antes de establecerlas a una instancia de un tipo de formulario que se ha declarado en tiempo de diseño. Del mismo modo, puede pasar un argumento a un procedimiento como de tipo **Form**.
Los formularios pueden actuar como origen en una conversación DDE, con un control **Label**, **PictureBox** o **TextBox** proporcionando los datos.

Puede tener acceso a la colección de controles de un **Form** mediante la colección **Controls**. Por ejemplo, para ocultar todos los controles de un **Form** puede usar código similar al siguiente:

```
For Each Control in Form1.Controls
   Control.Visible = False
Next Control
```

PROPIEDADES DE Form (Objeto), Forms (Colección)

ActiveControl (Propiedad)
Appearance (Propiedad)
AutoRedraw (Propiedad)
BackColor, ForeColor (Propiedades)
BorderStyle (Propiedad)
Caption (Propiedad)
ClipControls (Propiedad)
ControlBox (Propiedad)
Controls (Propiedad)
Count (Propiedad, colecciones de VB)
CurrentX, CurrentY (Propiedades)
DrawMode (Propiedad)
DrawStyle (Propiedad)
DrawWidth (Propiedad)
Enabled (Propiedad)
FillColor (Propiedad)
FillStyle (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
FontTransparent (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Icon (Propiedad)
Image (Propiedad)
KeyPreview (Propiedad)
Left, Top (Propiedades)
LinkMode (Propiedad)
LinkTopic (Propiedad)
MDIChild (Propiedad)
MaxButton (Propiedad)
MinButton (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Moveable (Propiedad)
Name (Propiedad)
NegotiateMenus (Propiedad)

OLEDropMode (Propiedad)
Palette (Propiedad)
PaletteMode (Propiedad)
Picture (Propiedad)
RightToLeft (Propiedad)
ScaleHeight, ScaleWidth (Propiedades)
ScaleLeft, ScaleTop (Propiedades)
ScaleMode (Propiedad)
ShowInTaskbar (Propiedad)
StartUpPosition (Propiedad)
Tag (Propiedad)
Visible (Propiedad)
WhatsThisButton (Propiedad)
WhatsThisHelp (Propiedad)
WindowState (Propiedad)
hDC (Propiedad)
hWnd (Propiedad)

METODOS DE Form (Objeto), Forms (Colección)

Circle (Método)
Cls (Método)
Hide (Método)
Line (Método)
Move (Método)
OLEDrag (Método)
PSet (Método)
PaintPicture (Método)
Point (Método)
PopupMenu (Método)
PrintForm (Método)
Refresh (Método)
Scale (Método)
ScaleX, ScaleY (Métodos)
SetFocus (Método)
Show (Método)
TextHeight (Método)
TextWidth (Método)
WhatsThisMode (Método)
ZOrder (Método)

EVENTOS DE Form (Objeto), Forms (Colección)

Activate, Deactivate (Eventos)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
Initialize (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LinkClose (Evento)
LinkError (Evento)
LinkExecute (Evento)
LinkOpen (Evento)
Load (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Paint (Evento)
QueryUnload (Evento)
Resize (Evento)
Terminate (Evento)
Unload (Evento)

HScrollBar, VScrollBar (Controles)

Vea también Ejemplo Propiedades Métodos Eventos

Las barras de desplazamiento permiten explorar fácilmente una larga lista de elementos o una gran cantidad de información. Además, proporcionan una representación análoga de la posición actual. Puede usar una barra de desplazamiento como dispositivo de entrada o como indicador de velocidad o cantidad; por ejemplo, para controlar el volumen de un juego de PC o para ver el tiempo transcurrido en un proceso temporizado.

Sintaxis
HScrollBar
VScrollBar

Comentarios
Cuando utiliza una barra de desplazamiento como indicador de cantidad o velocidad, o como dispositivo de entrada, utilice las propiedades **Max** y **Min** para establecer el intervalo adecuado del control.
Para especificar la cantidad de cambio que va a indicar en una barra de desplazamiento, utilice la propiedad **LargeChange** para hacer clic en la barra de desplazamiento y la propiedad **SmallChange** para hacer clic en las flechas de los extremos de la barra. La propiedad **Value** de la barra de desplazamiento aumenta o disminuye los valores establecidos para las propiedades **LargeChange** y **SmallChange**. Puede colocar el cuadro de desplazamiento en tiempo de ejecución si establece **Value** entre 0 y 32.767, ambos inclusive.

PROPIEDADES DE HScrollBar, VScrollBar (Controles)

Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
LargeChange, SmallChange (Propiedades)
Left, Top (Propiedades)
Max, Min (Propiedades, barra de desplazamiento)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
Parent (Propiedad)
RightToLeft (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
Value (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE HScrollBar, VScrollBar (Controles)

Drag (Método)
Move (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE HScrollBar, VScrollBar (Controles)

Change (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
Scroll (Evento)
Validate (Evento)

ListBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos

Un control **ListBox** muestra una lista de elementos entre los cuales el usuario puede seleccionar uno o más. Si el número de elementos supera el número que puede mostrarse, se agregará automáticamente una barra de desplazamiento al control **ListBox**.

Si no se selecciona ningún elemento, el valor de la propiedad **ListIndex** será -1. El primer elemento de la lista es **ListIndex** 0 y el valor de la propiedad **ListCount** siempre es uno más que el mayor valor de **ListIndex**.

Sintaxis

ListBox

Comentarios

Para agregar o eliminar elementos de un control **ListBox**, use el método **AddItem** o **RemoveItem**. Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir que un usuario tenga acceso a elementos del **ListBox**. También puede agregar elementos a la lista mediante la propiedad **List** en tiempo de diseño.

PROPIEDADES DE ListBox (Control)

Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Columns (Propiedad, ListBox)
Container (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
IntegralHeight (Propiedad)
ItemData (Propiedad)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
MultiSelect (Propiedad)
Name (Propiedad)
NewIndex (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
RightToLeft (Propiedad)
SelCount (Propiedad)
Selected (Propiedad)
Sorted (Propiedad)
Style (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
Text (Propiedad)
ToolTipText (Propiedad)

TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE ListBox (Control)

AddItem (Método)
Clear (Método para Clipboard, ComboBox y ListBox)
Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
RemoveItem (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE ListBox (Control)

Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
ItemCheck (Evento, control ListBox)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Scroll (Evento)
Validate (Evento)

MSFlexGrid (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control Microsoft FlexGrid (**MSFlexGrid**) muestra datos de tablas y efectúa operaciones en ellos. Proporciona una flexibilidad completa para ordenar, combinar y aplicar formato a tablas que contienen cadenas e imágenes. Cuando se enlaza a un control **Data**, el control **MSFlexGrid** muestra datos de sólo lectura.

Sintaxis
MSFlexGrid

Comentarios
Es posible colocar texto, una imagen o ambas cosas en cualquier celda de un control **MSFlexGrid**. Las propiedades **Row** y **Col** especifican la celda actual de un control **MSFlexGrid**. Puede especificar la celda actual mediante código o bien el usuario puede cambiar la misma en tiempo de ejecución mediante el *mouse* o las flechas del cursor. La propiedad **Text** hace referencia al contenido de la celda actual.

Si el texto de una celda es demasiado largo para que se pueda mostrar en la celda y la propiedad **WordWrap** está establecida a **True**, el texto se ajusta a la línea siguiente dentro de la misma celda. Para mostrar el texto ajustado, es posible que sea necesario aumentar el ancho de columna de la celda (propiedad **ColWidth**) o el alto de la fila (propiedad **RowHeight**).

Utilice las propiedades **Col** y **Row** para determinar el número de columnas y filas de un control **MSFlexGrid**.

Nota Antes de poder usar un control **MSFlexGrid** en la aplicación, es necesario agregar el archivo MSFlxGrd.ocx al proyecto. Para incluir automáticamente el archivo en un proyecto, colóquelo en el archivo Autoload. Al distribuir la aplicación, conviene instalar el archivo MSFlxGrd.ocx en el directorio System de Microsoft Windows del usuario. Para obtener más información acerca de cómo agregar un control ActiveX a un proyecto, vea "Controles ActiveX estándar" en el *Manual del programador* de Visual Basic.

PROPIEDADES DE MSFlexGrid (Control)

AllowBigSelection (Propiedad)
AllowUserResizing (Propiedad)
Appearance (Propiedad, controles ActiveX)
BackColor, BackColorBkg, BackColorFixed, BackColorSel (Propiedades)
BorderStyle (Propiedad, controles ActiveX)
CellAlignment (Propiedad)
CellBackColor, CellForeColor (Propiedades)

CellFontBold (Propiedad)
CellFontItalic (Propiedad)
CellFontName (Propiedad)
CellFontSize (Propiedad)
CellFontStrikeThrough (Propiedad)
CellFontUnderline (Propiedad)
CellFontWidth (Propiedad)
CellHeight, CellLeft, CellTop, CellWidth (Propiedades, control MSHFlexGrid)
CellPicture (Propiedad)
CellPictureAlignment (Propiedad)
CellTextStyle (Propiedad)
Clip (Propiedad, control MSHFlexGrid)
Col, Row (Propiedades, control MSHFlexGrid)
ColAlignment, ColAlignmentBand, ColAlignmentHeader (Propiedades, control MSHFlexGrid)
ColData, RowData, BandData (Propiedades, control MSHFlexGrid)
ColIsVisible (Propiedad)
ColPos (Propiedad)
ColPosition, RowPosition (Propiedades)
Cols, Rows (Propiedades, control MSHFlexGrid)
ColSel, RowSel (Propiedades)
ColWidth (Propiedad, control MSHFlexGrid)
Container (Propiedad)
DataBindings (Propiedad)
DataSource (Propiedad, controles ActiveX)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad, controles ActiveX)
FillStyle (Propiedad, control MSHFlexGrid)
FixedCols, FixedRows (Propiedades, control MSHFlexGrid)
FocusRect (Propiedad)
FontWidth, FontWidthBand, FontWidthFixed, FontWidthHeader (Propiedades, control MSHFlexGrid)
ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader, ForeColorSel (Propiedades)
FormatString (Propiedad)
GridColor, GridColorBand, GridColorFixed, GridColorHeader, GridColorIndent, GridColorUnpopulated (Propiedades)
GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent, GridLinesUnpopulated (Propiedades, control MSHFlexGrid)

GridLineWidth, GridLineWidthBand, GridLineWidthFixed, GridLineWidthHeader, GridLineWidthIndent, GridLineWidthUnpopulated (Propiedades, control MSHFlexGrid)
Height, Width (Propiedades, controles ActiveX)
HelpContextID (Propiedad)
HighLight (Propiedad, control MSHFlexGrid)
hWnd (Propiedad, controles ActiveX)
Index (Propiedad, controles ActiveX)
Left, Top (Propiedades, controles ActiveX)
LeftCol (Propiedad, control MSHFlexGrid)
MergeCells (Propiedad)
MergeCol, MergeRow (Propiedades)
MouseCol, MouseRow (Propiedades)
MouseIcon (Propiedad)
MousePointer (Propiedad, controles ActiveX)
Name (Propiedad, control MSHFlexGrid)
Object (Propiedad, controles ActiveX)
OLEDropMode (Propiedad, controles ActiveX)
Parent (Propiedad)
Picture (Propiedad, control MSHFlexGrid)
PictureType (Propiedad)
Redraw (Propiedad)
RowHeight (Propiedad, control MSHFlexGrid)
RowHeightMin (Propiedad)
RowIsVisible (Propiedad)
RowPos (Propiedad)
ScrollBars (Propiedad, control MSHFlexGrid)
ScrollTrack (Propiedad)
SelectionMode (Propiedad)
Sort (Propiedad, control MSHFlexGrid)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad, controles ActiveX)
Text (Propiedad, control MSHFlexGrid)
TextArray (Propiedad)
TextMatrix (Propiedad)
TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader (Propiedades, control MSHFlexGrid)
ToolTipText (Propiedad)
TopRow (Propiedad, control MSHFlexGrid)
Version (Propiedad, control MSHFlexGrid)
Visible (Propiedad, controles ActiveX)
WhatsThisHelp (Propiedad)
WordWrap (Propiedad, control MSHFlexGrid)

METODOS DE MSFlexGrid (Control)

AddItem (Método, MSHFlexGrid)
Clear (Método, control MSHFlexGrid)
Drag (Método)
Move (Método)
OLEDrag (Método, controles ActiveX)
Refresh (Método, controles ActiveX)
RemoveItem (Método, control MSHFlexGrid)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE MSFlexGrid (Control)

Click (Evento, controles ActiveX)
Compare (Evento)
DragDrop (Evento)
DragOver (Evento)
EnterCell (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos, controles ActiveX)
KeyPress (Evento, controles ActiveX)
LeaveCell (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento, controles ActiveX)
OLEDragDrop (Evento, controles ActiveX)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento, controles ActiveX)
OLESetData (Evento, controles ActiveX)
OLEStartDrag (Evento, controles ActiveX)
RowColChange (Evento, control MSHFlexGrid)
Scroll (Evento, control MSHFlexGrid)
SelChange (Evento, control MSHFlexGrid)

MSHFlexGrid (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control **MSHFlexGrid** (Microsoft Hierarchical FlexGrid) muestra datos de tablas y efectúa operaciones en ellos. Proporciona una completa flexibilidad para ordenar, fusionar y dar formato a tablas que contienen cadenas e imágenes. Cuando está enlazado a un control de datos, **MSHFlexGrid** muestra únicamente datos de sólo lectura.

Sintaxis
MSHFlexGrid

Comentarios
Puede introducir texto, un gráfico o ambos en cualquier celda de un control **MSHFlexGrid**. Las propiedades **Row** y **Col** especifican la celda actual del control **MSHFlexGrid**. Puede especificar la celda actual utilizando código o el usuario puede cambiarla en tiempo de ejecución con el *mouse* (ratón) o con las teclas de dirección. La propiedad **Text** hace referencia al contenido de la celda actual.

Si el texto de una celda es demasiado grande para mostrarse en la celda y la propiedad **WordWrap** está establecida a **True**, el texto se ajusta a la siguiente línea de la misma celda. Para mostrar todo el texto, puede que necesite aumentar el ancho de la columna (propiedad **ColWidth**) o el alto de la fila (propiedad **RowHeight**).

Utilice las propiedades **Cols** y **Rows** para determinar el número de columnas y filas del control **MSHFlexGrid**. Utilice las propiedades **Band** para determinar el estilo de la banda en un control **MSHFlexGrid**.

Mostrar un conjunto de registros jerárquico

La principal característica de un control **MSHFlexGrid** es la posibilidad de mostrar conjuntos de registros jerárquicos (las tablas relacionadas se muestran de forma jerárquica). La forma más sencilla de crear conjuntos de registros jerárquicos es usar el diseñador de entorno de datos y asignar la propiedad **DataSource** del control **MSHFlexGrid** al entorno de datos. Puede también crear conjuntos de registros jerárquicos en el código utilizando el comando

Shape como el objeto **RecordSource** para un control **ADO Data**, como se muestra en el ejemplo siguiente:

```
' Crea un cadena de conexión.
Dim strCn As String
strCn = "Provider=MSDataShape.1;Data Source=Nwind;" & _
"Connect Timeout=15;Data Provider=MSDASQL"

' Crea un comando Shape.
```

```
Dim strSh As String
strSh = "SHAPE {SELECT * FROM 'Clientes'}  AS Clientes" & _
"APPEND ({SELECT * FROM 'Pedidos'}  AS Pedidos RELATE " & _
"IdCliente TO IdCliente ) AS Pedidos"

' Asignar la cadena de conexión ConnectionString a un
' control ADO Data y el comando Shape a la propiedad
' RecordSource del control.
With Adodc1
   .ConnectionString = strCn
   .RecordSource = strSh
End With
' Establece la propiedad DataSource del control HflexGrid al
' control ADO Data
Set HFlexGrid1.DataSource = Adodc1
```

Para obtener más información Para aprender más detalles acerca de los conjuntos de registros jerárquicos, vea Cursores jerárquicos y modelado de datos. Se puede encontrar también una descripción de los comandos **Shape** en Comandos de modelado en general.

Nota para la distribución Antes de poder usar el control **MSHFlexGrid** en la aplicación, debe agregar el archivo MSHFlxGd.ocx al proyecto. Para incluir el archivo en el proyecto automáticamente, colóquelo en el archivo Autoload. Cuando distribuya la aplicación, debe instalar el archivo MSHFlxGd.ocx en el directorio System de Microsoft Windows del usuario. Para obtener más información acerca de cómo puede agregar un control ActiveX a un proyecto, vea "Controles ActiveX estándar" en el *Manual del programador de Visual Basic.*

PROPIEDADES DE MSHFlexGrid (Control)

AllowBigSelection (Propiedad)
AllowUserResizing (Propiedad)
Appearance (Propiedad, controles ActiveX)
BackColor, BackColorBkg, BackColorFixed, BackColorSel (Propiedades)
BackColorBand, BackColorHeader, BackColorIndent, BackColorUnpopulated (Propiedades)
BandColIndex (Propiedad, MSHFlexGrid)
BandDisplay (Propiedad de MSHFlexGrid)
BandExpandable (Propiedad de MSHFlexGrid)
BandIndent (Propiedad de MSHFlexGrid)
BandLevel (Propiedad de MSHFlexGrid)
Bands (Propiedad de MSHFlexGrid)
BorderStyle (Propiedad, controles ActiveX)
CellAlignment (Propiedad)
CellBackColor, CellForeColor (Propiedades)
CellFontBold (Propiedad)

CellFontItalic (Propiedad)
CellFontName (Propiedad)
CellFontSize (Propiedad)
CellFontStrikeThrough (Propiedad)
CellFontUnderline (Propiedad)
CellFontWidth (Propiedad)
CellHeight, CellLeft, CellTop, CellWidth (Propiedades, control MSHFlexGrid)
CellPicture (Propiedad)
CellPictureAlignment (Propiedad)
CellTextStyle (Propiedad)
CellType (Propiedad de MSHFlexGrid)
Clip (Propiedad, control MSHFlexGrid)
Col, Row (Propiedades, control MSHFlexGrid)
ColAlignment, ColAlignmentBand, ColAlignmentHeader (Propiedades, control MSHFlexGrid)
ColAlignmentFixed (Propiedad)
ColData, RowData, BandData (Propiedades, control MSHFlexGrid)
ColHeader (Propiedad, MSHFlexGrid)
ColHeaderCaption Property (MSHFlexGrid)
ColIsVisible (Propiedad)
Collapse (Evento, control MSHFlexGrid)
CollapseAll (Método, control MSHFlexGrid)
ColPos (Propiedad)
ColPosition, RowPosition (Propiedades)
Cols, Rows (Propiedades, control MSHFlexGrid)
ColSel, RowSel (Propiedades)
ColWidth (Propiedad, control MSHFlexGrid)
Container (Propiedad)
DataBindings (Propiedad)
DataField (Propiedad, control MSHFlexGrid)
DataMember (Propiedad, controles ActiveX)
DataSource (Propiedad, controles ActiveX)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad, controles ActiveX)
FillStyle (Propiedad, control MSHFlexGrid)
FixedCols, FixedRows (Propiedades, control MSHFlexGrid)
FocusRect (Propiedad)
Font, FontBand, FontFixed, FontHeader (Propiedades, control MSHFlexGrid)
FontWidth, FontWidthBand, FontWidthFixed, FontWidthHeader (Propiedades, control MSHFlexGrid)

ForeColor, ForeColorBand, ForeColorFixed, ForeColorHeader, ForeColorSel (Propiedades)
FormatString (Propiedad)
GridColor, GridColorBand, GridColorFixed, GridColorHeader, GridColorIndent, GridColorUnpopulated (Propiedades)
GridLines, GridLinesBand, GridLinesFixed, GridLinesHeader, GridLinesIndent, GridLinesUnpopulated (Propiedades, control MSHFlexGrid)
GridLineWidth, GridLineWidthBand, GridLineWidthFixed, GridLineWidthHeader, GridLineWidthIndent, GridLineWidthUnpopulated (Propiedades, control MSHFlexGrid)
Height, Width (Propiedades, controles ActiveX)
HelpContextID (Propiedad)
HighLight (Propiedad, control MSHFlexGrid)
hWnd (Propiedad, controles ActiveX)
Index (Propiedad, controles ActiveX)
Left, Top (Propiedades, controles ActiveX)
LeftCol (Propiedad, control MSHFlexGrid)
MergeCells (Propiedad)
MergeCol, MergeRow (Propiedades)
MouseCol, MouseRow (Propiedades)
MouseIcon (Propiedad, controles ActiveX)
MousePointer (Propiedad, controles ActiveX)
Name (Propiedad, control MSHFlexGrid)
Object (Propiedad, controles ActiveX)
OLEDropMode (Propiedad, controles ActiveX)
Parent (Propiedad)
Picture (Propiedad, control MSHFlexGrid)
PictureType (Propiedad)
Recordset (Propiedad)
Redraw (Propiedad)
RowExpandable, RowExpanded (Propiedades, control MSHFlexGrid)
RowHeight (Propiedad, control MSHFlexGrid)
RowHeightMin (Propiedad)
RowIsVisible (Propiedad)
RowPos (Propiedad)
RowSizingMode (Propiedad, control MSHFlexGrid)
ScrollBars (Propiedad, control MSHFlexGrid)
ScrollTrack (Propiedad)
SelectionMode (Propiedad)
Sort (Propiedad, control MSHFlexGrid)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad, controles ActiveX)

Text (Propiedad, control MSHFlexGrid)
TextArray (Propiedad)
TextMatrix (Propiedad)
TextStyle, TextStyleBand, TextStyleFixed, TextStyleHeader (Propiedades, control MSHFlexGrid)
TopRow (Propiedad, control MSHFlexGrid)
Version (Propiedad, control MSHFlexGrid)
Visible (Propiedad, controles ActiveX)
WhatsThisHelpID (Propiedad)
WordWrap (Propiedad, control MSHFlexGrid)

METODOS DE MSHFlexGrid (Control)

AddItem (Método, MSHFlexGrid)
Clear (Método, control MSHFlexGrid)
ClearStructure (Método, MSHFlexGrid)
CollapseAll (Método, control MSHFlexGrid)
Drag (Método)
ExpandAll (Método, control MSHFlexGrid)
Move (Método)
OLEDrag (Método, controles ActiveX)
Refresh (Método, controles ActiveX)
RemoveItem (Método, control MSHFlexGrid)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE MSHFlexGrid (Control)

Click (Evento, controles ActiveX)
Collapse (Evento, control MSHFlexGrid)
Compare (Evento)
DragDrop (Evento)
DragOver (Evento)
EnterCell (Evento)
Expand (Evento, control MSHFlexGrid)
GotFocus (Evento)
KeyDown, KeyUp (Eventos, controles ActiveX)
KeyPress (Evento, controles ActiveX)
LeaveCell (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento, controles ActiveX)

OLEDragDrop (Evento, controles ActiveX)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento, controles ActiveX)
OLESetData (Evento, controles ActiveX)
OLEStartDrag (Evento, controles ActiveX)
RowColChange (Evento, control MSHFlexGrid)
Scroll (Evento, control MSHFlexGrid)
SelChange (Evento, control MSHFlexGrid)

Masked Edit (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control **Masked Edit** proporciona una entrada de datos restringida así como un resultado de datos con formato. Este control proporciona pistas visuales acerca del tipo de datos que se escribe o se muestra. El icono que lo representa en el cuadro de herramientas es el siguiente:

Nombre de archivo
MSMASK32.OCX

Nombre de clase
MaskEdBox

Comentarios
El control **Masked Edit** generalmente se comporta como un cuadro de texto estándar con mejoras para realizar entradas de datos con máscara y salidas con formato. Si no utiliza una máscara de entrada, el control **Masked Edit** se comporta como un cuadro de texto estándar, excepto por su capacidad de realizar intercambio dinámico de datos (DDE, dynamic data exchange).

Si define una máscara de entrada mediante la propiedad **Mask**, cada posición de un carácter del control **Masked Edit** se asigna a un marcador de posición de un tipo específico o a un carácter literal. Los caracteres literales, o *literales*, dan una indicación visual acerca del tipo de datos que está utilizando. Por ejemplo, los paréntesis que rodean el código de área de un número de teléfono son literales: `(206)`.
Si intenta especificar un carácter que entra en conflicto con la máscara de entrada, el control generará un evento ValidationError. La máscara de entrada le impide especificar caracteres no válidos en el control.

El control **Masked Edit** tiene tres propiedades dependientes: **DataChanged**, **DataField** y **DataSource**. Esto significa que se puede vincular a un control de datos y mostrar

valores de campo para el registro actual del recordset. El control **Masked Edit** puede también escribir valores de resultado al recordset.

Cuando se lee el valor del campo al que se hace referencia mediante la propiedad **DataField**, si es posible se convierte a una cadena para la propiedad **Text**. Si el conjunto de registros es actualizable, la cadena se convierte al tipo de datos del campo.
Para borrar la propiedad **Text** cuando tiene definida una máscara, primero necesita establecer la propiedad **Mask** a una cadena vacía y después hacer lo mismo con la propiedad **Text**:

```
MaskedEdit1.Mask = ""
MaskedEdit1.Text = ""
```

Cuando define una máscara de entrada, el control **Masked Edit** se comporta de forma diferente a un cuadro de texto estándar. El punto de inserción salta automáticamente los literales a medida que se mueve o se escriben datos.
Cuando inserta o elimina un carácter, todos los caracteres no literales que hay a la derecha del punto de inserción se desplazan, según sea necesario. Si el desplazamiento de estos caracteres produce un error de validación, se impide la inserción o eliminación y se desencadena un evento ValidationError.

Suponga que la propiedad **Mask** está definida como "?###" y el valor actual de la propiedad **Text** es "A12". Si intenta insertar la letra "B" delante de la letra "A", la "A" se desplazaría a la derecha. Como el segundo valor de la máscara de entrada requiere un número, la letra "A" haría que el control generara un evento ValidationError.

El control **Masked Edit** también valida los valores de la propiedad **Text** en tiempo de ejecución. Si establece la propiedad **Text** de modo que entre en conflicto con la máscara de entrada, el control generará un error en tiempo de ejecución.

Puede seleccionar texto del mismo modo que en un cuadro de texto estándar. Cuando el texto seleccionado se elimina, el control intenta desplazar los caracteres restantes a la derecha de la selección. Sin embargo, se elimina cualquier carácter restante que pueda causar un error de validación durante este desplazamiento y no se genera ningún evento ValidationError.

Normalmente, cuando una selección del control **Masked Edit** se copia al Portapapeles, la selección entera (incluyendo los literales) se transfiere al Portapapeles. Puede usar la propiedad **ClipMode** para transferir al Portapapeles únicamente los datos introducidos por el usuario: los caracteres literales que forman parte de la máscara de entrada no se copian.

PROPIEDADES DE Masked Edit (Control)

AllowPrompt (Propiedad)
Appearance (Propiedad, controles ActiveX)
AutoTab (Propiedad)
BackColor, ForeColor (Propiedades, controles ActiveX)
BorderStyle (Propiedad, controles ActiveX)
ClipMode (Propiedad)
ClipText (Propiedad)
Container (Propiedad)
DataBinding (Objeto)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad, controles ActiveX)
Font (Propiedad)
Format (Propiedad, control Masked Edit)
FormattedText (Propiedad)
Height, Width (Propiedades, controles ActiveX)
HelpContextID (Propiedad)
HideSelection (Propiedad, controles ActiveX)
Index (Propiedad, controles ActiveX)
Left, Top (Propiedades, controles ActiveX)
Mask (Propiedad)
MaxLength (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad, controles ActiveX)
Name (Propiedad)
OLEDragMode (Propiedad, controles ActiveX)
OLEDropMode (Propiedad, controles ActiveX)
Object (Propiedad, controles ActiveX)
Parent (Propiedad)
PromptChar (Propiedad)
PromptInclude (Propiedad)
SelLength, SelStart, SelText (Propiedades)
SelLength, SelStart, SelText (Propiedades, controles ActiveX)
SelText (Propiedad, control Masked Edit)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad, controles ActiveX)
Text (Propiedad, control Masked Edit)
ToolTipText (Propiedad)
Visible (Propiedad, controles ActiveX)

WhatsThisHelpID (Propiedad)
hWnd (Propiedad, controles ActiveX)

METODOS DE Masked Edit (Control)
DataBindings (Propiedad)
Drag (Método)
Move (Método)
OLEDrag (Método, controles ActiveX)
Refresh (Método, controles ActiveX)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE Masked Edit (Control)
Change (Evento, controles ActiveX)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos, controles ActiveX)
KeyPress (Evento, controles ActiveX)
LostFocus (Evento)
OLECompleteDrag (Evento, controles ActiveX)
OLEDragDrop (Evento, controles ActiveX)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento, controles ActiveX)
OLESetData (Evento, controles ActiveX)
OLEStartDrag (Evento, controles ActiveX)
Validate (Evento)
ValidationError (Evento)

OptionButton (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **OptionButton** muestra una opción que se puede activar o desactivar.

Sintaxis
OptionButton

Comentarios
Generalmente, los controles **OptionButton** se utilizan en un grupo de opciones para mostrar opciones entre las cuales el usuario sólo puede seleccionar una. Los controles **OptionButton** se agrupan si los dibuja dentro de un contenedor como un control **Frame**, un control **PictureBox** o un formulario. Para agrupar controles **OptionButton** en un **Frame** o **PictureBox**, dibuje en primer lugar el **Frame** o el **PictureBox** y, a continuación, dibuje dentro los controles **OptionButton**. Todos los controles **OptionButton** que están dentro del mismo contenedor actúan como un único grupo.

Aunque puede parecer que los controles **OptionButton** y **CheckBox** funcionan de forma similar, hay una diferencia importante: cuando un usuario selecciona un **OptionButton**, los otros controles **OptionButton** del mismo grupo dejan de estar disponibles automáticamente. Por el contrario, es posible seleccionar cualquier número de controles **CheckBox**.

PROPIEDADES DE OptionButton (Control)

Alignment (Propiedad)
Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
Caption (Propiedad)
Container (Propiedad)
DataFormat (Propiedad)
DisabledPicture (Propiedad)
DownPicture (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
MaskColor (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)

Parent (Propiedad)
Picture (Propiedad)
RightToLeft (Propiedad)
Style (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
UseMaskColor (Propiedad)
Value (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE OptionButton (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE OptionButton (Control)

Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Validate (Evento)

PictureBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos

Un control **PictureBox** puede mostrar un gráfico a partir de un mapa de bits, un icono o un metarchivo, así como un metarchivo mejorado, un archivo JPEG o un archivo GIF. Recorta el gráfico si el control no es lo bastante grande para mostrar la imagen completa.

Sintaxis
PictureBox

Comentarios
También puede usar un control **PictureBox** para agrupar controles **OptionButton** y mostrar el resultado de métodos gráficos y texto escrito con el método **Print**.

Para hacer que un control **PictureBox** ajuste automáticamente su tamaño para mostrar un gráfico completo, establezca su propiedad **AutoSize** a **True**.

Para crear animación o simulación, puede manipular propiedades y métodos gráficos en código. Las propiedades y los eventos gráficos son útiles para operaciones de impresión en tiempo de ejecución, como modificar el formato de un formulario de pantalla para imprimirlo.

Un control **PictureBox** también puede actuar como vínculo de destino en una conversación DDE.

Los controles **PictureBox** y **Data** son los únicos controles estándar de Visual Basic que puede situar en el área interna de un formulario MDI. Puede emplearlo para agrupar controles en la parte superior o inferior del área interna para crear una barra de herramientas o de estado.

Nota La compañía Unisys tiene una patente que pretende abarcar ciertos aspectos de la compresión GIF-LZW , que utilizan los controles **PictureBox** e **Image**. Microsoft Corporation obtuvo una licencia de la patente de Unisys LZW en Septiembre de 1996. La licencia de Microsoft, sin embargo, no se extiende a programadores de software o a terceras partes que utilicen cualquier conjunto de herramientas, lenguaje de desarrollo o productos de sistemas operativos para proporcionar la posibilidad de leer o escribir archivos GIF y/o cualquier otra característica LZW en sus propios productos (por ejemplo, mediante DLLs y APIs).

Si su aplicación comercial utiliza uno de estos controles (y, por lo tanto, la tecnología LZW), puede conseguir una opinión legal independiente acerca de los efectos de la patente o bien puede ponerse en contacto con Unisys USA en http://www.unisys.com/ para obtener más información.

PROPIEDADES DE PictureBox (Control)

Align (Propiedad)
Appearance (Propiedad)
AutoRedraw (Propiedad)
AutoSize (Propiedad)
BackColor, ForeColor (Propiedades)
BorderStyle (Propiedad)
ClipControls (Propiedad)
Container (Propiedad)
CurrentX, CurrentY (Propiedades)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
DrawMode (Propiedad)
DrawStyle (Propiedad)
DrawWidth (Propiedad)
Enabled (Propiedad)
FillColor (Propiedad)
FillStyle (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
FontTransparent (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Image (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
LinkItem (Propiedad)
LinkMode (Propiedad)
LinkTimeout (Propiedad)
LinkTopic (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
Negotiate (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)

BIBLIOTECA UNADECA ALAJUELA COSTA RICA
005.4 A656c 38-749

Picture (Propiedad)
RightToLeft (Propiedad)
ScaleHeight, ScaleWidth (Propiedades)
ScaleLeft, ScaleTop (Propiedades)
ScaleMode (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hDC (Propiedad)
hWnd (Propiedad)

METODOS DE PictureBox (Control)

Circle (Método)
Cls (Método)
Drag (Método)
Line (Método)
LinkExecute (Método)
LinkPoke (Método)
LinkRequest (Método)
LinkSend (Método)
Move (Método)
OLEDrag (Método)
PSet (Método)
PaintPicture (Método)
Point (Método)
Refresh (Método)
Scale (Método)
ScaleX, ScaleY (Métodos)
SetFocus (Método)
ShowWhatsThis (Método)
TextHeight (Método)
TextWidth (Método)
ZOrder (Método)

EVENTOS DE PictureBox (Control)

Change (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)

DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LinkClose (Evento)
LinkError (Evento)
LinkNotify (Evento)
LinkOpen (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Paint (Evento)
Resize (Evento)
Validate (Evento)

PropertyPage (Objeto)

Vea también Ejemplo Propiedades Métodos Eventos
El objeto base utilizado para crear una página de propiedades ActiveX.

Comentarios
Las páginas de propiedades proporcionan una alternativa a la ventana Propiedades para ver las propiedades. Puede agrupar en una página varias propiedades relacionadas o usar una página para proporcionar una interfaz de tipo cuadro de diálogo para una propiedad que sea demasiado compleja para la ventana Propiedades. Un objeto **PropertyPage** representa una página, es decir, una ficha del cuadro de diálogo **Páginas de propiedades**.

PROPIEDADES DE PropertyPage (Objeto)

ActiveControl (Propiedad)
Appearance (Propiedad)
AutoRedraw (Propiedad)
BackColor, ForeColor (Propiedades)
Caption (Propiedad)
Changed (Propiedad)
ClipControls (Propiedad)

Controls (Propiedad)
Count (Propiedad, colecciones de VB)
CurrentX, CurrentY (Propiedades)
DrawMode (Propiedad)
DrawStyle (Propiedad)
DrawWidth (Propiedad)
FillColor (Propiedad)
FillStyle (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
FontTransparent (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Image (Propiedad)
KeyPreview (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Palette (Propiedad)
PaletteMode (Propiedad)
Picture (Propiedad)
RightToLeft (Propiedad)
ScaleHeight, ScaleWidth (Propiedades)
ScaleLeft, ScaleTop (Propiedades)
ScaleMode (Propiedad)
SelectedControls (Propiedad)
StandardSize (Propiedad)
Tag (Propiedad)
hDC (Propiedad)
hWnd (Propiedad)

METODOS DE PropertyPage (Objeto)

Circle (Método)
Cls (Método)
Line (Método)
OLEDrag (Método)
PSet (Método)
PaintPicture (Método)
Point (Método)
PopupMenu (Método)

Refresh (Método)
Scale (Método)
ScaleX, ScaleY (Métodos)
SetFocus (Método)
TextHeight (Método)
TextWidth (Método)

EVENTOS DE PropertyPage (Objeto)
Activate, Deactivate (Eventos)
ApplyChanges (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
EditProperty (Evento)
GotFocus (Evento, objetos UserControl y UserDocument)
Initialize (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
Load (Evento)
LostFocus (Evento, objetos UserControl y UserDocument)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Paint (Evento)
SelectionChanged (Evento)
Terminate (Evento)
Unload (Evento)

RichTextBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
El control **RichTextBox** permite al usuario escribir y modificar texto al tiempo que proporciona características de formato más avanzadas que el control **TextBox** convencional.

Sintaxis
RichTextBox

Comentarios
El control **RichTextBox** proporciona una serie de propiedades que puede usar para aplicar formato a cualquier parte del texto de un control. Para cambiar el formato del texto, debe seleccionarlo previamente. Sólo puede asignar formato de carácter y de párrafo al texto seleccionado. Mediante estas propiedades puede poner el texto en negrita o en cursiva, cambiar el color, y crear superíndices y subíndices. También puede ajustar el formato de párrafo si establece sangrías izquierda y derecha, así como sangrías francesas.

El control **RichTextBox** también abre y guarda archivos con formato RTF y texto ASCII normal. Puede usar métodos del control (**LoadFile** y **SaveFile**) para leer y escribir archivos directamente o usar propiedades del control como **SelRTF** y **TextRTF** junto con las instrucciones de entrada y salida de archivos de Visual Basic.

El control **RichTextBox** admite la incrustación de objetos mediante la colección **OLEObjects**. Cada objeto insertado en el control está representado por un objeto **OLEObject**. Esto le permite crear documentos con el control que contenga otros documentos u objetos. Por ejemplo, puede crear un documento que tenga incrustada una hoja de cálculo de Microsoft Excel o un documento de Microsoft Word o cualquier otro objeto OLE que esté registrado en su sistema. Para insertar objetos en el control **RichTextBox**, simplemente arrastre un archivo (desde el Explorador de Windows 95 por ejemplo) o una parte resaltada de un archivo utilizada en otra aplicación (como Microsoft Word) y coloque el contenido directamente en el control.

El control **RichTextBox** admite el Portapapeles y permite arrastrar y colocar objetos OLE. Cuando se pega un objeto desde el Portapapeles, se inserta en el punto de inserción actual. Cuando se arrastra un objeto y se coloca en el control, el punto de inserción seguirá al cursor del *mouse* (ratón) hasta que se suelte el botón del *mouse*, haciendo que se inserte el objeto. Este comportamiento es el mismo que el de Microsoft Word.

Para imprimir todo o parte de un texto de un control **RichTextBox**, utilice el método **SelPrint**.
Puesto que **RichTextBox** es un control enlazado a datos, puede enlazarlo con un control **Data** a un campo Binary o Memo en una base de datos de Microsoft Access o un campo de parecida capacidad de otra base de datos (como el campo de tipo de datos TEXT en SQL Server).

El control **RichTextBox** admite la mayoría de las propiedades, eventos y métodos utilizados con el control **TextBox** estándar: **MaxLength**, **MultiLine**, **ScrollBars**, **SelLength**, **SelStart** y **SelText**. Las aplicaciones que ya utilizan controles **TextBox** pueden adaptarse fácilmente para usar los controles **RichTextBox**. Sin embargo, el

control **RichTextBox** no tiene el mismo límite de capacidad de 64 KB del control **TextBox** convencional.

Nota para la distribución Para usar el control **RichTextBox** en su aplicación, deberá agregar el archivo Richtx32.ocx al proyecto. Cuando distribuya la aplicación, instale el archivo Richtx32.ocx en el directorio SYSTEM de Microsoft Windows del usuario. Para obtener más información acerca de cómo agregar un control personalizado a un proyecto, vea el *Manual del programador*.

PROPIEDADES DE RichTextBox (Control)

Appearance (Propiedad, controles ActiveX)
AutoVerbMenu (Propiedad, control RichTextBox)
BackColor, ForeColor (Propiedades, controles ActiveX)
BorderStyle (Propiedad, controles ActiveX)
BulletIndent (Propiedad)
CausesValidation (Propiedad)
Container (Propiedad)
DataBindings (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
DisableNoScroll (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad, controles ActiveX)
FileName (Propiedad, control RichTextBox)
Font (Propiedad)
Height, Width (Propiedades, controles ActiveX)
HelpContextID (Propiedad)
HideSelection (Propiedad, controles ActiveX)
hWnd (Propiedad)
Index (Propiedad, controles ActiveX)
Left, Top (Propiedades, controles ActiveX)
Locked (Propiedad)
MaxLength (Propiedad, control RichTextBox)
MouseIcon (Propiedad)
MousePointer (Propiedad, controles ActiveX)
MultiLine (Propiedad, control RichTextBox)
Name (Propiedad)
Object (Propiedad, controles ActiveX)
OLEDragMode (Propiedad, controles ActiveX)
OLEDropMode (Propiedad, controles ActiveX)

OLEObjects (Propiedad)
Parent (Propiedad)
RightMargin (Propiedad)
ScrollBars (Propiedad, control RichTextBox)
SelAlignment (Propiedad)
SelBold, SelItalic, SelStrikethru, SelUnderline (Propiedades)
SelBullet (Propiedad)
SelCharOffset (Propiedad)
SelColor (Propiedad)
SelFontName (Propiedad)
SelFontSize (Propiedad)
SelHangingIndent, SelIndent, SelRightIndent (Propiedades)
SelLength, SelStart, SelText (Propiedades, controles ActiveX)
SelProtected (Propiedad)
SelRTF (Propiedad)
SelTabCount, SelTabs (Propiedades)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad, controles ActiveX)
Text (Propiedad, controles ActiveX)
TextRTF (Propiedad)
ToolTipText (Propiedad)
Top (Propiedad)
Visible (Propiedad, controles ActiveX)
WhatsThisHelpID (Propiedad)

METODOS DE RichTextBox (Control)

Drag (Método)
Find (Método)
GetLineFromChar (Método)
LoadFile (Método)
Move (Método)
OLEDrag (Método, controles ActiveX)
Refresh (Método, controles ActiveX)
SaveFile (Método)
SelPrint (Método)
SetFocus (Método)
ShowWhatsThis (Método)
Span (Método)
Upto (Método)
ZOrder (Método)

EVENTOS DE RichTextBox (Control)
Change (Evento, controles ActiveX)
Click (Evento, controles ActiveX)
DblClick (Evento, controles ActiveX)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyDown, KeyUp (Eventos, controles ActiveX)
KeyPress (Evento)
KeyPress (Evento, controles ActiveX)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento, controles ActiveX)
OLEDragDrop (Evento, controles ActiveX)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento, controles ActiveX)
OLESetData (Evento, controles ActiveX)
OLEStartDrag (Evento, controles ActiveX)
SelChange (Evento, control RichTextBox)
Validate (Evento)

TextBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **TextBox**, también denominado control de campo de edición o control de edición, muestra información introducida en tiempo de diseño, introducida por el usuario o asignada al control en código en tiempo de ejecución.

Sintaxis
TextBox

Comentarios
Para mostrar múltiples líneas de texto en un control **TextBox**, establezca la propiedad **MultiLine** a **True**. Si un **TextBox** de múltiples líneas no tiene una barra de desplazamiento horizontal, el texto se ajustará automáticamente aunque se ajuste el tamaño de **TextBox**. Para personalizar la combinación de barra de desplazamiento en un **TextBox**, establezca la propiedad **ScrollBars**.

Las barras de desplazamiento siempre aparecerán en el **TextBox** cuando su propiedad **MultiLine** se haya establecido a **True** y su propiedad **ScrollBars** a cualquier cosa excepto **None** (0).

Si establece la propiedad **MultiLine** a **True**, podrá emplear la propiedad **Alignment** para establecer la alineación del texto dentro del **TextBox**. El texto está justificado a la izquierda de forma predeterminada. Si la propiedad **MultiLine** es **False**, el valor de la propiedad **Alignment** no tiene ningún efecto.
Un control **TextBox** también puede actuar como vínculo de destino en una conversación DDE.

PROPIEDADES DE TextBox (Control)

Alignment (Propiedad)
Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
BorderStyle (Propiedad)
Container (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
HideSelection (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
LinkItem (Propiedad)
LinkMode (Propiedad)
LinkTimeout (Propiedad)
LinkTopic (Propiedad)
Locked (Propiedad)
MaxLength (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
MultiLine (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
OLEDropMode (Propiedad, controles ActiveX)
Parent (Propiedad)

PasswordChar (Propiedad)
RightToLeft (Propiedad)
ScrollBars (Propiedad)
SelLength, SelStart, SelText (Propiedades)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
Text (Propiedad)
ToolTipText (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE TextBox (Control)

Drag (Método)
LinkExecute (Método)
LinkPoke (Método)
LinkRequest (Método)
LinkSend (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE TextBox (Control)

Change (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LinkClose (Evento)
LinkError (Evento)
LinkNotify (Evento)
LinkOpen (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLECompleteDrag (Evento, controles ActiveX)

OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Validate (Evento)

UserControl (Objeto)

Vea también Ejemplo Propiedades Métodos Eventos
El objeto **UserControl** es el objeto base utilizado para crear un control ActiveX.

Comentarios
Un control ActiveX creado en Visual Basic siempre se compone de un objeto **UserControl**, más otros controles, conocidos como controles constituyentes, que decida colocar en el objeto

UserControl.
De igual forma que los formularios Visual Basic, los objetos **UserControl** tienen módulos de código y diseñadores visuales. Coloque los controles constituyentes en el diseñador del objeto **UserControl**, de la misma manera que coloca controles en un formulario.

PROPIEDADES DE UserControl (Objeto)
AccessKeys (Propiedad)
ActiveControl (Propiedad)
Ambient (Propiedad)
Appearance (Propiedad)
AutoRedraw (Propiedad)
BackColor, ForeColor (Propiedades)
BackStyle (Propiedad, objeto UserControl)
ClipBehavior (Propiedad)
ClipControls (Propiedad)
ContainedControls (Propiedad)
ContainerHWnd (Propiedad)
Controls (Propiedad)
Count (Propiedad, colecciones de VB)
CurrentX, CurrentY (Propiedades)
DataMembers (Propiedad)
DrawMode (Propiedad)
DrawStyle (Propiedad)
DrawWidth (Propiedad)
EditAtDesignTime (Propiedad)

Enabled (Propiedad)
EventsFrozen (Propiedad)
Extender (Propiedad)
FillColor (Propiedad)
FillStyle (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
FontTransparent (Propiedad)
HasDC (Propiedad)
Height, Width (Propiedades)
HitBehavior (Propiedad)
Hyperlink (Propiedad)
Image (Propiedad)
KeyPreview (Propiedad)
MaskColor (Propiedad, Objeto UserControl)
MaskPicture (Propiedad, Objeto UserControl)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Palette (Propiedad)
PaletteMode (Propiedad)
Parent (Propiedad, Objeto UserControl)
ParentControls (Propiedad)
ParentControlsType (Propiedad)
Picture (Propiedad)
PropertyPages (Propiedad)
RightToLeft (Propiedad)
ScaleHeight, ScaleWidth (Propiedades)
ScaleLeft, ScaleTop (Propiedades)
ScaleMode (Propiedad)
Tag (Propiedad)
hDC (Propiedad)
hWnd (Propiedad)

METODOS DE UserControl (Objeto)

AsyncRead (Método)
CanPropertyChange (Método)
CancelAsyncRead (Método)
Circle (Método)
Cls (Método)
Line (Método)

OLEDrag (Método)
PSet (Método)
PaintPicture (Método)
Point (Método)
PopupMenu (Método)
PropertyChanged (Método)
Refresh (Método)
Scale (Método)
ScaleX, ScaleY (Métodos)
SetFocus (Método)
Size (Método)
TextHeight (Método)
TextWidth (Método)

EVENTOS DE UserControl (Objeto)

AccessKeyPress (Evento)
AmbientChanged (Evento)
AsyncReadComplete (Evento)
AsyncReadProgress (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
EnterFocus (Evento)
ExitFocus (Evento)
GetDataMember (Evento)
GotFocus (Evento, objetos UserControl y UserDocument)
Hide (Evento, objeto UserControl)
HitTest (Evento)
InitProperties (Evento)
Initialize (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento, objetos UserControl y UserDocument)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Paint (Evento)

ReadProperties (Evento)
Resize (Evento)
Show (Evento, objeto UserControl)
Terminate (Evento)
WriteProperties (Evento)

UserDocument (Objeto)

Vea también Ejemplo Propiedades Métodos Eventos
El objeto base de un documento ActiveX, el objeto **UserDocument** recuerda al objeto **Form** estándar de Visual Basic, con algunas excepciones.

Comentarios
El objeto **UserDocument** tiene la mayor parte de los eventos del objeto **Form**, pero no todos. Los eventos presentes en los objetos **Form** que no se encuentran en los objetos **UserDocument** son: Activate, Deactivate, LinkClose, LinkError, LinkExecute, LinkOpen, Load, QueryUnload y Unload.

Los eventos presentes en los objetos **UserDocument**, pero que no se encuentran en los objetos **Form** son: AsyncReadComplete, EnterFocus, ExitFocus, Hide, InitProperties, ReadProperties, Scroll, Show y WriteProperties.

No puede colocar objetos incrustados (como un documento de Excel o de Word) o un control contenedor ActiveX en un objeto **UserDocument**.

PROPIEDADES DE UserDocument (Objeto)

ActiveControl (Propiedad)
Appearance (Propiedad)
Appearance (Propiedad, controles ActiveX)
AutoRedraw (Propiedad)
BackColor, ForeColor (Propiedades)
ClipControls (Propiedad)
ContainedControls (Propiedad)
ContinuousScroll (Propiedad)
Controls (Propiedad)
Count (Propiedad, colecciones de VB)
CurrentX, CurrentY (Propiedades)
DrawMode (Propiedad)
DrawStyle (Propiedad)
DrawWidth (Propiedad)
Ejemplo de las propiedades ScaleHeight, ScaleWidth
FillColor (Propiedad)

FillStyle (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
FontTransparent (Propiedad)
HScrollSmallChange, VScrollSmallChange (Propiedades)
Height, Width (Propiedades)
Hyperlink (Propiedad)
Image (Propiedad)
KeyPreview (Propiedad)
MinHeight, MinWidth (Propiedades)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
OLEDropMode (Propiedad)
Palette (Propiedad)
PaletteMode (Propiedad)
Parent (Propiedad)
Picture (Propiedad)
RightToLeft (Propiedad)
ScaleLeft, ScaleTop (Propiedades)
ScaleMode (Propiedad)
ScrollBars (Propiedad)
Tag (Propiedad)
ViewportHeight, ViewportLeft, ViewportTop, ViewportWidth (Propiedades)
hDC (Propiedad)
hWnd (Propiedad)

METODOS DE UserDocument (Objeto)

AsyncRead (Método)
CancelAsyncRead (Método)
Circle (Método)
Cls (Método)
Line (Método)
OLEDrag (Método)
PSet (Método)
PaintPicture (Método)
Point (Método)
PopupMenu (Método)
PrintForm (Método)
PropertyChanged (Método)

Refresh (Método)
Scale (Método)
ScaleX, ScaleY (Métodos)
SetFocus (Método)
SetViewport (Método)
TextHeight (Método)
TextWidth (Método)

EVENTOS DE UserDocument (Objeto)

AsyncReadComplete (Evento)
AsyncReadProgress (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
EnterFocus (Evento)
ExitFocus (Evento)
GotFocus (Evento, objetos UserControl y UserDocument)
Hide (Evento, objeto UserControl)
InitProperties (Evento)
Initialize (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento, objetos UserControl y UserDocument)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Paint (Evento)
ReadProperties (Evento)
Resize (Evento)
Scroll (Evento)
Show (Evento, objeto UserControl)
Terminate (Evento)
WriteProperties (Evento)

EVENTOS

Activate, Deactivate (Eventos)

Vea también Ejemplo Se aplica a

- Activate: ocurre cuando un objeto se convierte en la ventana activa.
- Deactivate: ocurre cuando un objeto deja de ser la ventana activa.

Sintaxis
Private Sub *objeto*_**Activate( )**
Private Sub *objeto*_**Deactivate( )**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
Un objeto puede convertirse en activo mediante una acción del usuario, como hacer clic en él, o mediante los métodos **Show** o **SetFocus** en código.

El evento Activate sólo puede ocurrir cuando un objeto es visible. Por ejemplo, un formulario cargado con la instrucción **Load** no es visible a menos que utilice el método **Show** o establezca la propiedad **Visible** del formulario a **True**.

Los eventos Activate y Deactivate sólo ocurren al mover el enfoque dentro de una aplicación. Mover el enfoque hacia o desde un objeto de otra aplicación no desencadena ningún evento. El evento Deactivate no ocurre al descargar un objeto.

El evento Activate ocurre antes que el evento GotFocus y el evento LostFocus ocurre antes que el evento Deactivate.

Estos eventos se producen en formularios secundarios MDI sólo cuando el enfoque cambia de un formulario secundario a otro. En un objeto **MDIForm** con dos formularios secundarios, por ejemplo, los formularios secundarios reciben estos eventos cuando el enfoque se mueve entre ellos. Sin embargo, cuando el enfoque cambia entre un formulario secundario y un formulario secundario que no es MDI, el **MDIForm** primario recibe los eventos Activate y Deactivate.
Si un archivo .exe creado con Visual Basic muestra un cuadro de diálogo de un archivo .dll creado también con Visual Basic, el formulario del archivo .exe obtendrá los eventos Deactivate y LostFocus. Esto puede ser inesperado, ya que no debe obtener el evento Deactivate:

- Si el objeto es un componente fuera de proceso.
- Si el objeto no está escrito en Visual Basic.
- En el entorno de desarrollo, al llamar a un archivo DLL generado en Visual Basic.

SE APLICA A:

ÑDataReport (Objeto)
Form (Objeto), Forms (Colección)
MDIForm (Objeto)

Ejemplo de los eventos Activate y Deactivate

Este ejemplo actualiza el texto de la barra de estado para mostrar el título del formulario activo. Para probar este ejemplo, cree un objeto **Form** (Form1) y un nuevo objeto **MDIForm** (MDIForm1). En MDIForm1, dibuje un control **PictureBox** que contenga un control **Label**. En Form1, establezca la propiedad **MDIChild** a **True**. Pegue el código del procedimiento de evento MDIForm_Load en la sección Declaraciones del objeto **MDIForm**. Pegue el código del procedimiento de evento Form_Activate en la sección Declaraciones del formulario secundario MDI y después presione F5.

```
Private Sub MDIForm_Load ()
   Form1.Caption = "Formulario n°1"    ' Establece el título
de Form1.
   Dim NewForm As New Form1    ' Crea un nuevo formulario
secundario.
   Load NewForm
   NewForm.Caption = "Formulario n°2"    ' Establece el
título del nuevo
' formulario.
   NewForm.Show    ' Muestra el nuevo formulario.
End Sub

Private Sub Form_Activate ()
      ' Establece el texto de la barra de estado.
   MDIForm1.Label1.Caption = "Formulario actual: " &
Me.Caption
End Sub
```

Change (Evento)

Vea también Ejemplo Se aplica a

Indica que el contenido de un control ha cambiado. Cómo y cuándo ha ocurrido este evento varía según el control:

- **ComboBox**: cambia el texto de la parte de cuadro de texto del control. Ocurre sólo si la propiedad **Style** está establecida a 0 (Dropdown Combo) o 1 (Simple Combo) y el usuario cambia el texto o usted cambia la configuración de la propiedad **Text** mediante código.

- **DirListBox**: cambia el directorio seleccionado. Ocurre cuando el usuario hace doble clic en un nuevo directorio o cuando usted cambia la configuración de la propiedad **Path** mediante código.

- **DriveListBox**: cambia la unidad seleccionada. Ocurre cuando el usuario selecciona una nueva unidad o cuando usted cambia la configuración de la propiedad **Drive** mediante código.

- **HScrollBar** y **VScrollBar** (barras de desplazamiento horizontal y vertical): mueven la parte de cuadro de desplazamiento de la barra de desplazamiento. Ocurre cuando el usuario desplaza o cuando usted cambia la configuración de la propiedad **Value** mediante código.

- **Label**: cambia el contenido del control **Label**. Ocurre cuando un vínculo DDE actualiza los datos o cuando usted cambia la configuración de la propiedad **Caption** mediante código.

- **PictureBox**: cambia el contenido del control **PictureBox**. Ocurre cuando un vínculo DDE actualiza los datos o cuando usted cambia la configuración de la propiedad **Picture** mediante código.

- **TextBox**: cambia el contenido del cuadro de texto. Ocurre cuando un vínculo DDE actualiza los datos, cuando un usuario cambia el texto o cuando usted cambia la configuración de la propiedad **Text** mediante código.

Sintaxis
Private Sub *objeto*_**Change**([*índice* **As Integer**])
La sintaxis del evento Change consta de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica únicamente a un control si está en una matriz de controles.

Comentarios
El procedimiento del evento Change puede sincronizar o coordinar la presentación de datos entre controles. Por ejemplo, puede usar un procedimiento de evento Change de una barra de desplazamiento para actualizar la configuración de la propiedad **Value** de la barra de desplazamiento de un control **TextBox**. O bien, puede usar un procedimiento de evento Change para mostrar datos y fórmulas en un área de trabajo y los resultados en otra área.

Los procedimientos de evento Change son también útiles para actualizar propiedades de controles del sistema de archivos (**DirListBox**, **DriveListBox** y **FileListBox**). Por ejemplo, puede actualizar la configuración de la propiedad **Path** para que un control

DirListBox refleje un cambio en la configuración de la propiedad **Drive** de un control **DriveListBox**.

Nota Un procedimiento de evento Change puede algunas veces causar un evento en cascada. Esto ocurre cuando el procedimiento de evento Change del control altera el contenido del control, por ejemplo, estableciendo una propiedad en el código que determina el valor del control, como el valor de la propiedad **Text** para un control **TextBox**. Para impedir un evento en cascada:

- Si es posible, evite escribir un procedimiento de evento Change para un control que modifique el contenido de dicho control. Si escribe un procedimiento así, asegúrese de establecer un indicador que impida más cambios mientras el cambio actual está en marcha.

- Evite crear dos o más controles cuyos procedimientos de evento Change se afecten entre sí, por ejemplo, dos controles **TextBox** que se actualicen mutuamente durante sus eventos Change.

- Evite el uso de una función o instrucción **MsgBox** en este evento para los controles **HScrollBar** y **VScrollBar**.

SE APLICA A:

ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
HScrollBar, VScrollBar (Controles)
Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo del evento Change

Este ejemplo muestra la configuración numérica de la propiedad **Value** de una barra de desplazamiento horizontal en un control **TextBox**. Para probar este ejemplo, cree un formulario con un control **TextBox** y un control **HScrollBar** y después pegue el código en la sección Declaraciones de un formulario que contenga una barra de desplazamiento horizontal (control **HScrollBar**) y un control **TextBox**. Presione F5 y haga clic en la barra de desplazamiento horizontal.

```
Private Sub Form_Load ()
   HScroll1.Min = 0                  ' Establece Min.
   HScroll1.Max = 1000               ' Establece Max.
   HScroll1.LargeChange = 100     ' Establece LargeChange.
```

```
   HScroll1.SmallChange = 1        ' Establece SmallChange.
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = HScroll1.Value
End Sub
```

Change (Evento, controles ActiveX)

Vea también Ejemplo Se aplica a

Indica que el contenido de un control ha cambiado. Cómo y cuando ha ocurrido este evento varía con el control:

- **ComboBox**: cambia el texto de la parte de cuadro de texto del control. Ocurre sólo si la propiedad **Style** está establecida a 0 (Dropdown Combo) o 1 (Simple Combo) y el usuario cambia el texto o usted cambia el valor de la propiedad **Text** mediante código.

- **DirListBox**: cambia el directorio seleccionado. Ocurre cuando el usuario hace doble clic en un nuevo directorio o cuando usted cambia el valor de la propiedad **Path** mediante código.

- **DriveListBox**: cambia la unidad seleccionada. Ocurre cuando el usuario selecciona una nueva unidad o cuando usted cambia el valor de la propiedad **Drive** mediante código.

- **HScrollBar** y **VScrollBar** (barras de desplazamiento horizontal y vertical): mueven la parte de cuadro de desplazamiento de la barra de desplazamiento. Ocurre cuando el usuario se desplaza o cuando usted cambia el valor de la propiedad **Value** mediante código.

- **Label**: cambia el contenido del control **Label**. Ocurre cuando un vínculo DDE actualiza los datos o cuando usted cambia el valor de la propiedad **Caption** mediante código.

- **PictureBox**: cambia el contenido del **PictureBox**. Ocurre cuando un vínculo DDE actualiza los datos o cuando usted cambia el valor de la propiedad **Picture** mediante código.

- **TextBox**: cambia el contenido del cuadro de texto. Ocurre cuando un vínculo DDE actualiza los datos, cuando un usuario cambia el texto o cuando usted cambia el valor de la propiedad **Text** mediante código.

Sintaxis

Private Sub *objeto*_**Change**([*índice* **As Integer**])

La sintaxis del evento Change consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

índice Un entero que identifica unívocamente un control si está en una matriz de controles.

Comentarios

El procedimiento de evento Change puede sincronizar o coordinar la presentación de datos entre controles. Por ejemplo, puede usar el procedimiento de evento Change de una barra de desplazamiento para actualizar el valor de la propiedad **Value** de la barra de desplazamiento de un control **TextBox**. O bien, puede usar un procedimiento de evento Change para mostrar datos y fórmulas en un área de trabajo y los resultados en otra área distinta.

Los procedimientos de evento Change son también útiles para actualizar propiedades de controles de sistema de archivos (**DirListBox**, **DriveListBox** y **FileListBox**). Por ejemplo, puede actualizar el valor de la propiedad **Path** para que un control **DirListBox** refleje un cambio en el valor de la propiedad **Drive** de un control **DriveListBox**.

Nota Un procedimiento de evento Change puede algunas veces causar un evento en cascada. Esto ocurre cuando el procedimiento de evento Change altera el contenido del control, por ejemplo, estableciendo una propiedad en el código que determina el valor del control, como el valor de la propiedad **Text** para un control **TextBox**. Para impedir un evento en cascada:

- Si es posible, evite escribir un procedimiento de evento Change para un control que cambie el contenido de dicho control. Si lo escribre como un procedimiento, compruebe que establece un indicador que evite cambios posteriores mientras se están realizando los cambios.

- Evite crear dos o más controles cuyos procedimientos de evento Change afecten unos a otros, por ejemplo, dos controles **TextBox** que se actualizan mutuamente durante los eventos Change.

- Evite usar una función o una instrucción **MsgBox** en este evento para los controles **HScrollBar** y **VScrollBar**.

SE APLICA A:

Masked Edit (Control)
RichTextBox (Control)
TextBox (Control Ligero)

Click (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario presiona y suelta un botón del *mouse* (ratón) en un objeto. También puede ocurrir cuando se cambia el valor de un control.

Para un objeto **Form**, este evento ocurre cuando el usuario hace clic en un área en blanco o en un control desactivado. Para un control, este evento ocurre cuando el usuario:

- Hace clic en un control con el botón primario o secundario del *mouse*. Con un control **CheckBox**, **CommandButton**, **ListBox** o **OptionButton**, el evento Click sólo ocurre cuando el usuario hace clic con el botón primario del *mouse*.

- Selecciona un elemento de un control **ComboBox** o **ListBox**, ya sea presionando las teclas de dirección o haciendo clic con el botón del *mouse*.

- Presiona la BARRA ESPACIADORA cuando un control **CommandButton**, **OptionButton** o **CheckBox** tiene el enfoque.

- Presiona ENTRAR cuando un formulario tiene un control **CommandButton** con su propiedad **Default** establecida a **True**.

- Presiona ESC cuando un formulario tiene un botón **Cancelar**, un control **CommandButton** con su propiedad **Cancel** establecida a **True**.

- Presiona una tecla de acceso para un control. Por ejemplo, si el título de un control **CommandButton** es "&Ir", al presionar Alt-I se desencadena este evento.

También puede desencadenar el evento Click en el código si:

- Establece la propiedad **Value** de un control **CommandButton** a **True**.

- Establece la propiedad **Value** de un control **OptionButton** a **True**.

- Cambia el valor de la propiedad **Value** de un control **CheckBox**.

Sintaxis

Private Sub Form_Click( )

Private Sub *objeto*_**Click([***índice* **As Integer])**

La sintaxis del evento Click consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica únicamente a un control si está en una matriz de controles.

Comentarios

Por lo general se adjunta un procedimiento de evento Click a un control **CommandButton**, un objeto **Menu** o un control **PictureBox** para realizar comandos y acciones similares a comandos. Para los demás controles aplicables, utilice este evento para desencadenar acciones como respuesta a un cambio en el control.

Puede usar la propiedad **Value** de un control para comprobar el estado del control desde el código. Hacer clic en un control genera los eventos MouseDown y MouseUp además del evento Click. El orden en que ocurren estos tres eventos varía de un control a otro. Por ejemplo, para los controles **ListBox** y **CommandButton**, los eventos ocurren en este orden: MouseDown, Click, MouseUp. Pero para los controles **FileListBox**, **Label** o **PictureBox**, los eventos ocurren en este otro orden: MouseDown, MouseUp y Click. Cuando está adjuntando procedimientos para estos eventos relacionados, asegúrese de que sus acciones no entran en conflicto. Si el orden de los eventos es importante en la aplicación, pruebe el control para determinar el orden de los mismos.

Nota Para distinguir entre los botones primario, secundario y central del mouse, utilice los eventos MouseDown y MouseUp.

Si hay código en el evento Click, nunca se activará el evento DblClick ya que de los dos eventos, Click es el primero que se activa. Como resultado, el evento Click intercepta el clic del mouse, por lo que DblClick nunca se producirá.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
DirListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)

Ejemplo del evento Click

En este ejemplo, cada vez que se hace clic en un control **PictureBox** se mueve diagonalmente por un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **PictureBox** en la esquina inferior izquierda del mismo, después presione F5 y haga clic en el control **PictureBox**.

```
Private Sub Picture1_Click ()
   Picture1.Move Picture1.Left + 750, Picture1.Top - 550
End Sub
```

Click (Evento, controles ActiveX)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario presiona y suelta un botón del *mouse* (ratón) sobre un objeto. También puede ocurrir cuando se cambia el valor de un control.

Para un objeto **Form**, este evento ocurre cuando el usuario hace clic en un área en blanco o en un control desactivado. Para un control, este evento ocurre cuando el usuario:

- Hace clic en un control con el botón primario o secundario del *mouse*. Con un control **CheckBox**, **CommandButton**, **Listbox** u **OptionButton**, el evento Click ocurre sólo cuando el usuario hace clic con el botón primario del *mouse*.

- Selecciona un elemento de un control **ComboBox** o **ListBox**, ya sea presionando las teclas de dirección o haciendo clic con el botón del *mouse*.

- Presiona la BARRA ESPACIADORA cuando un control **CommandButton**, **OptionButton** o **CheckBox** tiene el enfoque.

- Presiona la tecla ENTRAR cuando un formulario tiene un control **CommandButton** con su propiedad **Default** establecida a **True**.

- Presiona ESC cuando un formulario tiene un botón **Cancelar**; un control **CommandButton** con su propiedad **Cancel** establecida a **True**.

- Presiona una tecla de acceso para un control. Por ejemplo, si el título de un control **CommandButton** es "&Ir a", presionar Alt-I desencadena este evento.

También puede desencadenar el evento Click en el código si:

- Establece la propiedad **Value** de un control **CommandButton** a **True**.

- Establece la propiedad **Value** de un control **OptionButton** a **True**.

- Cambia el valor de la propiedad **Value** de un control **CheckBox**.

Sintaxis

Private Sub Form_Click( )

Private Sub *objeto*_**Click(**[*índice* **As Integer**]**)**

La sintaxis del evento Click consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica unívocamente un control si está en una matriz de controles.

Comentarios

Por lo general, adjunta un procedimiento de evento Click a un control **CommandButton**, un objeto **Menu** o un control **PictureBox** para realizar comandos o acciones similares a comandos. Para los otros controles aplicables, utilice este evento para desencadenar acciones como respuesta a un cambio en el control.

Puede usar la propiedad **Value** de un control para comprobar el estado del control desde el código. Hacer clic en un control genera los eventos MouseDown y MouseUp además del evento Click. El orden en que ocurren estos tres eventos varía de un control a otro. Por ejemplo, para los controles **ListBox** y **CommandButton**, los eventos ocurren en este orden:

MouseDown, Click, MouseUp. Pero para los controles **FileListBox**, **Label** o **PictureBox**, los eventos ocurren en este otro: MouseDown, MouseUp, Click. Cuando está adjuntando procedimientos de evento, asegúrese de que sus acciones no entran en conflicto. Si el orden de los eventos es importante en su aplicación, compruebe el control para determinar el orden de los mismos.

Nota Para distinguir entre los botones izquierdo, derecho y central del *mouse*, utilice los eventos MouseDown y MouseUp.

Si hay código en el evento Click, el evento DblClick nunca se desencadenará, ya que el evento Click es el primero que se desencadena de los dos. Como resultado, el evento Click interceptará el clic con el *mouse*, por lo que el evento DblClick no ocurrirá.

SE APLICA A:

ACoolBar (Control)
DataRepeater (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
RichTextBox (Control)
TextBox (Control Ligero)

DblClick (Evento)

Vea también Ejemplo Se aplica a

Se produce cuando un usuario presiona y suelta un botón del *mouse* (ratón) dos veces sobre un objeto.

En un formulario, el evento DblClick se produce cuando el usuario hace doble clic en un control deshabilitado o en un área en blanco de un formulario. En un control, se produce cuando el usuario:

- Hace doble clic en un control con el botón primario del *mouse*.
- Hace doble clic en un elemento de un control **ComboBox** cuya propiedad **Style** es 1 (Simple) o en un control **FileListBox**, **ListBox**, **DataCombo** o **DataList**.

Sintaxis

Private Sub Form_DblClick ( )

Private Sub *objeto*_**DblClick (***índice* **As Integer)**

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Identifica el control si está dentro de una matriz de controles.

Comentarios

El argumento *índice* identifica de forma única un control si pertenece a una matriz de controles. Puede usar el procedimiento de evento DblClick para acciones implícitas, como hacer doble clic en un icono para abrir una ventana o un documento. También puede usar este tipo de procedimiento para realizar varios pasos con una única acción, como hacer doble clic para seleccionar un elemento de un cuadro de lista y cerrar el cuadro de diálogo.

Para conseguir estos efectos en Visual Basic puede usar el procedimiento de evento DblClick para un cuadro de lista o un cuadro de lista de archivos junto con un botón predeterminado (un control **CommandButton** con su propiedad **Default** establecida a **True**). Como parte del procedimiento de evento DblClick del cuadro de lista, simplemente se llama al evento Click del botón predeterminado.

En aquellos objetos que reciben eventos del Mouse, los eventos se producen en este orden: MouseDown, MouseUp, Click, DblClick y MouseUp.

Si DblClick no se produce dentro del límite de tiempo de doble clic del sistema, el objeto reconoce otro evento Click. El límite de tiempo de doble clic puede variar porque el usuario puede establecer la velocidad de doble clic en el Panel de control. Cuando escribe procedimientos para estos eventos relacionados, debe asegurarse de que sus acciones no entren en conflicto. Los controles que no reciben eventos DblClick pueden recibir dos clics en lugar de un doble clic.

Al depurar eventos, no use instrucciones MsgBox para indicar que ha ocurrido el evento, ya que interferirá en el funcionamiento normal de muchos eventos. (Por ejemplo, un MsgBox en el evento Click evitará que ocurra DblClick). En su lugar, use Debug.Print para mostrar el orden en el que ocurrirán los eventos.

Nota Para distinguir entre los botones primario, secundario y central, use los eventos MouseDown y MouseUp.

SE APLICA A:

ComboBox (Control)
Contenedor OLE (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento DblClick

Este ejemplo presenta un elemento seleccionado de una lista en un control **TextBox** cuando se hace clic en un control **CommandButton** o se hace doble clic en un elemento de la lista. Para probar este ejemplo, pegue el código en la sección Declaraciones de un objeto **Form** que contenga un control **ListBox**, un control **TextBox** y un control **CommandButton**. Después ejecute el ejemplo y haga clic en el control **CommandButton** o haga doble clic en un elemento del control **ListBox**.

```
Private Sub Form_Load ()
   List1.AddItem "John"    ' Agrega elementos a la lista.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub
```

```
Private Sub List1_DblClick ()
   Command1.Value = True    ' Provoca el evento Click.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text    ' Presenta la selección.
End Sub
```

DblClick (Evento, controles ActiveX)

Vea también Ejemplo Se aplica a

Se produce cuando un usuario presiona y suelta un botón del *mouse* dos veces sobre un objeto.

En un formulario, el evento DblClick se produce cuando el usuario hace doble clic en un control deshabilitado o en un área vacía de un formulario. En un control, se produce cuando el usuario:

- Hace doble clic en un control con el botón primario del *mouse*.
- Hace doble clic en un elemento de un control **ComboBox** cuya propiedad **Style** es 1 (Simple) o en un control **FileListBox**, **ListBox**, **DBCombo** o **DBList**.

Sintaxis

Private Sub Form_DblClick ( )

Private Sub *objeto*_**DblClick (***índice* **As Integer)**

La sintaxis del evento DblClick consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Identifica un control concreto dentro de una matriz de controles.

Comentarios

El argumento *índice* identifica de forma única un control si pertenece a una matriz de controles. El procedimiento de evento DblClick se puede usar para acciones implícitas, como hacer doble clic en un icono para abrir una ventana o un documento. También puede usar este tipo de procedimiento para ejecutar varios pasos con una única acción, como hacer doble clic para seleccionar un elemento de un cuadro de lista y cerrar el cuadro de diálogo.

Para conseguir estos efectos en Visual Basic, puede usar el procedimiento de evento DblClick de un cuadro de lista o un cuadro de lista de archivo junto con un botón predeterminado: un control **CommandButton** con su propiedad **Default** establecida a **True**. Como parte del procedimiento de evento DblClick del cuadro de lista, simplemente se llama al evento Click del botón predeterminado.

En aquellos objetos que reciban eventos del *mouse*, los eventos se producen en este orden: MouseDown, MouseUp, Click, DblClick y MouseUp.

Si DblClick no se produce dentro del límite de tiempo de doble clic del sistema, el objeto reconoce otro evento Click. El límite de tiempo para el doble clic puede variar porque el usuario puede definir la velocidad del doble clic en el Panel de control. Cuando escribe procedimientos para estos eventos relacionados, se debe asegurar que sus acciones no entren en conflicto. Los controles que no reciban eventos DblClick pueden recibir dos clics en lugar de un doble clic.

Nota Para distinguir entre los botones izquierdo, derecho y central, use los eventos MouseDown y MouseUp.
Si hay código en el evento Click, el evento DblClick nunca se desencadenará.

SE APLICA A:

CoolBar (Control)
DataRepeater (Control)
MSChart (Control)
RichTextBox (Control)
TextBox (Control Ligero)

Ejemplo del evento DblClick (Controles ActiveX)

Este ejemplo presenta un elemento seleccionado de una lista en un control **TextBox** cuando se hace clic en un control **CommandButton** o se hace doble clic en un elemento de la lista. Para ejecutar este ejemplo, pegue el código en la sección Declaraciones de un objeto **Form** que contenga un control **ListBox**, un control **TextBox** y un control **CommandButton**. Después, ejecute el ejemplo y haga clic en el control **CommandButton** o haga doble clic en un elemento del control **ListBox**.

```
Private Sub Form_Load ()
   List1.AddItem "John"    ' Agrega elementos a la lista.
   List1.AddItem "Paul"
   List1.AddItem "George"
   List1.AddItem "Ringo"
End Sub

Private Sub List1_DblClick ()
   Command1.Value = True    ' Desencadena el evento Click.
End Sub

Private Sub Command1_Click ()
   Text1.Text = List1.Text    ' Presenta la selección.
End Sub
```

Compare (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando la propiedad **Sort** del control **MSHFlexGrid** está establecida en Custom Sort (9) para permitir que el usuario personalice el proceso de orden.

Sintaxis

Private Sub *objeto*_**Compare**(*fila1*, *fila2*, *cmp*)

La sintaxis del evento Compare consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
fila1	Un entero de tipo Long que especifica la primera fila de una pareja de filas que deben compararse.
fila2	Un entero de tipo Long que especifica la segunda fila de una pareja de filas que deben compararse.
cmp	Un entero que representa el orden de cada pareja, como se describe en Valores.

Valores

El controlador de evento debe comparar *fila1* y *fila2* y establecer *cmp* en los valores siguientes:

Valor	Descripción
–1	Si *fila1* debe aparecer *antes de fila2*.
0	Si ambas filas son iguales o cualquiera de las dos puede aparecer antes de la otra.
1	Si *fila1* debe aparecer *después de fila2*.

Comentarios

Cuando la propiedad **Sort** está establecida en 9 (Custom Sort), el evento Compare ocurre una vez por cada pareja de filas incluidas en el control **MSHFlexGrid**. Debido a que el evento Compare utiliza números de filas en vez de valores de texto, puede comparar cualquier valor de propiedad de esa fila, incluido el valor de **RowData**.

Nota Aunque las ordenaciones personalizadas son más lentas que las integradas, ofrecen flexibilidad al permitir ordenar una fila mediante cualquier columna o cualquier propiedad de celda.

SE APLICA A:
MSFlexGrid (Control)
MSHFlexGrid (Control)

DragDrop (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando se completa una operación de arrastrar y colocar como resultado de arrastrar un control sobre un objeto y soltar el botón del *mouse* o usar el método **Drag** con su argumento *acción* establecido a 2 (Drop).

Sintaxis

Private Sub Form_DragDrop(*origen* **As Control,** *x* **As Single,** *y* **As Single)**
Private Sub MDIForm_DragDrop(*origen* **As Control,** *x* **As Single,** *y* **As Single)**
Private Sub *objeto***_DragDrop([***índice* **As Integer,]***origen* **As Control,** *x* **As Single,** *y* **As Single)**

La sintaxis del evento DragDrop consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de forma única a un control si está en una matriz de controles.
origen	El control que se está arrastrando. En el procedimiento de evento puede incluir propiedades y métodos con este argumento; por ejemplo, `Source.Visible = 0`.
x, y	Un número que especifica la posición horizontal (*x*) y vertical (*y*) actual del puntero del *mouse* dentro del formulario o control de destino. Estas coordenadas se expresan siempre en términos del sistema de coordenadas de destino, tal y como esté establecido en las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop**.

Comentarios

Utilice un procedimiento de evento DragDrop para controlar qué ocurre tras completarse una operación de arrastrar. Por ejemplo, puede mover el control de origen a una nueva ubicación o copiar un archivo de una ubicación a otra.

Cuando varios controles puedan usarse potencialmente en un argumento *origen*:

- Utilice la palabra clave **TypeOf** con la instrucción **If** para determinar el tipo de control utilizado con *origen.*

- Utilice la propiedad **Tag** del control para identificar un control y después utilice un procedimiento de evento DragDrop.

Nota Utilice la propiedad **DragMode** y el método **Drag** para especificar la manera en que se inicia el arrastre. Una vez iniciado el arrastre, puede controlar los eventos que preceden a un evento DragDrop con un procedimiento de evento DragOver.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento DragDrop

Este ejemplo muestra el efecto visual de colocar un control **PictureBox** sobre otro control **PictureBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga tres controles **PictureBox**. Establezca la propiedad **DragMode** para Picture1 y Picture2 a 1 (Automatic). Utilice la propiedad **Picture** para asignar mapas de bits a Picture1 y Picture2, y, después, presione F5 y arrastre Picture1 o Picture2 sobre Picture3.

```
Private Sub Picture3_DragDrop (Source As Control, X as
Single, Y As Single)
   If TypeOf Source Is PictureBox Then
      ' Establece el mapa de bits Picture3 al mismo que el
control de origen.
      Picture3.Picture = Source.Picture
   End If
End Sub
```

DragOver (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando una operación de arrastrar y colocar está en curso. Puede usar este evento para controlar el puntero del *mouse* a medida que entra, sale o descansa directamente sobre un destino válido. La posición del puntero del *mouse* determina el objeto de destino que recibe este evento.

Sintaxis

Private Sub Form_DragOver(*origen* As Control, *x* As Single, *y* As Single, *estado* As Integer)

Private Sub MDIForm_DragOver(*origen* As Control, *x* As Single, *y* As Single, *estado* As Integer)

Private Sub *objeto*_DragOver([*índice* As Integer,]*origen* As Control, *x* As Single, *y* As Single, *estado* As Integer)

La sintaxis del evento DragOver consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.
origen	El control que se está arrastrando. En el procedimiento de evento puede hacer referencia a las propiedades y métodos con este argumento; por ejemplo, `Source.Visible = False`.
x, *y*	Un número que especifica la posición horizontal (*x*) y vertical (*y*) actual del puntero del *mouse* dentro del control o el formulario de destino. Estas coordenadas se expresan siempre en términos del sistema de coordenadas del destino tal y como se establece en las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop**.

estado Un entero que corresponde al estado de transición del control que se está arrastrando en relación al formulario o control de destino:

0 = Entrar (el control de origen se está arrastrando dentro del intervalo de un destino).
1 = Salir (el control de origen se está arrastrando fuera del intervalo de un destino).
2 = Sobre (el control de origen se ha movido de una posición del destino a otra).

Comentarios
Utilice un procedimiento de evento DragOver para determinar lo que ocurre tras iniciarse el arrastre y antes de que un control se coloque en un destino. Por ejemplo, puede comprobar un intervalo de destino válido si resalta el destino (establezca la propiedad **BackColor** o **ForeColor** desde el código) o muestra un puntero de arrastre especial (establezca la propiedad **DragIcon** o **MousePointer** desde el código).
Utilice el argumento *estado* para determinar las acciones de los puntos de transición claves. Por ejemplo, podría resaltar un posible destino cuando *estado* se establezca a 0 (Entrar) y restaurar la apariencia anterior del objeto cuando *estado* se establezca a 1 (Salir).
Cuando un objeto recibe un evento DragOver mientras *estado* está establecido a 0 (Entrar):

- Si el control de origen se coloca en el objeto, ese objeto recibe un evento DragDrop.
- Si el control de origen no se coloca en el objeto, ese objeto recibe otro evento DragOver cuando *estado* se establezca a 1 (Salir).

Nota Utilice la propiedad **DragMode** y el método **Drag** para especificar la manera en que se inicia el arrastre. Si desea ver técnicas sugeridas con el argumento *origen*, vea la sección Comentarios del tema del evento DragDrop.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)

Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento DragOver

Este ejemplo muestra una forma de indicar un destino de colocación válido. El puntero cambia de la flecha predeterminada a un icono especial cuando se arrastra un control **TextBox** sobre un control **PictureBox**. El puntero vuelve a la forma predeterminada cuando el origen se arrastra a cualquier otra parte. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un pequeño **TextBox** y un control **PictureBox**. Establezca a 1 la propiedad **DragMode** del control **TextBox** y, después, presione F5 y arrastre el control **TextBox** sobre el **PictureBox**.

```
Private Sub Picture1_DragOver (Source As Control, X As
Single, Y As Single, State As Integer)
   Select Case State
      Case vbEnter
   ' Carga el icono.
         Source.DragIcon                              =
LoadPicture("ICONS\ARROWS\POINT03.ICO")
      Case vbLeave
         Source.DragIcon = LoadPicture()     ' Descarga el
icono.
   End Select
End Sub

Private Sub Picture1_DragDrop (Source As Control, X As
Single, Y As Single)
   Source.DragIcon = LoadPicture()    ' Descarga el icono.
End Sub
```

DropDown (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando la parte de lista de un control **ComboBox** está a punto de desplegarse; este evento no ocurre si la propiedad **Style** de un control **ComboBox** está establecida a 1 (Simple Combo).

Sintaxis

Private Sub *objeto*_**DropDown([***índice* **As Integer])**

La sintaxis del evento DropDown consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de forma única a un control si está en una matriz de controles.

Comentarios

Utilice un procedimiento de evento DropDown para hacer las actualizaciones finales a una lista **ComboBox** antes de que un usuario realice una selección. Esto le permite agregar o quitar elementos de la lista mediante los métodos **AddItem** o **RemoveItem**. Esta flexibilidad es útil cuando desea interacción entre los controles; por ejemplo, si lo que desea cargar en una lista **ComboBox** depende de lo que seleccione el usuario en un grupo **OptionButton**.

SE APLICA A:

CONSULTAR LAS PROPIEDADES METODOS Y EVENTOS DE ComboBox (Control)

Ejemplo del evento DropDown

Este ejemplo actualiza un control **ComboBox** basándose en la selección realizada por el usuario en un grupo de botones de opción. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ComboBox** y dos controles **OptionButton**. Establezca la propiedad **Name** de ambos controles **OptionButton** a OptionGroup y, después, presione F5 y haga clic en los controles **OptionButton**. El control **ComboBox** refleja los diferentes indicadores de arrastre dependiendo del **OptionButton** seleccionado.

```
Private Sub Form_Load ()
   Combo1.Text = ""    ' Desactiva el cuadro combinado.
End Sub

Private Sub Combo1_DropDown ()
```

```
   Combo1.Clear                          ' Elimina los elementos
existentes.
   If OptionGroup(0).Value = True Then
      Combo1.AddItem "Publicaciones Raimon", 0
      Combo1.AddItem "Publicaciones Scotland ", 1
   Else
      Combo1.AddItem "Editorial GG&G"
   End If
End Sub
```

EnterCell (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando la celda activa actual cambia a una celda diferente.

Sintaxis

Private Sub *objeto*_**EnterCell()**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Al hacer clic en una fila fija, se produce este evento en la primera columna no fija de dicha fila. Este evento no ocurre si se arrastra el *mouse* (ratón) sobre una celda.

SE APLICA A:

MSFlexGrid (Control)

MSHFlexGrid (Control)

GotFocus (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando un objeto recibe el enfoque, ya sea mediante una acción del usuario, como tabular o hacer clic en el objeto, o cambiando el enfoque en el código mediante el método **SetFocus**. Un formulario recibe el enfoque sólo cuando todos los controles visibles están desactivados.

Sintaxis

Private Sub Form_GotFocus( )

Private Sub *objeto*_**GotFocus([***índice* **As Integer])**

La sintaxis del evento GotFocus consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

índice Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios
Normalmente, el procedimiento de evento GotFocus se utiliza para especificar las acciones que ocurren cuando un control o un formulario recibe primero el enfoque. Por ejemplo, si adjunta un procedimiento de evento GotFocus a cada control de un formulario puede guiar al usuario mostrándole instrucciones breves o mensajes en la barra de estado. También puede proporcionar avisos visuales activando, desactivando o mostrando otros controles que dependan del control que tiene el enfoque.

Nota Un objeto puede recibir el enfoque sólo si sus propiedades **Enabled** y **Visible** están establecidas a **True**. Para personalizar la interfaz de teclado en Visual Basic para mover el enfoque, establezca el orden de tabulación o especifique teclas de acceso para controles de un formulario.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento GotFocus

Este ejemplo muestra un mensaje en la barra de estado cuando un botón de un grupo **OptionButton** obtiene el enfoque. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga dos controles **OptionButton** y un

control **Label**. Establezca la propiedad **Name** de ambos controles **OptionButton** a OptionGroup y, después, presione F5 y haga clic en los controles **OptionButton**.

```
Private Sub Form_Load ()
   Label1.AutoSize = True
End Sub

Private Sub OptionGroup_GotFocus (Index As Integer)
   Select Case Index
      Case 0
         Label1.Caption = "La opción 1 tiene el enfoque."
      Case 1
         Label1.Caption = "La opción 2 tiene el enfoque."
   End Select
End Sub

Private Sub OptionGroup_LostFocus (Index As Integer)
   Label1.Caption = ""
End Sub
```

ItemCheck (Evento, control ListBox)

Vea también Ejemplo Se aplica a

Se produce cuando la propiedad **Style** de un control **ListBox** se establece a 1 (casillas de verificación) y se activa o se desactiva la casilla de verificación de un elemento del control **ListBox**.

Sintaxis

Private Sub *objeto*_**ItemCheck**([*índice* **As Integer**])

La sintaxis del evento ItemCheck consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de forma única al elemento del cuadro de lista en el que se hizo clic.

Comentarios

Nota El evento ItemCheck no se produce cuando sólo se resalta un elemento de la lista; ocurre cuando se activa o se desactiva la casilla de verificación del elemento de la lista.

El evento ItemCheck también puede producirse por programa siempre que se cambie un elemento de la matriz Selected del **ListBox** (y su propiedad **Style** se establezca a 1).
El evento ItemCheck se produce antes del evento Click.

Intentar la descarga de un formulario utilizando **Unload Me** en el evento ItemCheck de un control **ListBox** produce un Error de protección general (GPF). Por ejemplo:

```
Private Sub Form_Load()
    Dim i%
    For i = 0 To 20
      List1.AddItem i
    Next i
  End Sub

Private Sub List1_ItemCheck(Item As Integer)
  Unload Me
End Sub
```

Este ejemplo también produce un GPF si se pulsa la barra espaciadora o si intenta seleccionar un elemento en el **ListBox** utilizando un **CommandButton**.

El procedimiento recomendado para descargar un formulario es agregar la sentencia **Unload** al evento Click en un control **CommandButton** o **Menu**.

SE APLICA A:

CONSULTAR PROPIEDADES METODOS Y EVENTOS DE ListBox (Control)

KeyDown, KeyUp (Eventos)

Vea también Ejemplo Se aplica a

Ocurren cuando el usuario presiona (KeyDown) o suelta (KeyUp) una tecla mientas un objeto tiene el enfoque. (Para interpretar los caracteres ANSI, utilice el evento KeyPress.)

Sintaxis

Private Sub Form_KeyDown(*códigoTecla* **As Integer,** *mayús* **As Integer)**

Private Sub *objeto***_KeyDown([***índice* **As Integer,]***códigoTecla* **As Integer,** *mayús* **As Integer)**

Private Sub Form_KeyUp(*códigoTecla* **As Integer,** *mayús* **As Integer)**

Private Sub *objeto***_KeyUp([***índice* **As Integer,]***códigoTecla* **As Integer,** *mayús* **As Integer)**

La sintaxis de los eventos KeyDown y KeyUp consta de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de forma única a un control si está en una matriz de controles.

códigoTecla Un código de tecla, como **vbKeyF1** (la tecla F1) o **vbKeyHome** (la tecla INICIO). Para especificar códigos de tecla, utilice las constantes de la biblioteca de objetos de Visual Basic (VB) del Examinador de objetos.

mayús Un entero que corresponde al estado de las teclas MAYÚS, CTRL y ALT en el momento del evento. El argumento *mayús* es un campo de bit con los bits menos significativos correspondientes a la tecla MAYÚS (bit 0), CTRL (bit 1) y ALT (bit 2). Estos bits corresponden a los valores 1, 2 y 4, respectivamente. Algunos, todos o ninguno de estos bits pueden estar establecidos, lo que indica que alguna, todas o ninguna de las teclas están presionadas. Por ejemplo, si están presionadas las teclas CTRL y ALT, el valor de *mayús* es 6.

Comentarios

Para ambos eventos, el objeto que tiene el enfoque recibe todas las pulsaciones de tecla. Un formulario sólo puede tener el enfoque si no tiene controles visibles y activados. Aunque los eventos KeyDown y KeyUp pueden aplicarse a la mayoría de las teclas, se suelen usar más comúnmente para:

- Teclas de caracteres extendidos como las teclas de función.
- Teclas de desplazamiento.
- Combinaciones de teclas con modificadores de teclado estándares.
- Distinguir entre el teclado numérico y las teclas de número normales.

Utilice los procedimientos de evento KeyDown y KeyUp si necesita responder a presionar y soltar una tecla.

KeyDown y KeyUp no se invocan para:

- La tecla ENTRAR si el formulario tiene un control **CommandButton** con la propiedad **Default** establecida a **True**.
- La tecla ESC si el formulario tiene un control **CommandButton** con la propiedad **Cancel** establecida a **True**.
- La tecla TAB.

KeyDown y KeyUp interpretan si los caracteres están en mayúsculas o en minúsculas mediante dos argumentos: *códigoTecla*, que indica la tecla física (devuelve A y a como la misma tecla) y *mayús*, que indica el estado de *mayús-tecla* y, por tanto, devuelve A o a.

Si necesita probar el argumento *mayús*, puede usar las constantes *mayús* que definen los bits en el argumento. Las constantes tienen los siguientes valores:

Constante	Valor	Descripción
vbShiftMask	1	Máscara de bits de la tecla MAYÚS.
VbCtrlMask	2	Máscara de bits de la tecla CTRL.
VbAltMask	4	Máscara de bits de la tecla ALT.

Las constantes actúan como máscaras de bits, que puede usar para comprobar cualquier combinación de teclas.

Para probar una condición, asigne primero cada resultado a una variable temporal entera y después compare *mayús* con una máscara de bits. Utilice el operador **And** con el argumento *mayús* para comprobar si la condición es mayor que 0, lo que indica que el modificador se presionó, como en este ejemplo:

```
ShiftDown = (Shift And vbShiftMask) > 0
```

En un procedimiento puede comprobar cualquier combinación de teclas, como en este ejemplo:

```
If ShiftDown And CtrlDown Then
```

Nota Si la propiedad **KeyPreview** está establecida a **True**, un formulario recibe estos eventos antes de que los controles del formulario reciban los eventos. Utilice la propiedad **KeyPreview** para crear rutinas globales de control del teclado.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de los eventos KeyDown y KeyUp

Este ejemplo muestra un controlador de teclado genérico que responde a la tecla F2 y a todas las combinaciones de teclas ALT, MAYÚS y CTRL. Las constantes de tecla se enumeran en la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** y, después, presione F5 y presione F2 con varias combinaciones de las teclas ALT, MAYÚS y CTRL.

```
Private Sub Text1_KeyDown (KeyCode As Integer, Shift As
Integer)
   Dim ShiftDown, AltDown, CtrlDown, Txt
   ShiftDown = (Shift And vbShiftMask) > 0
   AltDown = (Shift And vbAltMask) > 0
   CtrlDown = (Shift And vbCtrlMask) > 0
   If KeyCode = vbKeyF2 Then    ' Muestra combinaciones de
teclas.
   If ShiftDown And CtrlDown And AltDown Then
      Txt = "Mayús-Ctrl-Alt-F2."
   ElseIf ShiftDown And AltDown Then
      Txt = "Mayús-Alt-F2."
   ElseIf ShiftDown And CtrlDown Then
      Txt = "Mayús-Ctrl-F2."
   ElseIf CtrlDown And AltDown Then
      Txt = "Ctrl-Alt-F2."
   ElseIf ShiftDown Then
      Txt = "Mayús-F2."
   ElseIf CtrlDown Then
   Txt = "Ctrl-F2."
   ElseIf AltDown Then
      Txt = "Alt-F2."
   ElseIf SHIFT = 0 Then
      Txt = "F2."
   End If
   Text1.Text = "Ha presionado " & Txt
   End If
End Sub
```

KeyDown, KeyUp (Eventos, controles ActiveX)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario presiona (KeyDown) o suelta (KeyUp) una tecla mientras un objeto tiene el enfoque. (Para interpretar los caracteres ANSI, utilice el evento KeyPress.)

Sintaxis

Private Sub Form_KeyDown(*códigoTecla* As Integer, *mayús* As Integer)

Private Sub *objeto*_**KeyDown([***índice* **As Integer,]***códigoTecla* **As Integer,** *mayús* **As Integer)**
Private Sub Form_KeyUp(*códigoTecla* **As Integer,** *mayús* **As Integer)**
Private Sub *objeto*_**KeyUp([***índice* **As Integer,]** *códigoTecla* **As Integer,** *mayús* **As Integer)**

Las sintaxis de los eventos KeyDown y KeyUp constan de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

índice Un entero que identifica unívocamente a un control si está en una matriz de controles.

códigoTecla Un código de tecla, como **vbKeyF1** (la tecla F1) o **vbKeyHome** (la tecla Inicio). Para especificar códigos de tecla utilice las constantes de la biblioteca de objetos de Visual Basic (VB) del Examinador de objetos.

mayús Un entero que corresponde al estado de las teclas MAYÚS, CTRL, y ALT en el momento del evento. El argumento *mayús* es un campo de bits que contiene los bits menos significativos correspondientes a la tecla MAYÚS (bit 0), la tecla CTRL (bit 1) y la tecla ALT (bit 2). Estos bits corresponden a los valores 1, 2 y 4 respectivamente. Es posible establecer alguno, todos o ninguno de estos bits, indicando que alguna, todas, o ninguna de las teclas están presionadas. Por ejemplo, si están presionadas CTRL y ALT el valor de *mayús* es 6.

Comentarios

Para ambos eventos, el objeto que posee el enfoque recibe todas las pulsaciones de tecla. Un formulario puede tener el enfoque sólo si no tiene controles visibles y activados. Aunque los eventos KeyDown y KeyUp pueden aplicarse a la mayoría de las teclas, se suelen usar más comúnmente para:

- Teclas de caracteres extendidos como las teclas de función.
- Teclas de desplazamiento.
- Combinaciones de teclas con modificadores de teclado estándares.
- Distinguir entre el teclado numérico y las teclas de número normales.

Utilice los procedimientos de evento KeyDown y KeyUp si necesita responder a ambos presionando y soltando una tecla.
KeyDown y KeyUp no se invocan para:

- La tecla ENTRAR si el formulario tiene un control **CommandButton** con la propiedad **Default** establecida a **True**.

- La tecla ESC si el formulario tiene un control **CommandButton** con la propiedad **Cancel** establecida a **True**.

- La tecla TAB.

KeyDown y KeyUp interpretan si los caracteres están en mayúsculas o en minúsculas mediante dos argumentos: *códigoTecla*, que indica la tecla física (así devuelve A y a como la misma tecla) y *mayús*, que indica el estado de *mayús-tecla* y, por tanto, devuelve A o a.
Si necesita comprobar el argumento *mayús*, puede declarar constantes *mayús* que definan los bits dentro del argumento. Las constantes tienen los siguientes valores:

Constante	Valor	Descripción
vbShiftMask	1	Máscara de bit de la tecla MAYÚS.
VbCtrlMask	2	Máscara de bit de la tecla CTRL.
VbAltMask	4	Máscara de bit de la tecla ALT.

Las constantes actúan como máscaras de bits que puede usar para comprobar cualquier combinación de teclas.
Para comprobar una condición, asigne primero cada resultado a una variable entera temporal y después compare *mayús* con una máscara de bits. Utilice el operador **And** con el argumento *mayús* para comprobar si la condición es mayor que 0, lo que indica que el modificador se presionó, como en este ejemplo:

```
ShiftDown = (Shift And vbShiftMask) > 0
```

En un procedimiento, puede comprobar cualquier combinación de teclas, como en este ejemplo:

```
If ShiftDown And CtrlDown Then
```

Nota Si la propiedad **KeyPreview** está establecida a **True**, un formulario recibe estos eventos antes de que los controles del formulario reciban los eventos. Utilice la propiedad **KeyPreview** para crear rutinas globales de control del teclado.

SE APLICA A:
DataRepeater (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
RichTextBox (Control)
TextBox (Control Ligero)

KeyPress (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario presiona y suelta una tecla ANSI.

Sintaxis

Private Sub Form_KeyPress(*teclaAscii* As Integer)

Private Sub *objeto*_KeyPress([*índice* As Integer,]*teclaAscii* As Integer)

La sintaxis del evento KeyPress consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.
teclaAscii	Un entero que devuelve un código de tecla numérico ANSI estándar. *teclaAscii* se pasa por referencia; al cambiarlo se envía un carácter diferente al objeto. Cambiar *teclaAscii* a 0 cancela la pulsación de tecla, de forma que el objeto no recibe ningún carácter.

Comentarios

El objeto que tiene el enfoque recibe el evento. Un formulario puede recibir el evento sólo si no tiene controles visibles y activados o si la propiedad **KeyPreview** está establecida a **True**. Un evento KeyPress puede implicar a cualquier carácter imprimible del teclado, a la tecla CTRL combinada con un carácter del alfabeto estándar o uno de los caracteres especiales, y la tecla ENTRAR o RETROCESO. Un procedimiento de evento KeyPress es útil para interceptar pulsaciones de teclas realizadas en un control **TextBox** o **ComboBox**. Esto le permite comprobar inmediatamente la validez de las pulsaciones o el formato de los caracteres a medida que se escriben. Cambiar el valor del argumento *teclaAscii* cambia el carácter mostrado.

Puede convertir el argumento *teclaAscii* en un carácter mediante la expresión:

```
Chr(teclaAscii)
```

También puede realizar operaciones sobre cadenas y traducir los caracteres a un número ANSI que el control pueda interpretar, utilizando para ello la expresión:

```
teclaAscii = Asc(char)
```

Utilice los procedimientos de evento KeyDown y KeyUp para controlar cualquier pulsación de tecla no reconocida por KeyPress, como las teclas de función, las teclas de edición, las teclas de desplazamiento y cualquier combinación de éstas con modificadores del teclado. A diferencia de los eventos KeyDown y KeyUp, KeyPress no indica el estado físico del teclado; en su lugar, pasa un carácter.

KeyPress interpreta las mayúsculas y minúsculas de cada carácter como códigos de tecla distintos y, por tanto, como caracteres diferentes. KeyDown y KeyUp interpretan el estado de mayúsculas o minúsculas de cada carácter mediante dos argumentos:

códigoTecla, que indica la tecla física (devuelve A y a como la misma tecla) y *mayús*, que indica el estado de la combinación *mayús-tecla* y, por tanto, devuelve A o a.

Si la propiedad **KeyPreview** está establecida a **True**, un formulario recibe el evento antes de que lo reciban los controles del formulario. Utilice la propiedad **KeyPreview** para crear rutinas globales de control de teclado.

Nota El número ANSI para la combinación de teclado Ctrl-@ es 0. Puesto que Visual Basic reconoce un valor *teclaAscii* de 0 como una cadena de longitud cero ("""), evite usar Ctrl-@ en sus aplicaciones.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento KeyPress

Este ejemplo convierte a mayúsculas el texto escrito en un control **TextBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** y, después, presione F5 y escriba algo en el control **TextBox**.

```
Private Sub Text1_KeyPress (KeyAscii As Integer)
   Char = Chr(KeyAscii)
   KeyAscii = Asc(UCase(Char))
End Sub
```

KeyPress (Evento, controles ActiveX)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario presiona y suelta una tecla ANSI.

Sintaxis
Private Sub Form_KeyPress(*valorAscii* As Integer)
Private Sub *objeto*_KeyPress([*índice* As Integer,]*valorAscii* As Integer)
La sintaxis del evento KeyPress consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica unívocamente a un control si está en una matriz de controles.
valorAscii	Un entero que devuelve un código de tecla numérico del estándar ANSI. *ValorAscii* se pasa por referencia; cambiarlo envía un carácter diferente al objeto. Cambiar *valorAscii* a 0 cancela la pulsación de tecla, de forma que el objeto no recibe ningún carácter.

Comentarios
El objeto que posee el enfoque recibe el evento. Un formulario sólo puede recibir el evento si no tiene controles visibles y activados o si la propiedad **KeyPreview** está establecida a **True**. Un evento KeyPress puede implicar a cualquier carácter imprimible, a la tecla CTRL combinada con un carácter del alfabeto estándar o unos pocos caracteres especiales y la tecla ENTRAR o RETROCESO. Un procedimiento de evento KeyPress es útil para interceptar pulsaciones de teclas introducidas en un control **TextBox** o **ComboBox**. Esto le permite comprobar inmediatamente la validez o dar formato a los caracteres a medida que se escriben. Cambiar el valor del argumento *valorAscii* cambia el carácter mostrado.
Puede convertir el argumento *valorAscii* en un carácter mediante la siguiente expresión:

```
Chr(KeyAscii)
```

También puede realizar operaciones sobre cadenas y traducir los caracteres a un número ANSI que el control pueda interpretar, utilizando para ello la expresión:

```
KeyAscii = Asc(char)
```

Utilice los procedimientos de evento KeyDown y KeyUp para controlar cualquier pulsación de tecla no reconocida por KeyPress, como las teclas de función, las teclas de edición, las teclas de desplazamiento y cualquier combinación de éstas con modificadores del teclado. A diferencia de los eventos KeyDown y KeyUp, KeyPress no indica el estado físico del teclado; en su lugar, pasa un carácter.

KeyPress interpreta los caracteres en mayúsculas y en minúsculas como códigos de tecla distintos y, por tanto, como caracteres diferentes. KeyDown y KeyUp interpretan el estado de mayúsculas o minúsculas de cada carácter mediante dos argumentos: *códigoTecla*, que indica la tecla física (devuelve A y a como la misma tecla), y *mayús*, que indica el estado de la combinación *mayús-tecla* y, de esta manera, devuelve A o a.

Si la propiedad **KeyPreview** está establecida a **True**, un formulario recibe el evento antes que lo reciban los controles del formulario. Utilice la propiedad **KeyPreview** para crear rutinas de control de teclado globales.

Nota El número ANSI para la combinación de teclado de Ctrl-@ es 0. Puesto que Visual Basic reconoce un valor *valorAscii* de 0 como una cadena de longitud cero ("""), evite usar Ctrl-@ en sus aplicaciones.

SE APLICA A:
DataRepeater (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
RichTextBox (Control)
TextBox (Control Ligero)

LinkClose (Evento)

Vea también Ejemplo Se aplica a
Ocurre cuando termina una conversación DDE. Cualquier aplicación de una conversación DDE puede terminar una conversación en cualquier momento.

Sintaxis
Private Sub Form_LinkClose( )
Private Sub MDIForm_LinkClose( )
Private Sub *objeto*_**LinkClose([***índice* **As Integer])**
La sintaxis del evento LinkClose consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios
Normalmente, el procedimiento de evento LinkClose se utiliza para notificar al usuario de que una conversación DDE ha terminado. También puede incluir información sobre solución de problemas, sobre el restablecimiento de la conexión o dónde acudir en busca de asistencia. Para mostrar mensajes breves, utilice la función **MsgBox**.

SE APLICA A:
Form (Objeto), Forms (Colección)
Label (Control)
MDIForm (Objeto)
PictureBox (Control)
TextBox (Control)

LinkError (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando se produce un error durante una conversación DDE. Este evento sólo se reconoce como el resultado de un error relacionado con DDE que ocurre cuando no se está ejecutando código de Visual Basic. El número de error se pasa como un argumento.

Sintaxis

Private Sub Form_LinkError(*errorVínculo* **As Integer)**
Private Sub MDIForm_LinkError(*errorVínculo* **As Integer)**
Private Sub *objeto***_LinkError([***índice* **As Integer,]***errorVínculo* **As Integer)**
La sintaxis del evento LinkError consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
errorVínculo	Un número de error del error relacionado con DDE, tal y como se describe en Valores que se pueden obtener.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Valores que se pueden obtener

La tabla siguiente muestra una lista de todos los números de error que devuelve el argumento *errorVínculo* con una breve explicación de cada error:

Valor	Descripción
1	La otra aplicación ha pedido datos en un formato erróneo. Este error puede ocurrir varias veces sucesivas a medida que Visual Basic intenta encontrar un formato que la otra aplicación reconozca.
6	La aplicación de destino intentó continuar una conversación DDE después de que configurase la propiedad **LinkMode** de su formulario de origen a 0 (Ninguno).
7	Todos los vínculos de origen están en uso (hay un límite de 128 vínculos por origen).
8	Para controles de destino: un vínculo automático o un método **LinkRequest** no pudo actualizar los datos del control. Para formularios de origen: el destino intentó escribir datos en un control y el intento falló.
11	No hay memoria suficiente para DDE.

Comentarios

Utilice un procedimiento de evento LinkError para notificar al usuario de que ha ocurrido un determinado error. También puede incluir código para resolver el problema o informar sobre la solución de problemas en el restablecimiento de una conexión o a dónde acudir si se necesita asistencia. Para mostrar mensajes breves, utilice la función **MsgBox**.

SE APLICA A:
Form (Objeto), Forms (Colección)
Label (Control)
MDIForm (Objeto)
PictureBox (Control)
TextBox (Control)

LinkNotify (Evento)

Vea también Ejemplo Se aplica a
Ocurre cuando el origen ha cambiado los datos definidos por el vínculo DDE si la propiedad **LinkMode** del control de destino está establecida a 3 (Notify).

Sintaxis
Private Sub *objeto*_**LinkNotify**([*índice* **As Integer**])
La sintaxis del evento LinkNotify consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios
Normalmente, en el evento LinkNotify el código notifica al usuario, obtiene el nuevo dato inmediatamente o retrasa la obtención del dato hasta más tarde. Puede usar el método **LinkRequest** para obtener el nuevo dato del origen.

SE APLICA A:
Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo del evento LinkNotify

Este ejemplo se adjunta a un control **PictureBox**, Picture1, que tiene sus propiedades **LinkTopic** y **LinkItem** establecidas a un gráfico del origen y la propiedad **LinkMode** establecida a 3 (Notify). Cuando el origen cambia sus datos, el procedimiento actualiza el control **PictureBox** inmediatamente sólo si el control **PictureBox** está en el formulario

activo; de lo contrario, configura una variable indicadora. Este ejemplo es sólo ilustrativo.

```
Private Sub Picture1_LinkNotify ()
   If Screen.ActiveForm Is Me Then
   Picture1.LinkRequest          ' La imagen está en el
formulario activo, por lo tanto se actualiza.
   Else
      NewDataFlag = True      ' Se supone que está en una
variable a nivel del módulo.
   End If
End Sub
```

LinkOpen (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando se está iniciando una conversación DDE.

Sintaxis

Private Sub Form_LinkOpen(*cancelar* As Integer)

Private Sub MDIForm_LinkOpen(*cancelar* As Integer)

Private Sub *objeto*_LinkOpen([*índice* As Integer,]*cancelar* As Integer)

La sintaxis del evento LinkOpen consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
cancelar	Un entero que determina si la conversación DDE se ha establecido o no. Dejar *cancelar* establecido a 0 (el valor predeterminado) establece la conversación. Establecer *cancelar* a cualquier valor distinto de cero rechaza la conversación.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios

Este evento ocurre en los formularios cuando una aplicación de destino está iniciando una conversación DDE con el formulario. Ocurre en los controles cuando un control está iniciando una conversación DDE con una aplicación de origen.

SE APLICA A:

Form (Objeto), Forms (Colección)

Label (Control)

MDIForm (Objeto)
PictureBox (Control)
TextBox (Control)

LostFocus (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando un objeto pierde el enfoque, ya sea por una acción del usuario, como tabular o hacer clic en otro objeto, o bien mediante un cambio del enfoque en el código con el método **SetFocus**.

Sintaxis

Private Sub Form_LostFocus( )

Private Sub *objeto*_**LostFocus([***índice* **As Integer])**

La sintaxis del evento LostFocus consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios

Un procedimiento de evento LostFocus resulta especialmente útil para comprobar y validar actualizaciones. Usar LostFocus puede hacer que la validación tenga lugar conforme el usuario mueve el enfoque del control. Otro uso para este tipo de procedimiento de evento es activar, desactivar, ocultar y mostrar otros objetos, como en un procedimiento de evento GotFocus. También puede invertir o cambiar condiciones que estableció en el procedimiento de evento GotFocus del objeto.

Si un archivo .exe creado con Visual Basic muestra un cuadro de diálogo de un archivo .dll creado también con Visual Basic, el formulario del archivo .exe obtendrá los eventos Deactivate y LostFocus. Esto puede ser inesperado, ya que no debe obtener el evento Deactivate:

- Si el objeto es un componente fuera de proceso.
- Si el objeto no está escrito en Visual Basic.
- En el entorno de desarrollo, cuando se llame a un archivo DLL generado en Visual Basic.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO

DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento LostFocus

Este ejemplo cambia el color de un control **TextBox** cuando recibe o pierde el enfoque (se selecciona con el *mouse* o la tecla TAB) y muestra el texto apropiado en el control **Label**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga dos controles **TextBox** y un control **Label** y, después, presione F5 y mueva el enfoque entre Text1 y Text2.

```
Private Sub Text1_GotFocus ()
   ' Muestra el enfoque en rojo.
   Text1.BackColor = RGB(255, 0, 0)
   Label1.Caption = "Text1 tiene el enfoque."
End Sub

Private Sub Text1_LostFocus ()
   ' Muestra la pérdida del enfoque en azul.
   Text1.BackColor = RGB(0, 0, 255)
   Label1.Caption = "Text1 no tiene el enfoque."
End Sub
```

MouseDown, MouseUp (Eventos)

Vea también Ejemplo Se aplica a

Ocurren cuando el usuario presiona (MouseDown) o suelta (MouseUp) un botón del *mouse*.

Sintaxis

Private Sub Form_MouseDown(*botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

Private Sub MDIForm_MouseDown(*botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

Private Sub *objeto***_MouseDown([***índice* **As Integer,]***botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

Private Sub Form_MouseUp(*botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

Private Sub MDIForm_MouseUp(*botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

Private Sub *objeto* **_MouseUp([***índice* **As Integer,]***botón* **As Integer,** *mayús* **As Integer,** *x* **As Single,** *y* **As Single)**

La sintaxis del evento MouseDown y MouseUp consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.
botón	Devuelve un entero que identifica el botón que se presionó (MouseDown) o se soltó (MouseUp) para causar el evento. El argumento *botón* es un campo de bits que corresponde al botón primario (bit 0), al botón secundario (bit 1) y al botón central (bit 2). Estos bits corresponden a los valores 1, 2 y 4, respectivamente. Sólo uno de los bits está establecido, lo que indica el botón que causó el evento.
mayús	Devuelve un entero que corresponde al estado de las teclas MAYÚS, CTRL y ALT cuando el botón especificado en el argumento *botón* se presionó o se soltó. Se establece un bit si la tecla está presionada. El argumento *mayús* es un campo de bits con los bits menos significativos correspondientes a la tecla MAYÚS (bit 0), CTRL (bit 1) y ALT (bit 2). Estos bits corresponden a los valores 1, 2 y 4, respectivamente. El argumento *mayús* indica el estado de estas teclas. Algunos, todos o ninguno de los bits puede estar establecido, lo que indica que alguna, todas o ninguna de las teclas está presionada. Por ejemplo, si se presionaron las teclas CTRL y ALT, el valor de *mayús* sería 6.
x, y	Devuelve un número que especifica la ubicación actual del puntero del *mouse*. Los valores *x* e *y* se expresan siempre en términos del sistema de coordenadas establecido por las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** del objeto.

Comentarios
Utilice un procedimiento de evento MouseDown o MouseUp para especificar acciones que ocurrirán cuando se presione o se suelte un determinado botón del *mouse*. A diferencia de los eventos Click y DblClick, los eventos MouseDown y MouseUp le permiten distinguir entre los botones primario, secundario y central. También puede escribir código para combinaciones de teclado y *mouse* que utilicen los modificadores de teclado MAYÚS, CTRL y ALT.

Lo siguiente se aplica a los eventos Click y DblClick:

- Si un botón del *mouse* está presionado mientras el puntero está sobre un formulario o un control, ese objeto "captura" el *mouse* y recibe todos los eventos del mismo hasta ese momento e incluyendo el último evento MouseUp. Esto implica que las coordenadas del puntero *x*, *y* devueltas por un evento del *mouse* puede que no estén siempre en el área interna del objeto que las recibe.

- Si los botones del *mouse* se presionan sucesivamente, el objeto que captura el *mouse* tras la primera pulsación recibe todos los eventos del *mouse* hasta que se hayan soltado todos los botones.

Si necesita comprobar los argumentos *botón* o *mayús* puede usar las constantes que se enumeran en la biblioteca de objetos del Examinador de objetos de Visual Basic (VB) para definir los bits de argumento:

Constante (Botón)	Valor	Descripción
vbLeftButton	1	Está presionado el botón primario.
VbRightButton	2	Está presionado el botón secundario.
vbMiddleButton	4	Está presionado el botón central.

Constante (Mayús)	Valor	Descripción
vbShiftMask	1	Está presionada la tecla MAYÚS.
vbCtrlMask	2	Está presionada la tecla CTRL.
vbAltMask	4	Está presionada la tecla ALT.

Las constantes actúan entonces como máscaras de bits puede usar para probar cualquier combinación de botones sin tener que averiguar el valor único de campo de bit para cada combinación.

Nota Puede usar un procedimiento de evento MouseMove para responder a un evento causado por un movimiento del *mouse*. El argumento *botón* de MouseDown y MouseUp difiere del argumento *botón* utilizado para MouseMove. Para MouseDown y MouseUp, el argumento *botón* indica exactamente un botón por evento, mientras que para MouseMove, indica el estado actual de todos los botones.

SE APLICA A:

CheckBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DirListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de los eventos MouseDown y MouseUp

Este ejemplo muestra una aplicación de dibujo simple. El procedimiento de evento MouseDown funciona con un procedimiento de evento MouseMove relacionado para activar el dibujo cuando cualquier botón del *mouse* está presionado y se arrastra. El procedimiento de evento MouseUp desactiva el dibujo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario y, después, presione F5, haga clic en el formulario y mueva el *mouse* mientras presiona el botón del mismo.

```
Dim PaintNow As Boolean
Private Sub Form_MouseDown (Button As Integer, Shift As
Integer, X As
Single, Y As Single)
   PaintNow = True    ' Activa el dibujo.
End Sub

Private Sub Form_MouseUp (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   PaintNow = False    ' Desactiva el dibujo.
End Sub
```

```
Private Sub Form_MouseMove (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   If PaintNow Then
      PSet (X, Y)        ' Dibuja un punto.
   End If
End Sub

Private Sub Form_Load ()
   DrawWidth = 10        ' Utiliza un pincel más ancho.
   ForeColor = RGB(0, 0, 255)     ' Establece el color de
dibujo.
End Sub
```

MouseMove (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando el usuario mueve el *mouse*.

Sintaxis

Private Sub Form_MouseMove(*botón* As Integer, *mayús* As Integer, *x* As Single, *y* As Single)

Private Sub MDIForm_MouseMove(*botón* As Integer, *mayús* As Integer, *x* As Single, *y* As Single)

Private Sub *objeto*_MouseMove([*índice* As Integer,] *botón* As Integer, *mayús* As Integer, *x* As Single, *y* As Single)

La sintaxis del evento MouseMove consta de las siguientes partes:

Parte	Descripción
objeto	Un expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.
botón	Un entero que corresponde al estado de los botones del *mouse* en el cual un bit se establece si el botón está presionado. El argumento *botón* es un campo de bit con los bits correspondientes al botón primario (bit 0), al botón secundario (bit 1) y al botón central (bit 2). Estos bits corresponden a los valores 1, 2 y 4, respectivamente. Indica el estado completo de los botones del *mouse*; alguno, todos o ninguno de estos tres bits puede estar establecido, lo que indica que algunos, todos o ninguno de los botones está presionado.
mayús	Un entero que corresponde al estado de las teclas MAYÚS, CTRL y ALT. Un bit está establecido si la tecla está presionada. El argumento *mayús* es un campo de bits con los bits menos significativos correspondientes a la tecla MAYÚS (bit 0), CTRL (bit 1) y ALT (bit 2). Estos bits corresponden a los valores 1, 2 y 4, respectivamente. El argumento *mayús* indica el estado de estas teclas. Alguno, todos o ninguno de los bits puede estar establecido, lo que indica que alguna, todas o ninguna de las teclas está presionada. Por ejemplo, si se presionaron las teclas CTRL y ALT, el valor de *shift* sería 6.

x, y Un número que especifica la ubicación actual del puntero del *mouse*. Los valores *x* e *y* siempre se expresan en términos del sistema de coordenadas establecido por las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** del objeto.

Comentarios

El evento MouseMove se genera continuamente a medida que el puntero del *mouse* se mueve por los objetos. A menos que otro objeto haya capturado el *mouse*, un objeto reconoce un evento MouseMove siempre que la posición del *mouse* esté dentro de sus bordes.

Si necesita comprobar los argumentos *botón* o *mayús*, puede usar las constantes enumeradas en la biblioteca de objetos del Examinador de objetos de Visual Basic (VB) para definir los bits dentro del argumento:

Constante (Botón)	Valor	Descripción
vbLeftButton	1	Está presionado el botón primario.
vbRightButton	2	Está presionado el botón secundario.
vbMiddleButton	4	Está presionado el botón central.

Constante (Mayús)	Valor	Descripción
vbShiftMask	1	Está presionada la tecla MAYÚS.
vbCtrlMask	2	Está presionada la tecla CTRL.
vbAltMask	4	Está presionada la tecla ALT.

Las constantes actúan entonces como máscaras de bits que puede usar para comprobar cualquier combinación de botones sin tener que averiguar el valor único del campo de bits para cada combinación.

Puede comprobar una condición si asigna primero cada resultado a una variable temporal entera y compara después los argumentos *botón* o *mayús* con una máscara de bits. Utilice el operador **And** con cada argumento para comprobar si la condición es mayor que cero, lo que indica que la tecla o el botón está presionado, como en este ejemplo:

```
LeftDown = (Button And vbLeftButton) > 0
CtrlDown = (Shift And vbCtrlMask) > 0
```

Después, en un procedimiento, puede comprobar cualquier combinación de condiciones, como en este ejemplo:

```
If LeftDown And CtrlDown Then
```

Nota Puede usar los procedimientos de evento MouseDown y MouseUp para responder a los eventos causados por presionar y soltar los botones del *mouse*.

El argumento *botón* para MouseMove difiere del argumento *botón* para MouseDown y MouseUp. Para MouseMove, el argumento *botón* indica el estado actual de todos los botones; un único evento MouseEvent puede indicar que alguno, todos o ningún botón está presionado. Para MouseDown y MouseUp, el argumento *botón* indica exactamente un botón por evento.

Cada vez que mueva una ventana dentro de un evento MouseEvent puede provocar un evento en cascada. Los eventos MouseMove se generan cuando la ventana se mueve debajo del puntero. Un evento MouseMove puede generarse incluso si el *mouse* está totalmente quieto.

SE APLICA A:

CheckBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DirListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento MouseMove

Este ejemplo muestra una aplicación de dibujo simple. El procedimiento de evento MouseDown funciona con un procedimiento MouseMove relacionado para activar el dibujo cuando está presionado cualquier botón del *mouse*. El procedimiento de evento MouseUp desactiva el dibujo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, después presione F5, haga clic en el formulario y mueva el *mouse* mientras está presionado el botón del mismo.

```
Dim PaintNow As Boolean   ' Declara una variable.
Private Sub Form_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   PaintNow = True   ' Activa el dibujo.
```

```
End Sub

Private Sub Form_MouseUp (Button As Integer, X As Single, Y
As Single)
   PaintNow = False    ' Desactiva el dibujo.
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   If PaintNow Then
      PSet (X, Y)       ' Dibuja un punto.
   End If
End Sub

Private Sub Form_Load ()
   DrawWidth = 10       ' Utiliza un pincel más ancho.
   ForeColor = RGB(0, 0, 255)    ' Establece el color de
dibujo.
End Sub
```

ObjectEvent (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando un control asignado a una variable de objeto **VBControlExtender** genera un evento.

Sintaxis

Private Sub *objeto*_**ObjectEvent**(*info* **As EventInfo**)

La sintaxis del evento ObjectEvent consta de las siguientes partes:

Parte	Descripción
objeto	Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
info	Devuelve una referencia a un objeto **EventInfo**.

Comentarios

El evento ObjectEvent es un evento genérico que permite controlar los eventos de un control y devolver valores a través de los parámetros de evento del control.

Utilice el evento ObjectEvent para interceptar eventos genéricos generados por un control con el fin de asegurarse de que cualquier control desplegado contiene ciertas funciones básicas requeridas por la aplicación desplegada.

SE APLICA A:

CONSULTAR LAS PROPIEDADES METODOS Y EVENTOS DE Extender (Objeto)

ObjectEvent (Evento), ejemplos del objeto EventParameter

El primer ejemplo a continuación utiliza el evento ObjectEvent para imprimir los nombres y los valores de todos los parámetros.

```
Option Explicit
Private WithEvents extObj As VBControlExtender

' Código que establece la variable de objeto con un control
personalizado no mostrado.

Private Sub extObj_ObjectEvent(Info As EventInfo)
   Dim p As EventParameter
   Debug.Print Info.Name

   For Each p In Info.EventParameters
      Debug.Print p.Name, p.Value
   Next
End Sub
```

El segundo ejemplo comprueba si el control generó un evento genérico.

```
Private Sub extObj_ObjectEvent(Info As EventInfo)
   Dim p As EventParameter
   Select Case Info.Name
   Case "UserName"
      ' Comprueba el valor del nombre de usuario.
      MsgBox Info.EventParameters("UserName").Value
   ' Se muestran aquí otras situaciones
   Case Else ' Evento no identificado
      ' Controla los eventos no identificados aquí.
   End Select
End Sub
```

ObjectMove (Evento)

Vea también Ejemplo Se aplica a

Se produce inmediatamente después de mover o ajustar el tamaño del objeto que se encuentra dentro de un control contenedor **OLE** mientras el objeto está activo.

Sintaxis

Private Sub *objeto*_**ObjectMove(***izquierda* **As Single,** *arriba* **As Single,** *ancho* **As Single,** *alto* **As Single)**

La sintaxis del evento ObjectMove consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

izquierda La coordenada del borde izquierdo del control contenedor **OLE** inmediatamente después de moverlo o cambiarlo de tamaño.

arriba La coordenada del borde superior del control contenedor **OLE** inmediatamente después de moverlo o cambiarlo de tamaño.

ancho El ancho del control contenedor **OLE** inmediatamente después de moverlo o cambiarlo de tamaño.

alto El alto del control contenedor **OLE** inmediatamente después de moverlo o cambiarlo de tamaño.

Comentarios

Cuando un usuario mueve o cambia el tamaño de un control contenedor **OLE**, la aplicación puede usar el evento ObjectMove para determinar si debe cambiar realmente el tamaño y la posición del control. Si el procedimiento de evento ObjectMove no cambia la posición o el tamaño del control contenedor **OLE**, el objeto existente dentro del control contenedor **OLE** vuelve a su posición original y se informa de su nuevo tamaño. Las coordenadas pasadas como argumentos a este evento incluyen el borde del control contenedor **OLE**.

Los eventos ObjectMove y Resize se desencadenan cuando el control contenedor **OLE** recibe información acerca del tamaño del objeto que contiene. Sin embargo, el evento Resize no recibe información acerca de la posición del control. Si el control contenedor **OLE** sale del formulario, los argumentos tendrán valores negativos o positivos que representan la posición del objeto en relación con la parte superior e izquierda del formulario.

SE APLICA A:

CONSULTAR PROPIEDADES METODOS Y EVENTOS DE Contenedor OLE (Control)

OLECompleteDrag (Evento)

Vea también Ejemplo Se aplica a

Se produce cuando un componente de origen se coloca en un componente de destino e informa al componente de origen de que se ha realizado o cancelado una operación de arrastre.

Sintaxis

Private Sub *objeto*_**OLECompleteDrag**([*efecto* **As Long**])

La sintaxis del evento OLECompleteDrag consta de las partes siguientes:

Parte Descripción

objeto Expresión de objeto que da como resultado un objeto de la lista Se aplica a.

efecto Entero largo establecido por el objeto origen que identifica la acción realizada, para permitir al origen efectuar la acción apropiada si se ha movido el componente (como eliminar los datos si se han movido de un componente a otro). Los valores posibles se indican en Valores.

Valores
Los valores de *efecto* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino no puede aceptar los datos, o se ha cancelado la operación de colocación.
vbDropEffectCopy	1	El resultado de la colocación es una copia de los datos del origen en el destino. Los datos originales no se alteran por la operación de arrastre.
vbDropEffectMove	2	El resultado de la colocación es la creación de un vínculo entre los datos originales del origen y el destino del arrastre.

Comentarios
El evento OLECompleteDrag es el último que se produce en una operación OLE de arrastrar y colocar. Este evento informa al componente de origen de la acción realizada cuando el objeto se coloca en el componente de destino. El destino establece este valor a través del parámetro *efecto* del evento OLEDragDrop. Mediante este parámetro, el origen puede determinar las acciones que debe efectuar. Por ejemplo, si el objeto se ha movido al destino (**vbDropEffectMove**), el origen deberá eliminarlo después de la operación.

Si **OLEDragMode** está establecido a **Automatic**, Visual Basic se encarga del comportamiento predeterminado. Sin embargo, el evento continúa produciéndose, lo que permite al usuario completar o modificar este comportamiento.
La mayoría de los componentes admiten los eventos manuales OLE de arrastrar y colocar y algunos admiten también los automáticos.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
CoolBar (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)

OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

OLECompleteDrag (Evento, controles ActiveX)

Vea también Ejemplo Se aplica a

Se produce cuando un componente de origen se coloca en un componente de destino e informa al componente de origen de que se ha realizado o cancelado una operación de arrastre.

Sintaxis

Private Sub *objeto*_**OLECompleteDrag**([*efecto* **As Long**])

La sintaxis del evento OLECompleteDrag consta de las partes siguientes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
efecto	Un entero largo establecido por el objeto de origen que identifica la acción realizada para permitir al origen efectuar la acción apropiada si se ha movido el componente (como eliminar los datos si se han movido de un componente a otro). Los valores posibles se indican en Valores.

Valores

Los valores de *efecto* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino no puede aceptar los datos o se ha cancelado la operación de colocación.
vbDropEffectCopy	1	El resultado de la colocación es una copia de los datos del origen en el destino. Los datos originales no se alteran por la operación de arrastre.
vbDropEffectMove	2	El resultado de la colocación son datos que van a moverse del origen al destino de la operación de arrastrar y colocar. El origen de la operación arrastrar debería eliminar los datos en su propio lado antes de realizar el desplazamiento.

Comentarios

El evento OLECompleteDrag es el último que se produce en una operación OLE de arrastrar y colocar. Este evento informa al componente de origen de la acción realizada cuando el objeto se coloca en el componente de destino. El destino establece este valor a través del parámetro *efecto* del evento OLEDragDrop. Mediante este parámetro, el origen puede determinar las acciones que debe efectuar. Por ejemplo, si el objeto se ha movido al destino (**vbDropEffectMove**), el origen deberá eliminarlo después de la operación.

Si **OLEDragMode** está establecido a **Automatic**, Visual Basic se encarga del comportamiento predeterminado. Sin embargo, el evento continúa produciéndose, lo que permite al usuario completar o modificar este comportamiento.
La mayoría de los componentes admiten los eventos manuales OLE de arrastrar y colocar y algunos admiten también los automáticos.

SE APLICA A:
CoolBar (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
Multimedia MCI (Control)
RichTextBox (Control)
TextBox (Control Ligero)

OLEDragDrop (Evento)

Vea también Ejemplo Se aplica a
Se produce cuando un componente de origen se coloca en un componente de destino, si el componente de origen determina que la colocación puede realizarse.
Nota Este evento sólo se produce cuando **OLEDropMode** está establecido a **1 (Manual)**.

Sintaxis
Private Sub *objeto*_**OLEDragDrop(***datos* **As DataObject**, *efecto* **As Long**, *botón* **As Integer**, *mayúsculas* **As Integer**, *x* **As Single**, *y* **As Single)**
La sintaxis del evento OLEDragDrop consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
datos	Objeto **DataObject** que contiene los formatos que proporcionará el origen y posiblemente también los datos que corresponden a esos formatos. Si el objeto **DataObject** no contiene datos, estos se obtendrán cuando el control llame al método **GetData**. No puede usar aquí los métodos **SetData** y **Clear**.
efecto	Entero largo que establece el componente de destino y que identifica la acción realizada (si se ha llevado a cabo alguna), para permitir al origen efectuar la acción apropiada si el componente se ha movido (por ejemplo eliminar los datos en el origen). Los valores posibles de *efecto* son los indicados en Valores.
botón	Entero que actúa como un campo de bits que corresponde al estado de un botón del *mouse* (ratón) cuando se presiona. El botón primario es el bit 0, el derecho el bit 1 y el intermedio el bit 2. Estos bits corresponden a los valores 1, 2 y 4, respectivamente, e indican el estado de los botones. Pueden estar activados

algunos, todos o ninguno de estos tres bits para indicar que están presionados algunos, todos o ninguno de los botones.

mayúsculas Entero que actúa como un campo de bits que corresponde al estado de las teclas MAYÚS, CTRL y ALT cuando están presionadas. La tecla MAYÚS es el bit 0, la tecla CTRL el bit 1 y la tecla ALT el bit 2. Estos bits corresponden a los valores 1, 2 y 4, respectivamente. El parámetro *mayúsculas* indica el estado de estas teclas. Pueden estar activados algunos, todos o ninguno de los bits, para indicar que están presionadas algunas, todas o ninguna de las teclas. Por ejemplo, si están presionadas CTRL y ALT, el valor de *mayúsculas* es 6.

x,y Número que especifica la ubicación actual del puntero del *mouse* (ratón). Los valores de *x* e *y* se expresan siempre en los términos del sistema de coordenadas establecido con las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** del objeto.

Valores

Los valores de *efecto* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino de colocación no puede aceptar los datos.
vbDropEffectCopy	1	La colocación produce una copia de los datos del origen en el destino. Los datos originales no se alteran con la operación de arrastre.
vbDropEffectMove	2	La colocación produce que los datos se muevan del origen al destino. El origen del arrastre debe eliminar los datos después de la operación.
VbDropEffectScroll	-2147483648	Una máscara que indica que la ventana en la que se va a colocar se ha desplazado o se desplazará.

Comentarios

El componente ActiveX de origen siempre debe enmascarar los valores del parámetro *efecto* para asegurar la compatibilidad con futuras implementaciones de los componentes ActiveX. Actualmente sólo se utilizan tres de los 32 bits del parámetro *efecto*. Sin embargo, en futuras versiones de Visual Basic es posible que se utilice el resto. Por tanto, como precaución para evitar problemas en el futuro, los orígenes de arrastres y los destinos de colocaciones deben enmascarar estos valores de forma adecuada antes de realizar comparaciones.

Por ejemplo, un componente de origen no debe comparar *efecto* con **vbDropEffectCopy**, por ejemplo, de esta forma:

```
If Effect = vbDropEffectCopy...
```

En lugar de ello, el componente de origen debe enmascarar el valor o los valores que se buscan, de este modo:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-o bien-

```
If (Effect And vbDropEffectCopy)...
```

Esto permite la definición de nuevos efectos de colocación en versiones futuras de Visual Basic al tiempo que se mantiene la compatibilidad con el código existente.
La mayoría de los componentes admiten los eventos manuales OLE de arrastrar y colocar y algunos admiten también los automáticos.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

OLEDragOver (Evento)

Vea también Ejemplo Se aplica a
Se produce cuando un componente se arrastra sobre otro.

Sintaxis
Private Sub *objeto*_**OLEDragOver(***datos* **As DataObject**, *efecto* **As Long**, *botón* **As Integer**, *mayúsculas* **As Integer**, *x* **As Single**, *y* **As Single**, *estado* **As Integer)**
La sintaxis del evento OLEDragOver consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
datos	Objeto **DataObject** que contiene los formatos que proporcionará el origen y posiblemente también los datos que corresponden a esos formatos. Si el objeto **DataObject** no contiene datos, estos se obtendrán cuando el control llame al método **GetData**. No puede usar aquí los métodos **SetData** y **Clear**.
efecto	Entero largo que el objeto de origen establece inicialmente para identificar todos los efectos que admite. El componente de destino debe establecer correctamente

este parámetro durante este evento. El valor de *efecto* se determina realizando la operación **Or** lógica entre todos los efectos activos (indicados en Valores). El componente de destino debe comprobar estos efectos y otros parámetros para determinar las acciones apropiadas y después establecer este parámetro a uno de los efectos admitidos (especificados por el origen) para indicar las acciones que se realizarán si el usuario coloca lo seleccionado en el componente. Los valores posibles se indican en Valores.

botón Entero que actúa como un campo de bits que corresponde al estado de un botón del *mouse* cuando se presiona. El botón primario es el bit 0, el derecho el bit 1 y el intermedio el bit 2. Estos bits corresponden a los valores 1, 2 y 4, respectivamente, e indican el estado de los botones. Pueden estar activados algunos, todos o ninguno de estos tres bits, para indicar que están presionados algunos, todos o ninguno de los botones.

mayúsculas Entero que actúa como un campo de bits que corresponde al estado de las teclas MAYÚS, CTRL y ALT cuando están presionadas. La tecla MAYÚS es el bit 0, la tecla CTRL el bit 1 y la tecla ALT el bit 2. Estos bits corresponden a los valores 1, 2 y 4, respectivamente. El parámetro *mayúsculas* indica el estado de estas teclas. Pueden estar activados algunos, todos o ninguno de los bits, para indicar que están presionadas algunas, todas o ninguna de las teclas. Por ejemplo, si están presionadas CTRL y ALT, el valor de *mayúsculas* es 6.

x,y Número que especifica la posición horizontal (*x*) y vertical (*y*) actual del puntero del *mouse* dentro del formulario o control de destino. Los valores de *x* e *y* se expresan siempre en los términos del sistema de coordenadas establecido con las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** del objeto.

estado Entero que corresponde al estado de transición del control que se arrastra a un formulario o control de destino. Los valores posibles se indican en Valores.

Valores

Los valores de *efecto* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino de colocación no puede aceptar los datos.
vbDropEffectCopy	1	La colocación produce una copia de los datos del origen en el destino. Los datos originales no se alteran con la operación de arrastre.
vbDropEffectMove	2	La colocación produce que los datos se muevan del origen al destino. El origen del arrastre debe eliminar los datos después de la operación.
vbDropEffectScroll	-2147483648 (&H80000000)	Se está produciendo o está a punto de producirse un desplazamiento en el componente de destino. Este valor se utiliza junto con los otros valores. **Nota** Utilícelo sólo si va a realizar su propio desplazamiento en el componente de destino.

Los valores de *estado* son:

Constante	Valor	Descripción
vbEnter	0	Se está arrastrando el componente de origen dentro del área de un destino.
vbLeave	1	Se está arrastrando el componente de origen fuera del área de un destino.
vbOver	2	El componente de origen se ha movido de una posición del destino a otra.

Comentarios

Nota Si el parámetro *estado* es **vbLeave**, lo que indica que el puntero del *mouse* ha dejado el destino, los parámetros *x* e *y* contienen ceros.

El componente de origen siempre debe enmascarar los valores del parámetro *efecto* para asegurar la compatibilidad con futuras implementaciones de los componentes ActiveX. Actualmente sólo se utilizan tres de los 32 bits del parámetro *efecto*. Sin embargo, en futuras versiones de Visual Basic es posible que se utilice el resto. Por tanto, como precaución para evitar problemas en el futuro, los orígenes y destinos de las operaciones de arrastre y colocación deben enmascarar estos valores de forma adecuada antes de realizar comparaciones.

Por ejemplo, un componente de origen no debe comparar *efecto* con **vbDropEffectCopy**, por ejemplo, de esta forma:

```
If Effect = vbDropEffectCopy...
```

En lugar de esto, el componente de origen debe enmascarar el valor o los valores que se buscan, de este modo:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```

-o bien-

```
If (Effect And vbDropEffectCopy)...
```

Esto permite la definición de nuevos efectos de colocación en versiones futuras de Visual Basic al tiempo que se mantiene la compatibilidad con el código existente.

La mayoría de los componentes admiten los eventos manuales OLE de arrastrar y colocar y algunos admiten también los automáticos.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)

OLEGiveFeedback (Evento)

Vea también Ejemplo Se aplica a

Se produce después de cada evento OLEDragOver. OLEGiveFeedback permite al componente de origen proporcionar al usuario una indicación visual, como cambiar el cursor del *mouse* para indicar lo que ocurrirá si coloca el objeto o señalar visualmente la selección (en el componente de origen) para indicar lo que va a ocurrir.

Sintaxis

Private Sub *objeto*_**OLEGiveFeedback(***efecto* **As Long,** *cursoresPredeterminados* **As Boolean)**

La sintaxis del evento OLEGiveFeedback consta de las partes siguientes:

Parte Descripción

objeto Expresión de objeto que da como resultado un objeto de la lista Se aplica a.

efecto Entero largo establecido por el componente de destino del evento OLEDragOver y que especifica la acción a realizar si el usuario coloca en él la selección. Esto permite al origen realizar la acción adecuada (como dar una indicación visual). Los valores posibles de *efecto* se indican en Valores.

cursoresPredeterminados Valor booleano que determina si Visual Basic utiliza el cursor predeterminado del *mouse* que indica el componente o si utiliza un cursor definido por el usuario. **True** (predeterminado) = utiliza el cursor predeterminado. **False** = no utiliza el cursor predeterminado. Debe haber establecido un cursor de *mouse* con la propiedad **MousePointer** del objeto **Screen**.

Valores

Los valores de *efecto* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino de colocación no puede aceptar los datos.
vbDropEffectCopy	1	La colocación produce una copia de los datos del origen en el destino. Los datos originales no se alteran con la operación de arrastre.
vbDropEffectMove	2	La colocación produce que los datos se muevan del origen al destino. El origen del arrastre debe eliminar los datos después de la operación.
vbDropEffectScroll	-2147483648 (&H80000000)	Se está produciendo o está a punto de producirse un desplazamiento en el componente de destino. Este valor se utiliza junto con los otros valores.

Nota Utilícelo sólo si va a realizar su propio desplazamiento en el componente de destino.

Comentarios
Si no hay código en el evento OLEGiveFeedback o si se establece a **True** en el parámetro *cursoresPredeterminados*, Visual Basic establecerá automáticamente como cursor del *mouse* el cursor predeterminado que proporciona el componente.

El componente de origen siempre debe enmascarar los valores del parámetro *efecto* para asegurar la compatibilidad con futuras implementaciones de los componentes ActiveX. Actualmente sólo se utilizan tres de los 32 bits del parámetro *efecto*. Sin embargo, en futuras versiones de Visual Basic es posible que se utilice el resto. Por tanto, como precaución para evitar problemas en el futuro, los orígenes y destinos de las operaciones de arrastre y colocación deben enmascarar estos valores de forma adecuada antes de realizar comparaciones.

Por ejemplo, un componente de origen no debe comparar *efecto* con **vbDropEffectCopy**, por ejemplo, de esta forma: `If Effect = vbDropEffectCopy...`
En lugar de esto, el componente de origen debe enmascarar el valor o los valores que se buscan, de este modo:

```
If Effect And vbDropEffectCopy = vbDropEffectCopy...
```
-o bien-
```
If (Effect And vbDropEffectCopy)...
```

Esto permite la definición de nuevos efectos de colocación en versiones futuras de Visual Basic al tiempo que se mantiene la compatibilidad con el código existente.
La mayoría de los componentes admiten los eventos manuales OLE de arrastrar y colocar y algunos admiten también los automáticos.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

OLESetData (Evento)

Vea también Ejemplo Se aplica a

Se produce en un componente de origen cuando un componente de destino realiza el método **GetData** sobre el objeto **DataObject** del origen, pero antes de que se carguen los datos del formato especificado.

Sintaxis

Private Sub *objeto*_**OLESetData(***dato* **As DataObject,** *formatoDatos* **As Integer)**

La sintaxis del evento OLESetData consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
dato	Objeto **DataObject** en el que se colocan los datos solicitados. El componente llama al método **SetData** para cargar el formato solicitado.
formatoDatos	Entero que especifica el formato de los datos que solicita el componente de destino. El componente de origen utiliza este valor para determinar lo que debe cargar en el objeto **DataObject**.

Comentarios

En ciertos casos puede ser conveniente posponer la carga de los datos en el objeto **DataObject** de un componente de origen para ahorrar tiempo, especialmente si el componente de origen admite muchos formatos. Este evento permite al origen responder sólo a la solicitud de un formato de datos determinado. Cuando se llama al evento, el origen debe comprobar el parámetro *formato* para determinar qué debe cargar y ejecutar entonces el método **SetData** del objeto **DataObject** para cargar los datos que se vuelven a pasar entonces al componente de destino.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)

TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

OLEStartDrag (Evento)

Vea también Ejemplo Se aplica a

Se produce cuando se ejecuta el método **OLEDrag** de un componente o cuando un componente inicia una operación OLE de arrastrar y colocar, y la propiedad **OLEDragMode** está establecida a **Automatic**.

Este evento especifica los formatos de los datos y los efectos de colocación que admite el componente de origen. También puede usarlo para insertar datos en el objeto **DataObject**.

Sintaxis

Private Sub *objeto*_**OLEStartDrag(***datos* **As DataObject,** *efectosPermitidos* **As Long)**

La sintaxis del evento OLEStartDrag consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
dato	Objeto **DataObject** que contiene los formatos que proporcionará el origen y, opcionalmente, los datos que corresponden a esos formatos. Si el objeto **DataObject** no contiene datos, se obtendrán cuando el control llame al método **GetData**. El programador debe proporcionar los valores de este parámetro en este evento. No puede usar aquí los métodos **SetData** y **Clear**.
efectosPermitidos	Entero largo que contiene los efectos que admite el componente de origen. Los valores posibles se indican en Valores. El programador debe proporcionar los valores de este parámetro en este evento.

Valores

Los valores de *efectosPermitidos* son:

Constante	Valor	Descripción
vbDropEffectNone	0	El destino de colocación no puede aceptar los datos.
vbDropEffectCopy	1	La colocación produce una copia de los datos del origen en el destino. Los datos originales no se alteran con la operación de arrastre.
vbDropEffectMove	2	La colocación produce que los datos se muevan del origen al destino. El origen del arrastre debe eliminar los datos después de la operación.

Comentarios

El componente de origen debería usar el operador **Or** lógico en todos los valores permitidos y pasar el resultado al parámetro *efectosPermitidos*. El componente de destino puede usar este valor para determinar la acción adecuada (y qué indicación se debe dar al usuario).

El evento OLEStartDrag se produce también cuando la propiedad **OLEDragMode** del componente está establecida a **Automatic**. Esto permite agregar formatos y datos al objeto **DataObject** después de que lo ha hecho el componente. También puede anular el comportamiento predeterminado del componente si elimina el objeto **DataObject** (con el método **Clear**) y después agrega sus datos y formatos.

Puede encontrar conveniente posponer la colocación de datos en el objeto **DataObject** hasta que el componente de destino lo solicite. Esto permite al componente de origen ahorrar tiempo, al no tener que cargar múltiples formatos de datos. Cuando el destino ejecuta el método **GetData** del objeto **DataObject**, se producirá el evento OLESetData del origen si los datos solicitados no se encuentran en el objeto **DataObject**. En ese momento puede cargar los datos en el objeto **DataObject** que, a su vez, los pasará al destino.

Si el usuario no carga ningún formato en el objeto **DataObject**, se cancela la operación de arrastrar y colocar.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

PathChange (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando se cambia la ruta de acceso al establecer la propiedad **FileName** o **Path** en el código.

Sintaxis
Private Sub *objeto*_**PathChange([***índice* **As Integer])**
La sintaxis del evento PathChange consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios
Puede usar un procedimiento de evento PathChange para responder a los cambios en la ruta de un control **FileListBox**. Cuando asigna una cadena que contiene una nueva ruta a la propiedad **FileName**, el control **FileListBox** invoca al evento PathChange.

Ejemplo del evento PathChange

Este ejemplo muestra cómo actualizar un control **Label** para reflejar la ruta de acceso actual de un control **FileListBox**. Al hacer doble clic en un nombre de directorio se muestra una lista de los archivos de ese directorio en **FileListBox**; también se muestra la ruta completa del directorio en el control **Label**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **Label**, un control **DirListBox** y un control **FileListBox** y, después, presione F5. Haga doble clic en un directorio para cambiar la ruta.

```
Private Sub File1_PathChange ()
   Label1.Caption = "Ruta: " & Dir1.Path     ' Muestra la
ruta en la etiqueta.
End Sub

Private Sub Dir1_Change ()
   File1.Path = Dir1.Path                    ' Establece la ruta
del archivo.
End Sub

Private Sub Form_Load ()
   Label1.Caption = "Ruta: " & Dir1.Path     ' Muestra la
ruta en la etiqueta.
End Sub
```

PatternChange (Evento)

Vea también Ejemplo Se aplica a
Ocurre cuando el patrón de lista de archivos, como "*.*", se cambia al establecer la propiedad **FileName** o **Pattern** en el código.
Sintaxis

Private Sub *objeto*_**PatternChange([***índice* **As Integer])**
La sintaxis del evento PatternChange consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios
Puede usar un procedimiento de evento PatternChange para responder a cambios en el patrón en un control **FileListBox**. Cuando asigna una cadena que contiene un nuevo patrón a la propiedad **FileName**, el control **FileListBox** invoca el evento PathChange.

SE APLICA A:
CONSULTAR LOS METODOS PROPIEDADES Y EVENTOS DE **FileListBox (Control)**

Ejemplo del evento PatternChange

Este ejemplo actualiza un control **FileListBox** con archivos que coincidan con el patrón especificado en un control **TextBox**. Si escribe una ruta completa en el control **TextBox**, como C:\Bin*.exe, se reconocen automáticamente los componentes de ruta y patrón del texto. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox**, un control **Label**, un control **FileListBox** y un control **CommandButton** y, después, presione F5 e indique un patrón de archivo válido en **TextBox**.

```
Private Sub Form_Load ()
   Command1.Default = True          ' Establece la propiedad
Default.
   Command1.Caption = "Aceptar"     ' Establece la propiedad
Caption.
End Sub

Private Sub Command1_Click ()    ' Se hizo clic en Aceptar.
   ' El texto se descompone en los componentes de ruta y
patrón.
   File1.FileName = Text1.Text
   Label1.Caption = "Ruta: " & File1.Path
End Sub

Private Sub File1_PatternChange ()
   Text1.Text = File1.Pattern     ' Establece el texto a un
nuevo patrón.
End Sub
```

Paint (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando un objeto entero o una parte del mismo se expone después de haberse movido o ampliado, o después de haberse movido una ventana que lo estaba cubriendo.

Sintaxis

Private Sub Form_Paint( )

Private Sub *objeto*_**Paint(**[*índice* **As Integer**]**)**

La sintaxis del evento Paint consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un entero que identifica de manera única a un control si está en una matriz de controles.

Comentarios

Un procedimiento de evento Paint es útil si tiene resultado de métodos gráficos en el código. Con un procedimiento Paint puede asegurarse de que ese resultado se vuelva a dibujar cuando sea necesario.

El evento Paint se invoca cuando se utiliza el método **Refresh**. Si la propiedad **AutoRedraw** está establecida a **True**, se vuelve a dibujar de forma automática, por lo que no son necesarios los eventos Paint.

Si la propiedad **ClipControls** está establecida a **False**, los métodos gráficos del procedimiento de evento Paint afectan sólo a las áreas recién expuestas del formulario; de lo contrario, los métodos gráficos vuelven a dibujar todas las áreas del formulario no cubiertas por controles (excepto los controles **Image**, **Label**, **Line** y **Shape**).

Usar el método **Refresh** en un procedimiento de evento Resize hace que se vuelva a dibujar el objeto completo cada vez que un usuario cambia el tamaño del formulario.

Nota Usar un procedimiento de evento Paint para ciertas tareas puede provocar un evento en cascada. En general, evite usar un procedimiento de evento Paint para lo siguiente:

- Mover o ajustar el tamaño de un formulario o un control.
- Cambiar cualquier variable que afecte al tamaño o a la apariencia, como establecer la propiedad **BackColor** de un objeto.
- Invocar un método **Refresh**.

Un procedimiento de evento Resize puede ser más apropiado para algunas de estas tareas.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento Paint
Este ejemplo dibuja un diamante que se cruza con el punto medio de cada lado de un formulario y se ajusta automáticamente a medida que se cambia el tamaño del formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, después presione F5 y cambie el tamaño del formulario.

```
Private Sub Form_Paint ()
   Dim HalfX, HalfY   ' Declara variables.
   HalfX = ScaleLeft + ScaleWidth / 2    ' Establece a la
mitad del ancho.
   HalfY = ScaleTop + ScaleHeight / 2    ' Establece a la
mitad del alto.
   ' Dibuja un diamante.
   Line (ScaleLeft, HalfY) - (HalfX, ScaleTop)
   Line -(ScaleWidth + ScaleLeft, HalfY)
   Line -(HalfX, ScaleHeight + ScaleTop)
   Line -(ScaleLeft, HalfY)
End Sub

Private Sub Form_Resize
   Refresh
End Sub
```

Scroll (Evento)

Vea también Ejemplo Se aplica a
Ocurre cuando el cuadro de desplazamiento de un control **ScrollBar**, o un objeto que contenga una barra de desplazamiento, cambia de posición o se desplaza horizontal o verticalmente.

Sintaxis
Private Sub ***dataGrid*****_Scroll([***cancelar* **As Integer])**
Private Sub ***objeto*****_Scroll( )**
La sintaxis del evento Scroll consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

cancelar Determina si la operación de desplazamiento tiene éxito y se vuelve a dibujar el control **ScrollBar** o **DataGrid**, tal y como se describe en Comentarios.

Comentarios

Para un control **DataGrid**, este evento ocurre cuando el usuario desplaza horizontal o verticalmente la cuadrícula, pero antes se vuelve a dibujar la cuadrícula para mostrar el resultado de la operación de desplazamiento.

En un control **ComboBox**, este evento sólo ocurre cuando se manipulan las barras de desplazamiento en la parte desplegable del control.

Establecer *cancelar* a **True** hace que la operación de desplazamiento del control **DataGrid** falle y no ocurre la nueva operación de dibujado. Si se invoca el método **Refresh** dentro de este evento, la cuadrícula se vuelve a dibujar en su nueva organización (desplazada) incluso si *cancelar* está establecido a **True**. Sin embargo, en este caso, la cuadrícula se vuelve a dibujar de nuevo ya que la operación de desplazamiento falla y vuelve de nuevo a su posición previa.

Puede usar este evento para realizar cálculos o para manipular controles que deben estar coordinados con los cambios en curso en las barras de desplazamiento. Por el contrario, utilice el evento Change cuando desee que la actualización sólo se produzca una vez, después de que cambie un control **ScrollBar**.

Nota Evite usar una instrucción o una función **MsgBox** en este evento.

SE APLICA A:

ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
UserDocument (Objeto)

Ejemplo del evento Scroll

Este ejemplo cambia el tamaño de un control **Shape** para que corresponda al valor de una barra de desplazamiento horizontal (**HScrollBar**) a medida que arrastra el cuadro de desplazamiento por la barra de desplazamiento. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **Shape**, un control **Label** y un control **HScrollBar**. Establezca la propiedad **Index** del control **Shape** a 0 para crear una matriz de controles. Después presione F5 y mueva la barra de desplazamiento.

```
Private Sub Form_Load ()
   ' Mueve y cambia el tamaño del primer control Shape.
   Shape1(0).Move   HScroll1.Left,   HScroll1.Top   *   1.5,
HScroll1.Width, HScroll1.Height
   Label1.Caption  =  ""      '  Establece  el  título  de  la
etiqueta.
   Load Shape1(1)   ' Crea el segundo control Shape.
   ' Mueve y cambia el tamaño del segundo control Shape.
   Shape1(1).Move    Shape1(0).Left,    Shape1(0).Top,    1,
Shape1(0).Height
   Shape1(1).BackStyle = 1   ' Establece BackStyle a Opaco.
   Shape1(1).Visible = True    ' Muestra el segundo control
Shape.
   HScroll1.Min = 1    ' Establece los valores de la barra
de desplazamiento.
   HScroll1.Max = HScroll1.Width
End Sub

Private Sub HScroll1_Change ()
   Label1.Caption = "Cambiado"    ' Muestra el mensaje tras
el cambio.
End Sub

Private Sub HScroll1_Scroll ()
   Shape1(1).BackColor = &HFF0000    ' Establece el color de
Shape a azul.
   Label1.Caption  =  "Cambiando"      '  Muestra  el  mensaje
mientras se desplaza.
   Shape1(1).Width = HScroll1.Value ' Cambia el tamaño del
control Shape al valor del control de desplazamiento.
End Sub
```

Resize (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando un objeto se muestra primero o cuando cambia el estado de una ventana. (Por ejemplo, cuando se maximiza, minimiza o restaura un formulario.)

Sintaxis

Private Sub Form_Resize( )

Private Sub *objeto*_**Resize(***alto* **As Single,** *ancho* **As Single)**

La sintaxis del evento Resize consta de las siguientes partes:

Parte Descripción
objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
alto Un número que especifica el nuevo valor para el alto del control.
ancho Un número que especifica el nuevo valor para el ancho del control.

Comentarios
Utilice un procedimiento de evento Resize para mover o cambiar el tamaño de los controles cuando el formulario primario cambia de tamaño. También puede usar este procedimiento de evento para actualizar variables o propiedades, como **ScaleHeight** y **ScaleWidth**, que pueden depender del tamaño del formulario. Si desea que los gráficos mantengan tamaños proporcionales al formulario cuando éste cambia de tamaño, invoque el evento Paint mediante el método **Refresh** en un procedimiento de evento Resize.

Siempre que la propiedad **AutoRedraw** esté establecida a **False** y el formulario cambie de tamaño, Visual Basic también llama a los eventos relacionados, Resize y Paint, en ese orden. Cuando adjunte procedimientos a estos eventos relacionados, asegúrese de que sus acciones no entren en conflicto.
Cuando la propiedad **SizeMode** de un control contenedor **OLE** está establecida a 2 (Autosize), el control cambia de tamaño automáticamente de acuerdo con el tamaño de presentación del objeto contenido en el control. Si el tamaño de presentación del objeto cambia, el control cambia de tamaño automáticamente para dar cabida al objeto. Cuando esto ocurre, se invoca el evento Resize para el objeto antes de que cambie el tamaño del control contenedor **OLE**. Las partes *alto* y *ancho* indican el tamaño óptimo para mostrar el objeto (este tamaño lo determina la aplicación que creó el objeto). Puede cambiar el tamaño de un control de forma diferente si cambia los valores de las partes *alto* y *ancho* del evento Resize.

SE APLICA A:
Contenedor OLE (Control)
CoolBar (Control)
Data (Control)
DataReport (Objeto)
Form (Objeto), Forms (Colección)
MDIForm (Objeto)
PictureBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del evento Resize
Este ejemplo cambia automáticamente el tamaño de un control **TextBox** para llenar el formulario siempre que se cambie el tamaño del mismo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox**. Establezca la propiedad **MultiLine** del control **TextBox** a **True**, su propiedad **ScrollBars** a 3 y su propiedad **BorderStyle** a 0; después, presione F5 y cambie el tamaño del formulario.

```
Private Sub Form_Load ()
   Text1.Text = ""   ' Borra el texto.
End Sub

Private Sub Form_Resize ()
   Text1.Move 0,0, ScaleWidth, ScaleHeight
End Sub
```

Updated (Evento)

Vea también Ejemplo Se aplica a

Se produce cuando los datos del objeto se han modificado.

Sintaxis

Sub *objeto*_**Updated (***código* **As Integer)**

La sintaxis del evento **Updated** consta de las partes siguientes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

código Un entero que especifica cómo se ha actualizado el objeto, como se describe en Valores.

Valores

Los valores admitidos para *código* son:

Constante	Valor	Descripción
vbOLEChanged	0	Los datos del objeto han cambiado.
VbOLESaved	1	La aplicación que creó el objeto ha guardado los datos del objeto.
vbOLEClosed	2	La aplicación que creó el objeto ha cerrado el archivo que contiene los datos del objeto vinculado.
vbOLERenamed	3	La aplicación que creó el objeto ha cambiado el nombre del archivo que contiene los datos del objeto vinculado.

Comentarios

Estas constantes se enumeran en la biblioteca de objetos de Visual Basic (VB), en el Examinador de objetos. Puede usar este evento para determinar si los datos de un objeto han cambiado desde que se guardó por última vez. Para ello, defina una variable global en el evento Updated que indique que hay que guardar el objeto. Después de guardar el objeto, restablezca la variable.

SE APLICA A:

CONSULTAR LAS PROPIEDADES METODOS Y EVENTOS DE Contenedor OLE (Control)

Validate (Evento)

Vea también Ejemplo Se aplica a

Ocurre cuando el foco cambia a un (segundo) control que tiene su propiedad **CausesValidation** establecida a **True**.

Sintaxix

Private Sub *objeto*_**Validate(***mantenerFoco* **As Boolean)**

La sintaxis del evento Validate consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

mantenerFoco Valor que indica si el control pierde el foco. Establecer *mantenerFoco* con el valor **True** indica que el control mantiene el foco.

Comentarios

El evento Validate trabaja junto con la propiedad **CausesValidation** para evitar que un control pierda el foco hasta que se cumplan ciertos criterios.

Importante El evento Validate sólo ocurre cuando el control que va a recibir el foco tiene su propiedad **CausesValidation** establecida a **True**.

SE APLICA A:

CheckBox (Control Ligero)
CheckBox (Control)
ComboBox (Control Ligero)
ComboBox (Control)
CommandButton (Control Ligero)
CommandButton (Control)
Control de datos ADO
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
HScrollBar, VScrollBar (Controles Ligeros)
HScrollBar, VScrollBar (Controles)
ListBox (Control Ligero)
ListBox (Control)
MSChart (Control)
Masked Edit (Control)
Multimedia MCI (Control)
OptionButton (Control Ligero)
OptionButton (Control)
PictureBox (Control)
RichTextBox (Control)
TextBox (Control Ligero)
TextBox (Control)

Ejemplo del evento Validate y de la propiedad CausesValidation

Este ejemplo utiliza tres controles para demostrar el uso del evento Validate y de la propiedad **CausesValidation**. De manera predeterminada, la popiedad **CausesValidation** de dos de los controles TextBox está establecida a **True**. Así cuando intente cambiar el foco de un TextBox al otro, ocurre el evento Validate. Si Text1 no contiene una fecha, o si Text2 no contiene un número mayor de 10, se impide el cambio del foco. Ya que la propiedad **CausesValidation** del control Command1 está establecida a **False**, no obstante, siempre puede hacer clic en el botón Ayuda.

Para probar este ejemplo, coloque un control **CommandButton** y dos controles **TextBox** en un formulario. Pegue el código en la sección Declaraciones del formulario y ejecute el proyecto. Intente cambiar el foco presionando la tecla Tab.

```
Private Sub Form_Load()
   ' Establece la propiedad CausesValidation del botón a
False. Cuando
   ' el usuario hace clic el botón, no ocurre el evento
Validate.
   ' Establece la propiedad Caption del botón a "Ayuda".
   With Command1
      .CausesValidation = False
      .Caption = "Ayuda"
   End With

   Show
   With Text1 ' Seleccione texto de Text1 y establece el
foco en él.
      .SelLength = Len(Text1.Text)
      .SetFocus
   End With
End Sub

Private Sub Command1_Click()
   ' Proporciona la ayuda de usuario cuando hace clic en el
botón.
   MsgBox _
   "Text1 debe ser a una fecha." & VbCrLF & _
   "Text2 debe ser un número menor de 10."
End Sub
```

```
Private Sub Text1_Validate(KeepFocus As Boolean)
   ' Si el valor no es una fecha, mantiene el foco,menos
cuando el
   ' usuario hace clic en Ayuda.
   If Not IsDate(Text1.Text) Then
      KeepFocus = True
      MsgBox "Escriba una fecha en este campo.", ,
"Text1"
   End if
End Sub

Private Sub Text2_Validate(KeepFocus As Boolean)
   ' Si el valor es un número mayor de 10, mantiene el
foco.
   If Not IsNumeric(Text2.Text) Or Val(Text2.Text) > 10
Then
      KeepFocus = True
MsgBox _
"Escriba un número menor o igual que 10.", , "Text2"
   End If
End Sub
```

MÉTODOS

AddItem (Método)

Vea también Ejemplo Se aplica a

Agrega un elemento a un control **ListBox** o **ComboBox**, o agrega una fila a un control **MS Flex Grid**. No acepta argumentos con nombre.

Sintaxis

objeto.**AddItem** *elemento*, *índice*

La sintaxis del método **AddItem** consta de las siguientes partes:

Parte Descripción

objeto Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

elemento Requerido. Una expresión de cadena que especifica el elemento que se va a agregar al objeto. En el control **MS Flex Grid** solamente, use el carácter de tabulador (código de carácter 09) para separar las cadenas que desee insertar en cada columna de una fila recién agregada.

índice Opcional. Un entero que especifica la posición dentro del objeto donde se insertan el elemento o la fila nuevos. Para el primer elemento de un control **ListBox** o **ComboBox**, o para la primera fila de un control **MS Flex Grid**, *índice es* 0.

Comentarios

Si se especifica un valor válido para *índice*, *elemento* se sitúa en dicha posición dentro del *objeto*. Si se omite *índice*, *elemento* se agrega en la posición que le corresponda dentro del orden apropiado (si la propiedad **Sorted** es **True**) o al final de la lista (si **Sorted** es **False**).

Los controles **ListBox** o **ComboBox** que están enlazados a un control **Data** no aceptan el método **AddItem**.

SE APLICA A:

ComboBox (Control)

ListBox (Control)

Ejemplo del método AddItem

Este ejemplo utiliza el método **AddItem** para agregar 100 elementos a un cuadro de lista. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario con un control **ListBox** llamado List1, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg    ' Declara variables.
   Msg = "Haga clic en Aceptar para agregar 100 elementos a
su cuadro " _
         & "de lista."
   MsgBox Msg    ' Muestra el mensaje.
   For I = 1 To 100    ' Cuenta de 1 a 100.
      Entry = "Entrada " & I    ' Crea la entrada.
```

```
      List1.AddItem Entry    ' Agrega la entrada.
   Next I
   Msg = "Haga clic en Aceptar para quitar una de cada dos
entradas."
   MsgBox Msg    ' Muestra el mensaje.
   For I = 1 To 50          ' Determina cómo quitar
      List1.RemoveItem I    ' cada elemento
   Next I
   Msg = "Haga clic en Aceptar para quitar todos los
elementos "
         & "del cuadro de lista."
   MsgBox Msg    ' Muestra el mensaje.
   List1.Clear    ' Limpia el cuadro de lista.
End Sub
```

Circle (Método)

Vea también Ejemplo Se aplica a

Dibuja un círculo, una elipse o un arco en un objeto.

Sintaxis

objeto.**Circle** [**Step**] (*x, y*), *radio*, [*color, inicio, fin, aspecto*]

La sintaxis del método **Circle** consta de las partes siguientes:

Parte	Descripción
objeto	Opcional. Expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que el **Form** que tiene el enfoque es *objeto*.
Step	Opcional. Palabra clave que especifica que el centro del círculo, elipse o arco es relativo a las coordenadas actuales especificadas por las propiedades **CurrentX** y **CurrentY** de *objeto*.
(*x, y*)	Requerido. Valores de tipo **Single** que indican las coordenadas para el punto central del círculo, elipse o arco. La propiedad **ScaleMode** de *objeto* determina las unidades de medida utilizadas.
radio	Requerido. Valor de tipo **Single** que indica el radio del círculo, elipse o arco. La propiedad **ScaleMode** de *objeto* determina la unidad de medida utilizada.
color	Opcional. Valor entero de tipo **Long** que indica el color RGB del contorno del círculo. Si se omite, se utiliza el valor de la propiedad **ForeColor**. Puede usar la función **RGB** o **QBColor** para especificar el color.
inicio, fin	Opcional. Valores de precisión simple. Cuando se dibuja un arco o un círculo o una elipse parcial, *inicio* y *fin* especifican (en radianes) las posiciones de inicio y de fin del arco. El intervalo para ambos es de -2 pi radianes a 2 pi radianes. El valor predeterminado para *inicio* es 0 radianes; el valor predeterminado para *fin* es 2 * pi radianes.

aspecto Opcional. Valor de precisión simple que indica la relación de aspecto del círculo. El valor predeterminado es 1.0, lo que produce un círculo perfecto (no elíptico) en cualquier pantalla.

Comentarios

Para llenar un círculo, establezca las propiedades **FillColor** y **FillStyle** del objeto en el que se dibuja el círculo o la elipse. Sólo se puede llenar una figura cerrada. Entre las figuras cerradas se incluyen círculos, elipses o secciones circulares (arcos con líneas de radio dibujadas en ambos extremos).

Cuando dibuja un círculo o una elipse parcial, si *inicio* es negativo, **Circle** dibujará un radio hasta *inicio* y tratará el ángulo como positivo; si *fin* es negativo, **Circle** dibujará un radio hasta *fin* y tratará el ángulo como positivo. El método **Circle** siempre dibuja en dirección contraria a las agujas del reloj (positiva).

El ancho de la línea utilizada para dibujar el círculo, elipse o arco depende del valor de la propiedad **DrawWidth**. La forma en que se dibuja el círculo en el fondo depende del valor de las propiedades **DrawMode** y **DrawStyle**.

Al dibujar segmentos circulares, para dibujar un radio con un ángulo 0 (lo que produce un segmento de línea horizontal a la derecha), especifique un valor negativo pequeño para *inicio*, en lugar de cero.

Puede omitir un argumento del centro de la sintaxis, pero debe incluir la cola del argumento antes de incluir el siguiente argumento. Si omite un argumento opcional, omita la coma que sigue al último argumento especificado.

Cuando se ejecuta **Circle**, las propiedades **CurrentX** y **CurrentY** se establecen al punto central especificado por los argumentos.
Este método no puede usarse en un bloque **With...End With**.

SE APLICA A:

Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método Circle

Este ejemplo utiliza el método **Circle** para dibujar una serie de círculos concéntricos en el centro de un formulario. Para probar este ejemplo, pegue el código en la sección General de un formulario. Después, presione F5 y haga clic en el formulario.

```
Sub Form_Click ()
   Dim CX, CY, Radius, Limit   ' Declara variable.
   ScaleMode = 3   ' Establece la escala a píxeles.
   CX = ScaleWidth / 2   ' Establece la posición X.
   CY = ScaleHeight / 2   ' Establece la posición Y.
   If CX > CY Then Limit = CY Else Limit = CX
   For Radius = 0 To Limit   ' Establece el radio.
      Circle (CX, CY), Radius,RGB(Rnd * 255, Rnd * 255, Rnd
* 255)
   Next Radius
End Sub
```

Clear (Método para Clipboard, ComboBox y ListBox)

Vea también Ejemplo Se aplica a

Borra el contenido de los controles **ListBox** o **ComboBox**, o del Portapapeles del sistema.

Sintaxis

objeto.**Clear**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Los controles **ListBox** o **ComboBox** que estén enlazados a un control **Data** no aceptan el método **Clear**.

SE APLICA A:

Clipboard (Objeto)
ComboBox (Control)
ListBox (Control)

Ejemplo del método Clear

Este ejemplo utiliza el método **Clear** para borrar todos los elementos de un cuadro de lista. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario con un control **ListBox** llamado List1, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg   ' Declara variables.
   Msg = "Haga clic en Aceptar para agregar 100 elementos a
su cuadro de lista."
   MsgBox Msg   ' Muestra el mensaje.
   For I = 1 To 100   ' Cuenta de 1 a 100.
      Entry = "Entrada " & I   ' Crea la entrada.
```

```
    List1.AddItem Entry    ' Agrega la entrada.
  Next I
  Msg = "Haga clic en Aceptar para quitar cualquier otra
entrada."
  MsgBox Msg    ' Muestra el mensaje.
  For I = 1 To 50          ' Determina cómo quitar
    List1.RemoveItem I     ' uno de cada dos
  Next I    ' elementos.
  Msg = "Haga clic en Aceptar para quitar todos los
elementos " _
        & "del cuadro de lista."
  MsgBox Msg    ' Muestra el mensaje.
  List1.Clear    ' Limpia el cuadro de lista.
End Sub
```

Este ejemplo utiliza el método Clear para borrar el objeto Clipboard. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
  Const CF_BITMAP = 2     ' Define el formato de mapa de
bits.
  Dim Msg    ' Declara variable.
  On Error Resume Next    ' Establece el control de
errores.
  Msg = "Haga clic en Aceptar para cargar un mapa de bits
" _
        & "en el Portapapeles."
  MsgBox Msg    ' Muestra el mensaje.
  Clipboard.Clear    ' Limpia el Portapapeles.
  Clipboard.SetData LoadPicture("PAPEL.BMP")    ' Obtiene el
mapa de bits.
  If Err Then
    Msg = "No se encuentra el archivo .BMP."
    MsgBox Msg    ' Muestra un mensaje de error.
    Exit Sub
  End If
  Msg = "Ahora hay un archivo de mapa de bits en el
Portapapeles. " _              & "Haga clic en Aceptar para
copiar "
  Msg = Msg & "el archivo de mapa de bits desde el
Portapapeles " _
            & "al formulario."
  MsgBox Msg    ' Muestra el mensaje.
```

```
   Picture   =   Clipboard.GetData()          '  Copia   del
Portapapeles.
   Msg = "Haga clic en Aceptar para borrar la imagen."
   MsgBox Msg    ' Muestra el mensaje.
   Picture = LoadPicture()    ' Borra la imagen.
End Sub
```

Cls (Método)

Vea también Ejemplo Se aplica a

Borra los gráficos y el texto generados en tiempo de ejecución de los controles **Form** o **PictureBox**.

Sintaxis

objeto.**Cls**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que el *objeto* es el control **Form** que tenga el enfoque.

Comentarios

Cls borra el texto y los gráficos generados en tiempo de ejecución por instrucciones gráficas y de impresión. Los mapas de bits de fondo definidos mediante la propiedad **Picture** y los controles colocados en un **Form** en tiempo de diseño no se ven afectados por **Cls**. Los gráficos y el texto colocados en controles **Form** o **PictureBox** mientras la propiedad **AutoRedraw** es **True** no se ven afectados si **AutoRedraw** se establece a **False** antes de llamar a **Cls**. Es decir, puede conservar el texto y los gráficos de los controles **Form** o **PictureBox** si manipula la propiedad **AutoRedraw** del objeto con el que trabaja.

Después de llamar a **Cls**, las propiedades **CurrentX** y **CurrentY** del *objeto* se restablecen a 0.

SE APLICA A:

Form (Objeto), Forms (Colección)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método Cls

Este ejemplo utiliza el método **Cls** para eliminar la información impresa de un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Msg   ' Declara variable.
   AutoRedraw = -1   ' Activa AutoRedraw.
   ForeColor = QBColor(15)    ' Establece el color de primer
plano a blanco.
   BackColor = QBColor(1)     ' Establece el color de fondo
a azul.
   FillStyle = 7      ' Establece el relleno con barras
diagonales.
   Line (0, 0)-(ScaleWidth, ScaleHeight), , B    ' Pone un
cuadro en el
                                                  '
formulario.
   Msg = "Esta información se imprime en el fondo del
formulario."
   CurrentX = ScaleWidth / 2 - TextWidth(Msg) / 2    '
Establece la
                                                  '
posición del eje X.
   CurrentY = 2 * TextHeight(Msg)    ' Establece la posición
del eje  Y.
   Print Msg   ' Imprime el mensaje en el formulario.
   Msg = "Haga clic en Aceptar para borrar la información y
el " _
         & "patrón de fondo "
   Msg = Msg & "mostrado en el formulario."
   MsgBox Msg   ' Muestra el mensaje.
   Cls   ' Borra el fondo del formulario.
End Sub
```

Drag (Método)

Vea también Ejemplo Se aplica a

Inicia, termina o cancela una operación de arrastre de cualquier control excepto los controles **Line**, **Menu**, **Shape**, **Timer** o **CommonDialog**. No acepta argumentos con nombre.

Sintaxis

objeto.**Drag** *acción*

La sintaxis del método **Drag** consta de las siguientes partes:

Parte	Descripción
objeto	Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que se trata del objeto cuyo procedimiento de evento contiene el método **Drag**.
acción	Opcional. Una constante o un valor que especifica la acción que se va a efectuar, como se describe en Valores. Si se omite *acción*, la acción predeterminada es iniciar el arrastre del objeto.

Valores

Los valores admitidos para *acción* son:

Constante	Valor	Descripción
vbCancel	0	Cancela la operación de arrastre
vbBeginDrag	1	Inicia el arrastre del *objeto*
vbEndDrag	2	Termina el arrastre y coloca el *objeto*

Comentarios

Estas constantes se enumeran en la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos.

Sólo se requiere el método **Drag** para controlar una operación de arrastre y colocación cuando la propiedad **DragMode** del objeto tiene el valor Manual (0). Sin embargo, **Drag** puede usarse con objetos cuya propiedad **DragMode** tenga el valor Automatic (1 o **vbAutomatic**).

Si quiere que el puntero del *mouse* (ratón) cambie de forma mientras se arrastra el objeto, use las propiedades **DragIcon** o **MousePointer**. La propiedad **MousePointer** sólo se usa si no se especifica **DragIcon**.

El método **Drag** geeneralmente actúa sincronizadamente, sigificando que cada instrucción siguiente no se ejecuta hasta que la acción de arrastre ha finalizado. Sin embargo, actúa sincronizadamente si la propiedad **DragMode** del control se establece a Manual (0 o **vbManual**).

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)

FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control Ligero)
TextBox (Control)

Ejemplo del método Drag

Este ejemplo utiliza el método **Drag** para arrastrar el nombre de un archivo de mapa de bits (.bmp) a un cuadro de imagen donde se muestra el mapa de bits. Para probar este ejemplo, pegue todo el código en la sección Declaraciones de un formulario que contenga controles **DriveListBox**, **DirListBox**, **FileListBox**, **PictureBox** y **Label**.

Utilice los nombres predeterminados para todos los controles. Ajuste el tamaño y la posición de todos los controles de forma que se puedan ver y usar fácilmente. El tamaño y la posición de la etiqueta no son importantes porque cambian en tiempo de ejecución. Cuando el programa comienza, puede examinar el sistema de archivos y cargar mapas de bits. Cuando haya encontrado el mapa de bits que desea mostrar, haga clic en el nombre de archivo de ese mapa de bits y arrástrelo al cuadro de imagen.

```
Private Sub Form_Load ()
   Picture1.AutoSize = -1    ' Activa AutoSize.
   Label1.Visible = 0    ' Hace la etiqueta invisible.
   File1.Pattern  =  "*.BMP;  *.ICO;  *.WMF"      '  Establece
patrones de
                                          ' archivos.
End Sub

Private Sub Dir1_Change ()    ' Cualquier cambio en Dir1
   File1.Path = Dir1.Path    ' se refleja en File1.
End Sub

Private Sub Drive1_Change ()    ' Cualquier cambio en Drive1
   Dir1.Path = Drive1.Drive    ' se refleja en Dir1.
End Sub
```

```
Private Sub File1_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   Dim DY   ' Declara variable.
   DY = TextHeight("A")   ' Obtiene el alto de una línea.
   Label1.Move  File1.Left,  File1.Top  +  Y  -  DY  /2,
File1.Width, DY
   Label1.Drag   ' Arrastra el contorno de la etiqueta.
End Sub

Private Sub Dir1_DragOver (Source As Control, X As Single,
Y As Single, State As Integer)
   ' Cambia el puntero a no colocar.
   If State = 0 Then Source.MousePointer = 12
   ' Utiliza el puntero predeterminado del mouse.
   If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Drive1_DragOver (Source As Control, X As
Single, Y As Single, State As Integer)
   ' Cambia el puntero a no colocar.
   If State = 0 Then Source.MousePointer = 12
   ' Utiliza el puntero predeterminado del mouse.
   If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub Form_DragOver (Source As Control, X As Single,
Y As Single, State As Integer)
   ' Cambia el puntero a no colocar.
   If State = 0 Then Source.MousePointer = 12
   ' Utiliza el puntero predeterminado del mouse.
   If State = 1 Then Source.MousePointer = 0
End Sub

Private Sub File1_DragOver (Source As Control, X As Single,
Y As Single, State As Integer)
   On Error Resume Next
   If State = 0 And Right$(File1.Filename,4) = ".ICO" Then
      Label1.DragIcon  =  LoadPicture(File1.Path  +  "\"  +
File1.Filename)
   If  Err  Then  MsgBox "Imposible  cargar  el  archivo  de
iconos."
   ElseIf State = 1 Then
      Label1.DragIcon = LoadPicture ()    ' Usa icono de no
arrastrar.
   End If
End Sub
```

```
Private Sub Picture1_DragDrop (Source As Control, X As
Single, Y As Single)
   On Error Resume Next
   Picture1.Picture = LoadPicture(File1.Path + "\" +
File1.Filename)
   If Err Then MsgBox "Imposible cargar el archivo de
imagen."
End Sub
```

LinkExecute (Método)

Vea también Ejemplo Se aplica a

Envía una cadena de comandos a la aplicación de origen de una conversación DDE. No acepta argumentos con nombre.

Sintaxis

objeto.**LinkExecute** *cadena*

La sintaxis del método **LinkExecute** consta de las siguientes partes:

Parte Descripción

objeto Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

*cadena*Requerido. Una expresión de cadena que contiene un comando reconocido por la aplicación de origen.

Comentarios

El valor real de *cadena* varía dependiendo de la aplicación de origen. Por ejemplo, Microsoft Excel y Microsoft Word para Windows aceptan cadenas de comandos que constan de macrocomandos entre corchetes ([ ]). Para saber qué cadenas de comandos aceptan las aplicaciones de origen, vea la documentación de cada aplicación.

SE APLICA A:

Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo del método LinkExecute

Este ejemplo establece un vínculo DDE con Microsoft Excel, pone algunos valores en celdas de la primera fila de una nueva hoja de cálculo y representa gráficamente dichos valores. **LinkExecute** envía a Microsoft Excel el comando para activar una hoja de cálculo, selecciona algunos valores y los representa gráficamente. Para probar este ejemplo, Microsoft Excel debe estar instalado en su equipo y debe estar incluido en la instrucción Path del archivo Autoexec.bat. Pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** con el nombre predeterminado Text1, y después presione F5 y haga clic en el formulario.

BIBLIOTECA UNADECA ALAJUELA COSTA RICA

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declara variables.
   Q = Chr(34)   ' Define comillas.
   '  Crea  una  cadena  que  contenga  macrocomandos  de
Microsoft Excel.
   Cmd = "[ACTIVATE(" & Q &"HOJA1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)   ' Inicia Microsoft Excel.
      Text1.LinkTopic = "Excel|Hoja1"   ' Establece el tema
del vínculo.
      Text1.LinkItem = "L1C1"   ' Establece el elemento del
vínculo.
      Text1.LinkMode = vbLinkManual   ' Establece el modo
del vínculo.
   End If
   For I = 1 To 5
      Row = I   ' Define el número de fila.
      Text1.LinkItem = "L" & Row & "C1"   ' Establece el
elemento del
                                          ' vínculo.
      Text1.Text = Chr(64 + I)   ' Pone el valor en Text.
      Text1.LinkPoke   ' Envía el valor a la celda.
      Text1.LinkItem = "L" & Row & "C2"   ' Establece el
elemento del
                                          ' vínculo.
      Text1.Text = Row   ' Pone el valor en Text.
      Text1.LinkPoke   ' Envía el valor a la celda.
   Next I
   On Error Resume Next
   Text1.LinkExecute Cmd   ' Ejecuta comandos de Microsoft
Excel.
   MsgBox "Terminada la demostración DDE de LinkExecute con
" _
         & "Microsoft Excel.", 64
   End
End Sub
```

LinkPoke (Método)

Vea también Ejemplo Se aplica a

Transfiere el contenido de un control **Label**, **PictureBox** o **TextBox** a la aplicación de origen de una conversación DDE.

Sintaxis
objeto.**LinkPoke**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
Objeto es el nombre del control **Label**, **PictureBox** o **TextBox** involucrado en la conversación DDE como destino. Si *objeto* es un control **Label**, **LinkPoke** transfiere al origen el contenido de la propiedad **Caption**. Si *objeto* es un control **PictureBox**, **LinkPoke** transfiere al origen el contenido de la propiedad **Picture**. Si *objeto* es un control **TextBox**, **LinkPoke** transfiere al origen el contenido de la propiedad **Text**.
Normalmente, en una conversación DDE la información fluye desde el origen hacia el destino. Sin embargo, **LinkPoke** permite que un objeto de destino suministre datos al origen. No todas las aplicaciones de origen aceptan información suministrada de esta forma; si la aplicación de origen no acepta los datos, se producirá un error.

SE APLICA A:
Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo del método LinkPoke

Este ejemplo establece un vínculo DDE con Microsoft Excel, pone algunos valores en celdas de la primera fila de una nueva hoja de cálculo y representa gráficamente dichos valores. **LinkPoke** envía a la hoja de cálculo de Microsoft Excel los valores que se van a representar gráficamente. Para probar este ejemplo, Microsoft Excel debe estar instalado en su equipo y debe estar incluido en la instrucción Path del archivo Autoexec.bat. Pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** con el nombre predeterminado Text1, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declara variables.
   Q = Chr(34)   ' Define comillas.
   '  Crea  una  cadena  que  contenga  macrocomandos  de
Microsoft Excel.
   Cmd = "[ACTIVATE(" & Q &"HOJA1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)   ' Inicia Microsoft Excel.
      Text1.LinkTopic = "Excel|Hoja1"   ' Establece el tema
del vínculo.
      Text1.LinkItem = "L1C1"   ' Establece el elemento del
vínculo.
      Text1.LinkMode = vbLinkManual   ' Establece el modo
del vínculo.
```

```
    End If
    For I = 1 To 5
        Row = I   ' Define el número de fila.
        Text1.LinkItem = "L" & Row & "C1"     ' Establece el
elemento del
                                              ' vínculo.
        Text1.Text = Chr(64 + I)   ' Pone el valor en Text.
        Text1.LinkPoke   ' Envía el valor a la celda.
        Text1.LinkItem = "L" & Row & "C2"     ' Establece el
elemento del
                                              ' vínculo.
        Text1.Text = Row   ' Pone el valor en Text.
        Text1.LinkPoke   ' Envía el valor a la celda.
    Next I
    Text1.LinkExecute Cmd   ' Ejecuta comandos de Microsoft
Excel.
    On Error Resume Next
    MsgBox "Terminada la demostración DDE de LinkPoke con "
_
            & "Microsoft Excel.", 64
    End
End Sub
```

LinkRequest (Método)

Vea también Ejemplo Se aplica a

Pide a la aplicación de origen de una conversación DDE que actualice el contenido de un control **Label**, **PictureBox** o **TextBox**.

Sintaxis

objeto.**LinkRequest**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Objeto es el nombre de un control **Label**, **PictureBox** o **TextBox** involucrado en una conversación DDE como destino. **LinkRequest** hace que la aplicación de origen envíe los datos más actuales al *objeto*, actualizando la propiedad **Caption** si *objeto* es un control **Label**, la propiedad **Picture** si *objeto* es un control **PictureBox** o la propiedad **Text** si *objeto* es un control **TextBox**.

Si la propiedad **LinkMode** del *objeto* está establecida a Automatic (1 o **vbLinkAutomatic**), la aplicación de origen actualiza automáticamente *objeto* y **LinkRequest** no es necesario. Si la propiedad **LinkMode** del *objeto* está establecida a Manual (2 o **vbLinkManual**), la aplicación de origen actualiza *objeto* sólo cuando se usa

LinkRequest. Si la propiedad **LinkMode** del *objeto* está establecida a Notify (3 o **vbLinkNotify**), el origen notifica al destino de que los datos han cambiado invocando el evento LinkNotify. El destino debe entonces usar el método **LinkRequest** para actualizar los datos.

SE APLICA A:

Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo del método LinkRequest

Este ejemplo utiliza **LinkRequest** para actualizar el contenido de un cuadro de texto con los valores de una hoja de cálculo de Microsoft Excel. Para probar este ejemplo, Microsoft Excel debe estar ejecutándose en su equipo. Ponga algunos datos en las primeras celdas de la primera columna de la hoja de cálculo predeterminada (Hoja1.xls). Pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** llamado Text1, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   If Text1.LinkMode = vbNone Then     ' Prueba el modo de
vínculo.
      Text1.LinkTopic = "Excel|Hoja1"    ' Establece el tema
del vínculo.
      Text1.LinkItem = "L1C1"    ' Establece el elemento del
vínculo.
      Text1.LinkMode = vbLinkManual     ' Establece el modo
del vínculo.
      Text1.LinkRequest    ' Actualiza el cuadro de texto.
   Else
      If Text1.LinkItem  = "L1C1" Then
         Text1.LinkItem = "L2C1"
         Text1.LinkRequest       ' Actualiza el cuadro de
texto.
      Else
         Text1.LinkItem = "L1C1"
         Text1.LinkRequest       ' Actualiza el cuadro de
texto.
      End If
   End If
End Sub
```

LinkSend (Método)

Vea también Ejemplo Se aplica a

Transfiere el contenido de un control **PictureBox** a la aplicación de destino de una conversación DDE.

Sintaxis

objeto.**LinkSend**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Objeto debe ser un control **PictureBox** de un objeto **Form** que sea origen de una conversación DDE.

Cuando otras aplicaciones establecen vínculos automáticos con un objeto **Form** de su aplicación, Visual Basic las notifica cuando cambia el contenido de un control **TextBox** o **Label** del objeto **Form**. Sin embargo, Visual Basic no notifica automáticamente a una aplicación de destino DDE cuando cambia el valor de la propiedad **Picture** de un control **PictureBox** de un objeto **Form**. Como la cantidad de datos de un gráfico pueden ser muy grande y no suele tener sentido actualizar la aplicación de destino por cada cambio de píxel de la imagen, Visual Basic requiere el uso del método **LinkSend** para notificar explícitamente a las aplicaciones de destino DDE cuándo cambia el contenido de un control **PictureBox**.

SE APLICA A:

Label (Control)
PictureBox (Control)
TextBox (Control)

Move (Método)

Vea también Ejemplo Se aplica a

Mueve objetos **MDIForm** o **Form**, o controles. No acepta argumentos con nombre.

Sintaxis

objeto.**Move** *izquierda, superior, ancho, alto*

La sintaxis del método **Move** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que el *objeto* es el formulario que tenga el enfoque.
izquierda	Requerido. Un valor de precisión simple que indica la coordenada horizontal (eje x) del borde izquierdo del *objeto*.
superior	Opcional. Un valor de precisión simple que indica la coordenada vertical (eje y) del borde superior del *objeto*.

ancho Opcional. Un valor de precisión simple que indica el nuevo ancho del *objeto*.

alto Opcional. Un valor de precisión simple que indica el nuevo alto del *objeto*.

Comentarios

Sólo se requiere el argumento *izquierda*. Sin embargo, para especificar otros argumentos debe especificar todos los argumentos que aparecen en la sintaxis, delante del argumento que desee especificar. Por ejemplo, no puede especificar *ancho* sin especificar *izquierda* y *superior*. Los argumentos siguientes que no se especifiquen permanecen sin cambios.

En los formularios y en los controles dentro de un control **Frame**, el sistema de coordenadas se mide siempre en twips. El movimiento de un formulario en la pantalla o de un control dentro de un control **Frame** es siempre relativo al origen (0,0), que está en la esquina superior izquierda. Cuando se mueve un control en un objeto **Form** o en un control **PictureBox** (o un formulario MDI secundario de un objeto **MDIForm**), se usa el sistema de coordenadas del objeto contenedor. El sistema de coordenadas o unidad de medida se establece mediante la propiedad **ScaleMode** en tiempo de diseño. Puede cambiar el sistema de coordenadas en tiempo de ejecución con el método **Scale**.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
Multimedia MCI (Control)
OptionButton (Control)

PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
Shape (Control)
TextBox (Control Ligero)
TextBox (Control)

Ejemplo del método Move

Este ejemplo utiliza el método **Move** para mover un formulario por la pantalla. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Inch, Msg   ' Declara variables.
   Msg = "Haga clic en Aceptar para cambiar el tamaño y
mover este "
   Msg = Msg & "formulario cambiando el valor de algunas
propiedades."
   MsgBox Msg   ' Muestra el mensaje.
   Inch = 1440   ' Establece twips.
   Width = 4 * Inch   ' Establece el ancho.
   Height = 2 * Inch   ' Establece el alto.
   Left = 0   ' Establece el lado izquierdo en el origen.
   Top = 0   ' Establece el lado superior en el origen.
   Msg = "Ahora haga clic en Aceptar para cambiar el tamaño
y mover "
   Msg = Msg & "este formulario mediante el método Move."
   MsgBox Msg   ' Muestra el mensaje.
   Move Screen.Width - 2 * Inch, Screen.Height - Inch, 2 *
Inch, Inch
End Sub
```

OLEDrag (Método)

Vea también Ejemplo Se aplica a

Hace que un componente inicie una operación OLE de arrastrar y colocar.

Sintaxis

objeto.**OLEDrag**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Cuando se invoca al método **OLEDrag**, se produce el evento OLEStartDrag del componente, lo que le permite suministrar datos al componente de destino.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

PSet (Método)

Vea también Ejemplo Se aplica a
Establece un punto de un objeto a un color especificado.

Sintaxis
objeto.**PSet** [**Step**] (*x*, *y*), [*color*]
La sintaxis del método **PSet** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, el **Form** que tenga el enfoque se supondrá que es *objeto*.
Step	Opcional. Palabra clave que especifica que las coordenadas son relativas a la posición gráfica actual especificada por las propiedades **CurrentX** y **CurrentY**.
(*x*, *y*)	Requerido. Valores de tipo **Single** que indican las coordenadas horizontal (eje x) y vertical (eje y) del punto que se va a establecer.
color	Opcional. Valor entero de tipo **Long** que indica el color RGB especificado para el punto. Si se omite, se usará el valor actual de la propiedad **ForeColor**. Puede usar la función **RGB** o **QBColor** para especificar el color.

Comentarios
El tamaño del punto dibujado depende del valor de la propiedad **DrawWidth**. Cuando **DrawWidth** es 1, **PSet** establece un único píxel al color especificado. Cuando **DrawWidth** es mayor que 1, el punto se centra en las coordenadas especificadas.
La forma en que se dibuja el punto depende del valor de las propiedades **DrawMode** y **DrawStyle**.

Cuando se ejecuta **PSet**, las propiedades **CurrentX** y **CurrentY** se establecen al punto especificado por los argumentos.

Para borrar un único píxel con el método **PSet**, especifique las coordenadas del píxel y utilice el valor de la propiedad **BackColor** como argumento *color*.
Este método no se puede usar en un bloque **With...End With**.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método PSet
Este ejemplo utiliza el método **PSet** para dibujar confetti en un formulario. Para probar este ejemplo, pegue el código en la sección General de un formulario. Después, presione F5 y haga clic en el formulario.

```
Sub Form_Click ()
   Dim CX, CY, Msg, XPos, YPos   ' Declara variables.
   ScaleMode = 3   ' Establece ScaleMode a
         ' píxeles.
   DrawWidth = 5   ' Establece DrawWidth.
   ForeColor = QBColor(4)    ' Establece el color de primer
plano a rojo.
   FontSize = 24   ' Establece el tamaño en puntos.
   CX = ScaleWidth / 2   ' Obtiene el centro horizontal.
   CY = ScaleHeight / 2   ' Obtiene el centro vertical.
   Cls   ' Borra el formulario.
   Msg = "¡Feliz Año Nuevo!"
   CurrentX = CX - TextWidth(Msg) / 2      ' Posición
horizontal.
   CurrentY = CY - TextHeight(Msg)   ' Posición vertical.
   Print Msg   ' Imprime el mensaje.
   Do
```

```
    XPos = Rnd * ScaleWidth    ' Obtiene la posición
horizontal.
    YPos = Rnd * ScaleHeight   ' Obtiene la posición
vertical.
    PSet (XPos, YPos), QBColor(Rnd * 15)    ' Dibuja
confetti.
    DoEvents   ' Realiza otros
  Loop   ' procesos.
End Sub
```

PaintPicture (Método)

Vea también Ejemplo Se aplica a

Dibuja el contenido de un archivo gráfico (.bmp, .wmf, .emf, .cur, .ico o .dib) en un objeto **Form**, **PictureBox** o **Printer**. No acepta argumentos con nombre.

Sintaxis

objeto.**PaintPicture** *imagen*, *x1*, *y1*, *ancho1*, *alto1*, *x2*, *y2*, *ancho2*, *alto2*, *códigoOp*

La sintaxis del método **PaintPicture** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el objeto **Form** que tenga el enfoque.
imagen	Requerido. El origen del gráfico que se va a presentar en *objeto*. Debe ser la propiedad **Picture** de un objeto **Form** o control **PictureBox**.
x1, *y1*	Requerido. Valores de precisión simple que indican las coordenadas de destino (eje x y eje y) del *objeto* en el que se dibuja la *imagen*. La propiedad **ScaleMode** del *objeto* determina las unidades de medida que se usan.
ancho1	Opcional. Un valor de precisión simple que indica el ancho de destino de la *imagen*. La propiedad **ScaleMode** del *objeto* determina las unidades de medida que se usan. Si el ancho de destino es mayor o menor que el ancho de origen (*ancho2*), *imagen* se amplía o se comprime para que quepa. Si se omite, se usa el ancho de origen.
alto1	Opcional. Un valor de precisión simple que indica el alto de destino de la
imagen.	La propiedad **ScaleMode** del *objeto* determina las unidades de medida que se usan. Si el alto de destino es mayor o menor que el alto de origen (*alto2*), la *imagen* se amplía o se comprime para que quepa. Si se omite, se usa el alto de origen.
x2, *y2*	Opcional. Valores de precisión simple que indican las coordenadas (eje x y eje y) de la región de recorte dentro de la *imagen*. La propiedad **ScaleMode** del *objeto* determina las unidades de medida que se usan. Si se omite, se asume el valor 0.
ancho2	Opcional. Un valor de precisión simple que indica el ancho de la región de recorte de la *imagen*. La propiedad **ScaleMode** del *objeto* determina las

unidades de medida que se usan. Si se omite, se usa el ancho de origen total.

alto2 Opcional. Un valor de precisión simple que indica el alto de origen de una región de recorte de la *imagen*. La propiedad **ScaleMode** del *objeto* determina las unidades de medida que se usan. Si se omite, se usa el alto de origen total.

códigoOp Opcional. Un valor de tipo **Long** o código que se usa sólo con mapas de bits. Define una operación bit a bit (como **vbMergeCopy** o **vbSrcAnd**) que se realiza sobre *imagen* a medida que se dibuja sobre *objeto*. Para obtener la lista completa de constantes de operadores a nivel de bits, vea el tema Constantes de RasterOp en la Ayuda de Visual Basic. Existen algunas limitaciones del uso de códigos opcionales. Por ejemplo, no puede usar un código opcional distinto que **vbSrcCopy** si el origen es un icono o un metarchivo, y los códigos opcionales que interactúan con el patrón (o "brush" en términos SDK) como MERGECOPY, PATCOPY, PATPAINT y PATINVERT actualmente interactúan con la propiedad **FillStyle** del destino. **Nota** *CódigoOp* se utiliza para pasar una operación a nivel de bit en un mapa de bits. Colocar un valor en este argumento cuando pasa otro tipo de imagen puede provocar un error "El argumento o la llamada al procedimiento no son válidos". Ocurre en tiempo de diseño. Para evitar este error, deje el argumento *códigoOp* en blanco para cualquier tipo de imagen que no sea un mapa de bits.

Comentarios

Puede voltear un mapa de bits horizontal o verticalmente usando valores negativos para el alto de destino (*alto1*) y el ancho de destino (*ancho1*).

Puede omitir tantos argumentos finales como desee. Si omite uno o varios argumentos finales opcionales, no use comas a partir del último argumento especificado. Si desea especificar un argumento opcional, debe especificar todos los argumentos opcionales que aparecen delante del mismo en la sintaxis.

Tenga en cuenta la diferencia entre cargar un .Bmp en un control **PictureBox** y usar la función BitBlt() de la API de Windows para agregarle una imagen. Cuando realiza BitBlt en una imagen, el control **PictureBox** no sabe cómo cambiar el tamaño como lo haría si usara el método **LoadPicture**. Establecer las propiedades **ScaleWidth** y **ScaleHeight** para cambiar el tamaño de la imagen tampoco funciona. Si desea que **PictureBox** cambie al nuevo tamaño después de llamar a BitBlt, debe hacerlo manualmente mediante código, convirtiendo las unidades y trabajando con los bordes. Éste es un simple ejemplo de cómo hacer esto:

```
Sub ResizePictureBoxToImage(pic as PictureBox, twipWd _
  as Integer, twipHt as Integer)
 ' Este código asume que todas las unidades están en twips.
Si
 ' no, debe convertirlas a twips antes de llamar a esta
 ' rutina. También se asume que la imagen se
 ' ha ajustado a 0,0.
 Dim BorderHt as Integer, BorderWd as Integer
```

```
 BorderWd = Pic.Width - Pic.ScaleWidth
 BorderHt = Pic.Height - Pic.ScaleHeight
 pic.Move pic.Left, pic.Top, twipWd + BorderWd, _
   twipHt + BorderHt
End Sub
```

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Point (Método)

Vea también Ejemplo Se aplica a
Devuelve, como un valor de tipo Integer Long, el color rojo-verde-azul (RGB) del punto especificado de un objeto **Form** o un control **PictureBox**. No acepta argumentos con nombre.

Sintaxis
objeto.**Point**(*x*, *y*)
La sintaxis del método **Point** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que el *objeto* es el objeto **Form** que tenga el enfoque.
x, *y*	Requerido. Valores de precisión simple que indican las coordenadas horizontal (eje x) y vertical (eje y) del punto según la propiedad **ScaleMode** del objeto **Form** o el control **PictureBox**. Deben colocarse entre paréntesis.

Comentarios
Si el punto definido por las coordenadas *x* e *y* está fuera del *objeto*, el método **Point** devuelve -1.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método Point

Este ejemplo utiliza el método **Point** para determinar el color de un determinado punto de un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim LeftColor, MidColor, Msg, RightColor      ' Declara
variables.
   AutoRedraw = -1    ' Activa AutoRedraw.
   Height = 3 * 1440    ' Establece el alto a 3 pulgadas.
   Width = 5 * 1440    ' Establece el ancho a 5 pulgadas.
   BackColor = QBColor(1)    ' Establece el color de fondo a
azul.
   ForeColor = QBColor(4)    ' Establece el color de primer
plano a rojo.
   Line (0, 0)-(Width / 3, Height), , BF    ' Cuadro rojo.
   ForeColor = QBColor(15)    ' Establece el color de primer
plano a blanco.
   Line (Width / 3, 0)-((Width / 3) * 2, Height), , BF
   LeftColor = Point(0, 0)    ' Busca el color del cuadro de
la,
   MidColor = Point(Width / 2, Height / 2)     ' izquierda,
el centro
   RightColor = Point(Width, Height)    ' y la derecha.
   Msg = "El número de color para el cuadro rojo que hay en
la parte " _
         & "izquierda "
   Msg = Msg & "del formulario es " & LeftColor & ". El "
   Msg = Msg & "color del cuadro blanco que hay en el
centro es "
   Msg = Msg & MidColor & ". El color del cuadro azul que
hay "
   Msg = Msg & "a la derecha es " & RightColor & "."
   MsgBox Msg    ' Muestra el mensaje.
End Sub
```

PopupMenu (Método)

Vea también Ejemplo Se aplica a

Presenta un menú emergente en un objeto **MDIForm** o **Form** en la posición actual del *mouse* o en las coordenadas especificadas. No acepta argumentos con nombre.

Sintaxis

objeto.**PopupMenu** *nombreMenú, indicadores, x, y, negrita*

La sintaxis del método **PopupMenu** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que el *objeto* es el formulario que tenga el enfoque.
nombreMenú	Requerido. El nombre del menú emergente que se va a presentar. El menú especificado debe tener al menos un submenú.
indicadores	Opcional. Un valor o una constante que especifica la ubicación y el comportamiento del menú emergente, como se describe en Valores.
x	Opcional. Especifica la coordenada x donde se presenta el menú emergente. Si se omite, se usa la coordenada del *mouse* (ratón).
y	Opcional. Especifica la coordenada y donde se presenta el menú emergente. Si se omite, se usa la coordenada del *mouse* (ratón).
negrita	Opcional. Especifica el nombre de un control **Menu** del menú emergente para presentar su título en negrita. Si se omite, ningún control del menú emergente aparecerá en negrita.

Valores

Los valores admitidos para *indicadores* son:

Constante (ubicación)	Valor	Descripción
VbPopupMenuLeftAlign	0	(Predeterminado) El lado izquierdo del menú emergente se sitúa en *x*.
vbPopupMenuCenterAlign	4	El menú emergente se centra en *x*.
vbPopupMenuRightAlign	8	El lado derecho del menú emergente se sitúa en *x*.

Constante (comportamiento)	Valor	Descripción
vbPopupMenuLeftButton	0	(Predeterminado) Los elementos del menú emergente sólo reaccionan a los clics del *mouse* cuando se usa el botón primario del *mouse*.
vbPopupMenuRightButton	2	Los elementos del menú emergente reaccionan a los clics del *mouse* cuando se usan los botones primario o secundario.

Comentarios

Estas constantes se enumeran en la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos.

Las unidades de medida para las coordenadas x e y se especifican mediante la propiedad **ScaleMode**. Las coordenadas x e y definen la ubicación del menú emergente con respecto al formulario especificado. Si las coordenadas x e y no se incluyen, el menú emergente se presentará en la ubicación actual del *mouse*.

Cuando se presenta un menú emergente, el código que sigue a la llamada al método **PopupMenu** no se ejecuta hasta que el usuario elige un comando del menú (en cuyo caso se ejecuta antes el código del evento Click que el código siguiente a la instrucción **PopupMenu**) o cancela el menú. Además, sólo puede presentarse un menú emergente cada vez; por tanto, las llamadas a este método se pasan por alto si el menú emergente ya se presenta actualmente o si hay un menú desplegable abierto.

SE APLICA A:
Form (Objeto), Forms (Colección)
MDIForm (Objeto)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método PopupMenu
Este ejemplo muestra un menú emergente en la ubicación del cursor cuando el usuario hace clic con el botón secundario del *mouse* (ratón) en un formulario. Para probar este ejemplo, cree un formulario que contenga un control **Menu** llamado mnuFile (mnuFile debe tener al menos un submenú). Copie el código a la sección Declaraciones del formulario y presione F5.

```
Private Sub Form_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   If Button = 2 Then
      PopupMenu mnuFile
   End If
End Sub
```

Refresh (Método)

Vea también Ejemplo Se aplica a
Fuerza el volver a dibujar un formulario o un control completo.

Sintaxis
objeto.**Refresh**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
Utilice el método **Refresh** cuando desee:

- Presentar completamente un formulario mientras se carga otro formulario.

- Actualizar el contenido de un cuadro de lista del sistema de archivos, como un control **FileListBox**.
- Actualizar las estructuras de datos de un control **Data**.

Refresh no se puede usar con formularios MDI, pero se puede usar con formularios MDI secundarios. No puede usar **Refresh** con controles **Menu** o **Timer**.

Generalmente, el dibujo de un formulario o control se controla automáticamente cuando no ocurren eventos. Sin embargo, hay muchas situaciones en las que querrá actualizar inmediatamente un formulario o un control. Por ejemplo, si usa un cuadro de lista de archivos, un cuadro de lista de directorios o un cuadro de lista de unidades para mostrar el estado actual de la estructura de directorios, puede usar el método **Refresh** para actualizar la lista siempre que se produzca un cambio en la estructura de directorios.

Puede usar el método **Refresh** con un control **Data** para abrir o volver a abrir la base de datos (si han cambiado los valores de las propiedades **DatabaseName**, **ReadOnly**, **Exclusive** o **Connect**) y reconstruir el dynaset de la propiedad **Recordset** del control.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataReport (Objeto)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
Shape (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método Refresh

Este ejemplo usa el método **Refresh** para actualizar un control **FileListBox** cuando se crean archivos de prueba. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario con un control **FileListBox** llamado File1 y después ejecute el ejemplo y haga clic en el formulario.

```
Private Sub Form_Click ()
   ' Declara variables.
   Dim FilName, Msg as String, I as Integer
   File1.Pattern = "Prueba.*"   ' Patrón de archivos.
   For I = 1 To 8   ' Hacer el bucle ocho veces.
      FilName = "PRUEBA." & I
      ' Crea un archivo vacío.
      Open FilName For Output As FreeFile
      File1.Refresh     ' Actualiza el cuadro de lista de
archivos.
      Close   ' Cierra el archivo.
   Next I
   Msg = "Haga clic en Aceptar para quitar los archivos de
prueba creados."
   MsgBox Msg   ' Presenta un mensaje.
   Kill "PRUEBA.*"   ' Quita los archivos de prueba.
   File1.Refresh     ' Actualiza el cuadro de lista de
archivos.
End Sub
```

RemoveItem (Método)

Vea también Ejemplo Se aplica a

Quita un elemento de un control **ListBox** o **ComboBox**, o una fila de un control **MS Flex Grid**. No acepta argumentos con nombre.

Sintaxis

objeto.**RemoveItem** *índice*

La sintaxis del método **RemoveItem** consta de las siguientes partes:

Parte	Descripción
objeto	Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Requerido. Un entero que especifica la posición dentro del objeto del elemento o la fila que se va a quitar. Para los primeros elementos de los controles **ListBox** o **ComboBox**, o para la primera fila de un control **MS Flex Grid**, *índice* es 0.

Comentarios
Los controles **ListBox** o **ComboBox** que están enlazados a un control **Data** no aceptan el método **RemoveItem**.

SE APLICA A:
ComboBox (Control)
ListBox (Control)

Ejemplo del método RemoveItem

Este ejemplo utiliza el método **RemoveItem** para quitar entradas de un cuadro de lista. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ListBox** llamado List1, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim Entry, I, Msg   ' Declara variables.
   Msg = "Haga clic en Aceptar para agregar 100 elementos
al cuadro de" _
         & " lista."
   MsgBox Msg   ' Muestra el mensaje.
   For I = 1 To 100   ' Cuenta de 1 a 100.
      Entry = "Entrada " & I   ' Crea la entrada.
      List1.AddItem Entry   ' Agrega la entrada.
   Next I
   Msg = "Haga clic en Aceptar para quitar una de cada dos
entradas."
   MsgBox Msg   ' Muestra el mensaje.
   For I = 1 To 50   ' Determina cómo quitar
      List1.RemoveItem I   ' uno de cada dos
   Next I   ' elementos.
   Msg = "Haga clic en Aceptar para quitar todos los
elementos " _
         & "del cuadro de lista."
   MsgBox Msg   ' Muestra el mensaje.
   List1.Clear   ' Borra el cuadro de lista.
End Sub
```

Scale (Método)

Vea también Ejemplo Se aplica a

Define el sistema de coordenadas para un control **Form**, **PictureBox** o **Printer**. No acepta argumentos con nombre.

Sintaxis
objeto.**Scale** (*x1*, *y1*) - (*x2*, *y2*)
La sintaxis del método **Scale** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el objeto **Form** que tenga el enfoque.
x1, *y1*	Opcional. Valores de precisión simple que indican las coordenadas horizontal (eje x) y vertical (eje y) que definen la esquina superior izquierda del *objeto*. Deben ir entre paréntesis. Si se omiten, también hay que omitir el segundo conjunto de coordenadas.
x2, *y2*	Opcional. Valores de precisión simple que indican las coordenadas horizontal (eje x) y vertical (eje y) que definen la esquina inferior derecha del *objeto*. Deben ir entre paréntesis. Si se omite, también hay que omitir el primer conjunto de coordenadas.

Comentarios
El método **Scale** le permite restablecer el sistema de coordenadas a cualquier escala elegida. **Scale** afecta al sistema de coordenadas de las instrucciones gráficas en tiempo de ejecución y a la colocación de los controles.
Si utiliza **Scale** sin argumentos (si se omiten los dos conjuntos de coordenadas), restablecerá el sistema de coordenadas a twips.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método Scale

Este ejemplo utiliza el método **Scale** para establecer un sistema de coordenadas personalizado de forma que se pueda dibujar un gráfico de barras en un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim I, OldFontSize    ' Declara variables.
```

```
Width = 8640: Height = 5760    ' Establece el tamaño del
formulario en
                                    ' twips.
Move 100,100    ' Mueve el origen del formulario.
AutoRedraw = -1    ' Activa AutoRedraw.
OldFontSize = FontSize    ' Guarda el tamaño de fuente
anterior.
BackColor = QBColor(7)    ' Establece el color de fondo a
gris.
Scale (0, 110)-(130, 0)    ' Establece el sistema de
coordenadas
                                ' personalizado.
For I = 100 To 10 Step -10
   Line (0, I)-(2, I)    ' Dibuja marcas de escala cada
10 unidades.
   CurrentY = CurrentY + 1.5    ' Mueve la posición del
cursor.
   Print I    ' Imprime el valor de marca de escala a la
izquierda.
   Line (ScaleWidth - 2, I)-(ScaleWidth, I)
   CurrentY = CurrentY + 1.5    ' Mueve la posición del
cursor.
   CurrentX = ScaleWidth - 9
   Print I    ' Imprime el valor de marca de escala a la
derecha.
Next I
' Dibuja un gráfico de barras.
Line (10, 0)-(20, 45), RGB(0, 0, 255), BF    ' Primera
barra azul.
Line (20, 0)-(30, 55), RGB(255, 0, 0), BF    ' Primera
barra roja.
Line (40, 0)-(50, 40), RGB(0, 0, 255), BF
Line (50, 0)-(60, 25), RGB(255, 0, 0), BF
Line (70, 0)-(80, 35), RGB(0, 0, 255), BF
Line (80, 0)-(90, 60), RGB(255, 0, 0), BF
Line (100, 0)-(110, 75), RGB(0, 0, 255), BF
Line (110, 0)-(120, 90), RGB(255, 0, 0), BF
CurrentX = 18: CurrentY = 100    ' Mueve la posición del
cursor.
FontSize = 14    ' Aumenta la fuente para el título.
Print "Ventas trimestrales"    ' Imprime el título.
FontSize = OldFontSize    ' Restaura el tamaño de fuente.
CurrentX = 27: CurrentY = 93    ' Mueve la posición del
cursor.
Print "Previsto frente a real"    ' Imprime el subtítulo.
Line (29, 86)-(34, 88), RGB(0, 0, 255), BF    ' Imprime
leyenda.
```

```
    Line (43, 86)-(49, 88), RGB(255, 0, 0), BF
End Sub
```

ScaleX, ScaleY (Métodos)

Vea también Ejemplo Se aplica a

Convierten el valor del ancho o el alto de un control **Form**, **PictureBox** o **Printer** de una unidad de medida de la propiedad **ScaleMode** a otra. No aceptan argumentos con nombre.

Sintaxis

objeto.**ScaleX** (*ancho*, *escalaOrigen*, *escalaDestino*)
objeto.**ScaleY** (*alto*, *escalaOrigen*, *escalaDestino*)

La sintaxis de los métodos **ScaleX** y **ScaleY** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el objeto **Form** que tenga el enfoque.
ancho	Requerido. Especifica, para *objeto*, el número de unidades de medida que se van a convertir.
alto	Requerido. Especifica, para *objeto*, el número de unidades de medida que se van a convertir.
escalaOrigen	Opcional. Una constante o un valor que especifica el sistema de coordenadas de origen de *ancho* o *alto* del *objeto* que se va a convertir, como se describe en Valores. Los posibles valores de *escalaOrigen* son los mismos que para la propiedad **ScaleMode**, así como el nuevo valor HiMetric.
escalaDestino	Opcional. Una constante o un valor que especifica el sistema de coordenadas de destino de *ancho* o *alto* del *objeto* que se va a convertir, como se describe en Valores. Los posibles valores de *escalaDestino* son los mismos que para la propiedad **ScaleMode**, así como el nuevo valor HiMetric.

Valores

Los valores admitidos para *escalaOrigen* y *escalaDestino* son:

Constante	Valor	Descripción
vbUser	0	Definido por el usuario: indica que el ancho o el alto del *objeto* tiene un valor personalizado.
vbTwips	1	Twip (1440 twips por pulgada lógica; 567 twips por centímetro lógico).
vbPoints	2	Punto (72 puntos por pulgada lógica).
vbPixels	3	Píxel (unidad mínima de resolución del monitor o la impresora).
vbCharacters	4	Carácter (horizontal = 120 twips por unidad; vertical = 240 twips por unidad).

vbInches	5	Pulgada.
vbMillimeters	6	Milímetro.
vbCentimeters	7	Centímetro.
vbHimetric	8	HiMetric. Si se omite *escalaOrigen*, HiMetric es el valor predeterminado.
vbContainerPosition	9	Determina la posición del control.
vbContainerSize	10	Determina el tamaño del control.

Comentarios
Los métodos **ScaleX** y **ScaleY** toman un valor (*ancho* o *alto*), con su unidad de medida especificada en *escalaOrigen*, y lo convierte al valor correspondiente en las unidades de medida especificadas por *escalaDestino*.
También puede usar **ScaleX** y **ScaleY** con el método **PaintPicture**.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

SetFocus (Método)

Vea también Ejemplo Se aplica a
Mueve el enfoque al control o formulario especificado.

Sintaxis
objeto.**SetFocus**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
El objeto debe ser un objeto **Form**, un objeto **MDIForm** o un control que pueda recibir el enfoque. Después de invocar el método **SetFocus**, cualquier entrada del usuario se dirige al formulario o al control especificado.

El enfoque sólo se puede mover a un formulario o un control visible. Como un formulario y los controles de un formulario no son visibles hasta que el evento Load del formulario ha terminado, no puede usar en su propio evento Load el método **SetFocus** para mover el enfoque al formulario que se está cargando a menos que use primero el método **Show**

para mostrar el formulario antes de que el procedimiento de evento Form_Load haya terminado.

Tampoco puede mover el enfoque a un formulario o un control si su propiedad **Enabled** es **False**. Si la propiedad **Enabled** se ha establecido a **False** en tiempo de diseño, primero debe establecerla a **True** antes de poder recibir el enfoque mediante el método **SetFocus**.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control Ligero)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

ShowWhatsThis (Método)

Vea también Ejemplo Se aplica a

Muestra un tema seleccionado de un archivo de Ayuda utilizando el menú emergente **¿Qué es esto?** que ofrece la Ayuda de Windows 95.

Sintaxis

objeto.**ShowWhatsThis**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
El método **ShowWhatsThis** resulta muy útil para proporcionar Ayuda interactiva en un menú contextual de una aplicación. Este método muestra el tema indicado por la propiedad **WhatsThisHelpID** del objeto especificado en la sintaxis.

SE APLICA A:
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control Ligero)
TextBox (Control)

Ejemplo del método ShowWhatsThis
Este ejemplo muestra el tema de Ayuda ¿Qué es esto? de un control **CommandButton** seleccionando un comando de un menú contextual creado para el botón. Establezca la propiedad **WhatsThisHelp** del formulario a **True**. Coloque un control **CommandButton** en un formulario, cree un menú utilizando el Editor de menús con un elemento invisible al más alto nivel llamado mnuBtnContextMenu, y un submenú llamado mnuBtnWhatsThis con el título "¿Qué es esto?".

```
Private ThisControl As Control

Private Sub Command1_MouseUp(Button As Integer, Shift As
Integer, X As Single, Y As Single)
      If Button = vbRightButton Then
         Set ThisControl = Command1
         PopupMenu mnuBtnContextMenu
      End If
   Set ThisControl = Nothing
End Sub

Private Sub mnuBtnWhatsThis_Click()
   ThisControl.ShowWhatsThis
End Sub
```

TextHeight (Método)

Vea también Ejemplo Se aplica a

Devuelve el alto de una cadena de texto tal y como se imprimiría con la fuente actual de un objeto **Form**, **PictureBox** o **Printer**. No acepta argumentos con nombre.

Sintaxis

objeto.**TextHeight**(*cadena*)

La sintaxis del método **TextHeight** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el objeto **Form** que tenga el enfoque.
cadena	Requerido. Una expresión de cadena que da como resultado una cadena para la que se quiere determinar el alto del texto. Debe ir entre paréntesis.

Comentarios

El alto se expresa en los términos del valor de la propiedad **ScaleMode** o del sistema de coordenadas del método **Scale** vigente para *objeto*. Utilice **TextHeight** para determinar la cantidad de espacio vertical necesario para presentar el texto. El alto devuelto incluye el espacio de separación normal por encima y por debajo del texto, de forma que puede usar el alto para calcular y colocar varias líneas de texto dentro de *objeto*.

Si *cadena* contiene retornos de carro, **TextHeight** devuelve el alto de las líneas acumuladas, incluyendo el espacio que hay por encima y por debajo de cada línea.

SE APLICA A:

Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)

UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método TextHeight

El método **TextHeight** se utiliza para centrar una línea de texto verticalmente en un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim HalfWidth, HalfHeight, Msg   ' Declara variable.
   AutoRedraw = -1   ' Activa AutoRedraw.
   BackColor = QBColor(4)   ' Establece el color de fondo.
   ForeColor = QBColor(15)   ' Establece el color de primer
plano.
   Msg = "Visual Basic"   ' Crea el mensaje.
   FontSize = 48   ' Establece el tamaño de fuente.
   HalfWidth =  TextWidth(Msg) / 2     ' Calcula la mitad
del ancho.
   HalfHeight = TextHeight(Msg) / 2    ' Calcula la mitad
del alto.
   CurrentX = ScaleWidth / 2 - HalfWidth    ' Establece el
eje X.
   CurrentY = ScaleHeight / 2 - HalfHeight   ' Establece el
eje Y.
   Print Msg   ' Imprime el mensaje.
End Sub
```

TextWidth (Método)

Vea también Ejemplo Se aplica a

Devuelve el ancho de una cadena de texto, tal como se imprimiría con la fuente actual de un objeto **Form**, **PictureBox** o **Printer**. No acepta argumentos con nombre.

Sintaxis

objeto.**TextWidth**(*cadena*)

La sintaxis del método **TextWidth** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el objeto **Form** que tenga el enfoque.
cadena	Requerido. Una expresión de cadena que da como resultado una cadena para la que se quiere determinar el ancho del texto. Debe ir entre paréntesis.

Comentarios
El ancho se expresa en los términos del valor de la propiedad **ScaleMode** o del sistema de coordenadas del método **Scale** vigente para *objeto*. Utilice **TextWidth** para determinar la cantidad de espacio horizontal necesario para presentar el texto. Si *cadena* contiene retornos de carro, **TextWidth** devuelve el ancho de la línea más larga.

SE APLICA A:
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo del método TextWidth

El método **TextWidth** se utiliza para centrar una línea de texto horizontalmente en un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim HalfHeight, HalfWidth, Msg   ' Declara variables.
   AutoRedraw = -1   ' Activa AutoRedraw.
   BackColor = QBColor(4)   ' Establece el color de fondo.
   ForeColor = QBColor(15)   ' Establece el color de primer
plano.
   Msg = "Visual Basic"   ' Crea el mensaje.
   FontSize = 48   ' Establece el tamaño de fuente.
   HalfWidth = TextWidth(Msg) / 2    ' Calcula la mitad del
ancho.
   HalfHeight = TextHeight(Msg) / 2    ' Calcula la mitad
del alto.
   CurrentX = ScaleWidth / 2 - HalfWidth    ' Establece el
eje X.
   CurrentY = ScaleHeight / 2 - HalfHeight    ' Establece el
eje Y.
   Print Msg   ' Imprime el mensaje.
End Sub
```

ZOrder (Método)

Vea también Ejemplo Se aplica a

Coloca un objeto **MDIForm** o **Form**, o un control al principio o al final del orden z dentro de su nivel gráfico. No acepta argumentos con nombre.

Sintaxis

objeto.**ZOrder** *posición*

La sintaxis del método **ZOrder** consta de las siguientes partes:

Parte	Descripción
objeto	Opcional. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supone que *objeto* es el formulario que tenga el enfoque.
posición	Opcional. Un entero que indica la posición del *objeto* con respecto a otras instancias del mismo *objeto*. Si posición es 0 o se omite, *objeto* se coloca al principio del orden z. Si posición es 1, *objeto* se coloca al final del orden z.

Comentarios

El orden z de los objetos se puede establecer en tiempo de diseño si elige los comandos **Traer al frente** o **Enviar hacia atrás** en el menú **Formato**, submenú **Orden**.

Dentro de un objeto **MDIForm**, **ZOrder** envía los formularios MDI secundarios al frente o hacia atrás del área cliente MDI, dependiendo del valor de *posición*. En objetos **MDIForm** o **Form**, **ZOrder** envía el formulario al frente o al fondo de la pantalla, dependiendo del valor de *posición*. Como resultado, los formularios pueden presentarse delante o detrás de otras aplicaciones en ejecución.

Hay tres niveles gráficos asociados con los formularios y los contenedores. El nivel del fondo es el espacio de dibujo donde se presentan los resultados de los métodos gráficos. El siguiente es el nivel medio donde se presentan objetos gráficos y controles **Label**. El nivel frontal es donde se presentan todos los controles no gráficos como **CommandButton**, **CheckBox** o **ListBox**. Todo lo que esté contenido en un nivel más cercano al frontal cubre lo que se encuentre en los niveles que hay detrás de ellos. **ZOrder** organiza los objetos solamente dentro del nivel donde se presenta el objeto.

SE APLICA A:

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)

DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
Shape (Control)
TextBox (Control Ligero)
TextBox (Control)

PROPIEDADES

Alignment (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina la alineación de un control **CheckBox** u **OptionButton**, del texto de un control o de los valores de una columna de un control **DataGrid**. Es de sólo lectura en tiempo de ejecución para los controles **CheckBox**, **OptionButton** y **TextBox**.

Sintaxis

objeto.**Alignment** [= *número*]

La sintaxis de la propiedad **Alignment** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
Número	Un entero que especifica el tipo de alineación, tal como se describe en Valores.

Valores

En los controles **CheckBox** y **OptionButton**, los valores admitidos para *número* son:

Constante	Valor	Descripción
vbLeftJustify	0	(Predeterminado) El texto está alineado a la izquierda y el control está alineado a la derecha.
vbRightJustify	1	El texto está alineado a la derecha y el control está alineado a la izquierda.

En los controles **Label** y **TextBox**, los valores admitidos para *número* son:

Constante	Valor	Descripción
vbLeftJustify	0	(Predeterminado) El texto está alineado a la izquierda.
VbRightJustify	1	El texto está alineado a la derecha.
vbCenter	2	El texto está centrado.

En la columna de un control **DataGrid**, los valores admitidos para *número* son:

Constante	Valor	Descripción
dbgLeft	0	El texto está alineado a la izquierda.
dbgRight	1	El texto está alineado a la derecha.
dbgCenter	2	El texto está centrado.
dbgGeneral	3	(Predeterminado) General: el texto está alineado a la izquierda y los números están alineados a la derecha.

Comentarios

Puede presentar texto a la izquierda o derecha de los controles **OptionButton** y **CheckBox**. De forma predeterminada, el texto está alineado a la izquierda.

La propiedad **MultiLine** en un control **TextBox** debe estar establecida a **True** para que la propiedad **Alignment** funcione correctamente. Si el valor de la propiedad **MultiLine** de un control **TextBox** es **False**, la propiedad **Alignment** se pasa por alto.

SE APLICA A
CheckBox (Control)
Function (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
OptionButton (Control)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)

Align (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si un objeto se presenta en cualquier tamaño y posición dentro de un formulario o si se presenta en la parte superior, inferior, izquierda o derecha del mismo y si su tamaño se ajusta automáticamente al ancho del formulario.

Sintaxis

objeto.**Align** [= *número*]

La sintaxis de la propiedad **Align** consta de la siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Número Un entero que especifica cómo se presenta el objeto, según se describe en Valores.

Valores

Los valores admitidos para *número* son:

Constante	Valor	Descripción
vbAlignNone	0	(Predeterminado en formularios no MDI) Ninguno: el tamaño y la posición pueden establecerse en tiempo de diseño o en el código. Este valor se pasa por alto si el objeto está en un formulario MDI.
vbAlignTop	1	(Predeterminado en formularios MDI) Arriba: el objeto está en la parte superior del formulario y su ancho es igual al valor de la propiedad **ScaleWidth** del formulario.
VbAlignBottom	2	Abajo: el objeto está en la parte inferior del formulario y su ancho es igual al valor de la propiedad **ScaleWidth** del formulario.
vbAlignLeft	3	Izquierda: el objeto está en la parte izquierda del formulario y su ancho es igual al valor de la propiedad **ScaleWidth** del formulario.
vbAlignRight	4	Derecha: el objeto está en la parte derecha del formulario y su ancho es igual al valor de la propiedad **ScaleWidth** del formulario.

Comentarios

Puede usar la propiedad **Align** para crear rápidamente una barra de herramientas o una barra de estado en la parte superior o inferior de un formulario. Cuando un usuario cambie el tamaño del formulario, un objeto con la propiedad **Align** establecida a 1 ó 2 modifica su tamaño automáticamente para ajustarse al ancho del formulario.

Los controles **PictureBox** y **Data** son los únicos controles estándar que se pueden colocar en un formulario MDI. El área interna de un formulario MDI está definida por el espacio no cubierto por controles. Cuando un formulario secundario MDI se maximiza dentro del formulario MDI primario, no cubrirá ningún control.

Use los valores de *número* 3 y 4 para alinear barras de herramientas a los lados izquierdo y derecho de un formulario o de un formulario MDI. Si en la esquina de un formulario MDI hay dos barras de herramientas, la alineada arriba o abajo es la que se extiende hasta la esquina, teniendo preferencia sobre las alineadas a la izquierda o a la derecha. Los objetos alineados a la izquierda o a la derecha ocupan el área interna de un formulario MDI, de igual forma que los objetos alineados arriba o abajo.

SE APLICA A

Control de datos ADO
CoolBar (Control)
Data (Control)
Extender (Objeto)
PictureBox (Control)
RemoteData (Control)
VBControlExtender (Objeto)

Ejemplo de la propiedad Align

Este ejemplo usa un control **PictureBox** como barra de herramientas en un objeto **MDIForm**, con un control **CommandButton** para mover el control **PictureBox** desde la parte superior a la parte inferior del formulario. Para probar este ejemplo, cree un nuevo **MDIForm** y establezca la propiedad **MDIChild** de Form1 a **True**. Dibuje un control **PictureBox** en el **MDIForm** y coloque un control **CommandButton** en el control **PictureBox**. Pegue el código en la sección Declaraciones del **MDIForm** y después presione F5. Haga clic en el **CommandButton** para mover el **PictureBox**.

```
Private Sub Command1_Click ()
   If Picture1.Align = vbAlignTop Then
      Picture1.Align = vbAlignBottom
      ' Alinear en la parte inferior del formulario.
   Else
      Picture1.Align = vbAlignTop
      ' Alinear en la parte superior del formulario.
   End If
End Sub
```

Appearance (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el estilo de dibujo de los controles de un objeto **MDIForm** o **Form** en tiempo de diseño. Es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**Appearance**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Valores

Los valores admitidos para la propiedad **Appearance** son:

Valor	Descripción
0	Plano. Dibuja los controles y los formularios sin efectos visuales.
1	(Predeterminado) 3D. Dibuja los controles con efectos tridimensionales.

Comentarios

Si se establece a 1 en tiempo de diseño, la propiedad **Appearance** dibujará los controles con efectos tridimensionales. Si la propiedad **BorderStyle** del formulario tiene el valor Doble fijo (**vbFixedDouble**, o 3), el título y el borde del formulario también se dibujarán con efectos tridimensionales. El valor 1 en **Appearance** también hace que el formulario y sus controles tengan establecido en su propiedad **BackColor** el color seleccionado para **Objetos 3D** en la ficha **Apariencia** del cuadro de diálogo **Propiedades de Pantalla** del sistema operativo.

Al establecer a 1 la propiedad **Appearance** de un objeto **MDIForm**, sólo se ve afectado el formulario MDI primario. Para mostrar efectos tridimensionales en los formularios MDI secundarios, deberá establecer a 1 la propiedad **Appearance** de cada uno de ellos.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)

MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Archive, Hidden, Normal, System (Propiedades)

Vea también Ejemplo Se aplica a

Devuelven o establecen un valor que determina si un control **FileListBox** presenta los archivos con los atributos Modificado, Oculto, Normal o Sistema.

Sintaxis

objeto.**Archive** [= *booleano*]
objeto.**Hidden** [= *booleano*]
objeto.**Normal** [= *booleano*]
objeto.**System** [= *booleano*]

Las sintaxis de las propiedades **Archive**, **Hidden**, **Normal** y **System** constan de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica el tipo de archivos presentados, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado para **Archive** y **Normal**) Presenta archivos con el atributo de la propiedad en el control **FileListBox**.
False	(Predeterminado para **Hidden** y **System**) Presenta archivos sin el atributo de la propiedad en el control **FileListBox**.

Comentarios

Use estas propiedades para especificar el tipo de archivos que desea presentar en un control **FileListBox**, basándose en los atributos de archivos estándar usados en el entorno operativo. Establecer estas propiedades mediante código en tiempo de ejecución hace que el control **FileListBox** presente sólo aquellos archivos que tengan los atributos especificados.

Por ejemplo, en una operación de buscar y reemplazar se podrían presentar solamente archivos del sistema si estableciera la propiedad **System** a **True** y las demás propiedades a **False**. O bien, como parte del procedimiento de copia de seguridad, podría establecer la propiedad **Archive** a **True** para mostrar únicamente los archivos modificados desde que se hizo la última copia de seguridad.

PROPIEDADES DE FileListBox (Control)

Appearance (Propiedad)
Archive, Hidden, Normal, System (Propiedades)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
FileName (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
Locked (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
MultiSelect (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Path (Propiedad)
Pattern (Propiedad)
ReadOnly (Propiedad)
Selected (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)

WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE FileListBox (Control)
Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE FileListBox (Control)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
PathChange (Evento)
PatternChange (Evento)
Scroll (Evento)
Validate (Evento)

AutoRedraw (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece el resultado de un método gráfico como gráfico persistente.

Sintaxis
objeto.**AutoRedraw** [= *booleano*]
La sintaxis de la propiedad **AutoRedraw** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

booleano Una expresión booleana que especifica la forma en que se vuelve a dibujar objeto, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor Descripción

True Activa el redibujado automático de un objeto **Form** o un control **PictureBox**. Los gráficos y el texto se escriben en la pantalla y en una imagen almacenada en memoria. El objeto no recibe eventos Paint; se vuelve a dibujar cuando es necesario, usando la imagen almacenada en memoria.

False (Predeterminado) Desactiva el redibujado automático de un objeto y escribe los gráficos o el texto solamente en la pantalla. Visual Basic invoca el evento Paint del objeto cuando se necesita volver a dibujar dicho objeto.

Comentarios

Esta propiedad es importante cuando se trabaja con los siguientes métodos gráficos: **Circle**, **Cls**, **Line**, **Point**, **Print** y **PSet**. Al establecer **AutoRedraw** a **True**, el resultado de estos métodos se vuelve a dibujar automáticamente en un objeto **Form** o en un control **PictureBox** cuando, por ejemplo, el objeto cambia de tamaño o se vuelve a presentar después de haber estado oculto por otro objeto.

Puede establecer **AutoRedraw** en el código en tiempo de ejecución para alternar entre dibujar gráficos persistentes (como color de fondo o cuadrícula) y gráficos temporales. Si establece **AutoRedraw** a **False**, el resultado anterior se convierte en parte del fondo de la pantalla. Cuando **AutoRedraw** se establece a **False**, los gráficos de fondo no se eliminan si borra el área de dibujo con el método **Cls**. Al volver a establecer **AutoRedraw** a **True** y después usar **Cls** se borran los gráficos de fondo.

Nota Si establece la propiedad **BackColor**, todos los gráficos y el texto, incluyendo los gráficos persistentes, se borran. En general, todos los gráficos deben presentarse mediante el evento Paint a menos que **AutoRedraw** esté establecida a **True**.

Para recuperar el gráfico persistente creado cuando **AutoRedraw** estaba establecida a **True**, use la propiedad **Image**. Para pasar el gráfico persistente a una API de Windows cuando **AutoRedraw** tiene el valor **True**, use la propiedad **hDC** del objeto.

Si establece la propiedad **AutoRedraw** de un formulario a **False** y después minimiza el formulario, las propiedades **ScaleHeight** y **ScaleWidth** se establecen de acuerdo con el tamaño del icono. Cuando **AutoRedraw** tiene el valor **True**, **ScaleHeight** y **ScaleWidth** conservan el tamaño de la ventana restaurada.

Si **AutoRedraw** está establecida a **False**, el método **Print** no dibujará encima de los controles gráficos como los controles **Image** y **Shape**.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad AutoRedraw

Este ejemplo presenta alternativamente dos gráficos en un control **PictureBox**: un círculo relleno persistente y líneas verticales temporales. Haga clic en el **PictureBox** para dibujar o redibujar las líneas. Al cambiar el tamaño del formulario se vuelve a dibujar el gráfico temporal. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que tenga un control **PictureBox** llamado Picture1. Presione F5 para ejecutar el programa y haga clic en el gráfico cada vez que cambie el tamaño del formulario.

```
Private Sub Form_Load ()
   Picture1.ScaleHeight = 100     ' Establece la escala a
100.
   Picture1.ScaleWidth = 100
   Picture1.AutoRedraw = True    ' Activa AutoRedraw.
   Picture1.ForeColor = 0    ' Establece ForeColor.
   Picture1.FillColor = QBColor(9)    ' Establece FillColor.
   Picture1.FillStyle = 0    ' Establece FillStyle.
   Picture1.Circle (50, 50), 30    ' Dibuja un círculo.
   Picture1.AutoRedraw = False    ' Desactiva AutoRedraw.
End Sub

Private Sub Picture1_Click ()
   Dim I    ' Declara una variable.
   Picture1.ForeColor = Rgb(Rnd * 255, 0, 0)
   ' Selecciona color aleatorio.
   For I = 5 To 95 Step 10    ' Dibuja líneas.
      Picture1.Line (I, 0)-(I, 100)
   Next
End Sub
```

AutoSize (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si el tamaño de un control cambia automáticamente para presentar todo su contenido.

Sintaxis
objeto.**AutoSize** [= *booleano*]
La sintaxis de la propiedad **AutoSize** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica si el tamaño del control se modifica, como se describe en Valores.

Valores
Los valores admitidos para *booleano* son:

Valor	Descripción
True	Cambia automáticamente el tamaño del control para presentar todo el contenido del mismo.
False	(Predeterminado) Mantiene constante el tamaño del control. El contenido se recorta cuando excede el área del control.

SE APLICA A
Label (Control)
PictureBox (Control)

BackColor, ForeColor (Propiedades)

Vea también Ejemplo Se aplica a

- **BackColor**: devuelve o establece el color de fondo de un objeto.
- **ForeColor**: devuelve o establece el color de primer plano utilizado para mostrar texto y gráficos en un objeto.

Sintaxis
objeto.**BackColor** [= *color*]
objeto.**ForeColor** [= *color*]
La sintaxis de las propiedades **BackColor** y **ForeColor** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
color	Un valor o una constante que determina el color de fondo o de primer plano del objeto, según se indica en Valores.

Valores
Visual Basic utiliza el esquema de colores rojo-verde-azul (RGB) del entorno operativo Microsoft Windows. Los valores admitidos para *color* son:

Valor Descripción
Colores RGB normales Colores especificados con la paleta de colores o en el código mediante las funciones **RGB** o **QBColor**.

Colores predeterminados del sistema Colores especificados por las constantes de color del sistema indicadas en la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos. El entorno operativo Windows sustituye las opciones especificadas por el usuario en el Panel de control.

Para todos los formularios y controles, los valores predeterminados en tiempo de diseño son:

- **BackColor**: el color predeterminado del sistema especificado por la constante **vbWindowBackground**.

- **ForeColor**: el color predeterminado del sistema especificado por la constante **vbWindowText**.

Comentarios
En los controles **Label** y **Shape**, la propiedad **BackColor** se pasa por alto cuando el valor de la propiedad **BackStyle** es 0 (transparente).
Si establece la propiedad **BackColor** de un objeto **Form** o de un control **PictureBox**, se borra todo el texto y todos los gráficos, incluidos los gráficos persistentes. Al establecer la propiedad **ForeColor**, los gráficos o el resultado impreso no se ven afectados. En el resto de los controles, el color de la pantalla cambia inmediatamente.

El intervalo válido para un color RGB normal va de 0 a 16.777.215 (&HFFFFFF). El byte más alto de los números de este intervalo es 0, y los 3 más bajos, del menos significativo al más significativo, determinan la intensidad de rojo, verde y azul, respectivamente. Cada uno de los componentes rojo, verde y azul se representa con un número entre 0 y 255 (&HFF). Si el byte más alto no es 0, Visual Basic utiliza los colores del sistema, definidos por el usuario en el Panel de control o por las constantes que se encuentran en la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos.

Para mostrar texto en el entorno operativo Windows, tanto el color del texto como el de fondo deben ser sólidos. Si no aparece el color que ha seleccionado para el texto o para el fondo, puede que uno de ellos sea de medios tonos; es decir, puede que esté formado por píxeles de hasta tres colores distintos. Si elige un color de medios tonos para el texto o el fondo, se usará el color sólido más próximo.

SE APLICA A
AmbientProperties (Objeto)
Band (Objeto)
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)

Control de datos ADO
CoolBar (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Function (Control, diseñador de informe de datos)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de las propiedades BackColor y ForeColor

En este ejemplo se restablecen aleatoriamente los colores de primer plano y de fondo dos veces por segundo en un formulario y en un control **PictureBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **PictureBox** y un control **Timer** control, y luego presione F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 500
End Sub

Private Sub Timer1_Timer ()
   BackColor = QBColor(Rnd * 15)
   ForeColor = QBColor(Rnd * 10)
   Picture1.BackColor = QBColor(Rnd * 15)
   Picture1.ForeColor = QBColor(Rnd * 10)
End Sub
```

BorderStyle (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el estilo del borde de un objeto. Para el objeto **Form** y el control **TextBox** es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**BorderStyleColor** = [*valor*]

La sintaxis de la propiedad **BorderStyleColor** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Un valor o una constante que determina el estilo del borde, según se indica en Valores.

Valores

Los valores admitidos para la propiedad **BorderStyle** para un objeto **Form** son:

Constante	Valor	Descripción
vbBSNone	0	Ninguno (ningún borde ni elemento relacionado con él).
vbFixedSingle	1	Simple fijo. Puede incluir un cuadro del menú **Control**, una barra de título, un botón Maximizar y un botón Minimizar. Sólo puede cambiar el tamaño con los botones **Maximizar** y **Minimizar**.
vbSizable	2	(Predeterminado) Tamaño ajustable. Puede cambiar el tamaño mediante cualquiera de los elementos opcionales de borde indicados para el valor 1.
vbFixedDouble	3	Diálogo fijo. Puede incluir un cuadro del menú **Control** y una barra de título, pero no los botones **Maximizar** ni **Minimizar**. No se puede cambiar de tamaño.
VbFixedToolWindow	4	Ventana fija. Muestra una ventana a la que no le puede cambiar el tamaño con un botón Cerrar y el texto de la barra de título aparece con un tamaño de fuente reducido. El formulario no aparece en la barra de tareas de Windows 95.
VbSizableToolWindow	5	Ventana de tamaño ajustable. Muestra una ventana ajustable con un botón Cerrar y el texto de la barra de título aparece con un tamaño de fuente reducido. El formulario no aparece en la barra de tareas de Windows 95.

Los valores admitidos para la propiedad **BorderStyle** para los controles **MS Flex Grid**, **Image**, **Label**, contenedor **OLE**, **PictureBox**, **Frame** y **TextBox** son los siguientes:

Valor	Descripción
0	(Predeterminado para los controles **Image** y **Label**) Ninguno.
1	(Predeterminado para los controles **MS Flex Grid**, **PictureBox**, **TextBox** y contenedor **OLE**) Simple fijo.

Los valores admitidos para la propiedad **BorderStyle** para los controles **Line** y **Shape** son los siguientes:

Constante	Valor	Descripción
vbTransparent	0	Transparente
vbBSSolid	1	(Predeterminado) Sólido. El borde se centra en el perímetro de la forma.
vbBSDash	2	Raya
vbBSDot	3	Punto
vbBSDashDot	4	Raya-punto
vbBSDashDotDot	5	Raya-punto-punto
vbBSInsideSolid	6	Sólido interior. El exterior del borde es el exterior de la forma.

Comentarios

Para un formulario, la propiedad **BorderStyle** determina características clave que lo identifican visualmente como una ventana de propósito general o como un cuadro de diálogo. El valor 3 (Fixed Dialog, Diálogo fijo) es útil para los cuadros de diálogo estándar, mientras que los valores 4 (Fixed ToolWindow, Ventana fija) y 5 (Sizable ToolWindow, Ventana de tamaño ajustable) son útiles para crear ventanas de tipo cuadro de herramientas.

Los formularios MDI secundarios con el valor 2 (Sizable, Tamaño ajustable) se muestran dentro del formulario MDI primario con un tamaño predeterminado definido por el entorno operativo Windows en tiempo de ejecución. Para los valores restantes, el formulario aparece con el tamaño especificado en tiempo de diseño.

Al modificar el valor de la propiedad **BorderStyle** de un objeto **Form** pueden alterarse los valores de las propiedades **MinButton**, **MaxButton** y **ShowInTaskbar**. Cuando **BorderStyle** se establece a 1 (Fixed Single, Simple fijo) o 2 (Sizable, Tamaño ajustable), las propiedades **MinButton**, **MaxButton** y **ShowInTaskbar** se establecen automáticamente a **True**. Cuando **BorderStyle** se establece a 0 (None, Ninguno), 3 (Fixed Dialog, Diálogo fijo), 4 (Fixed ToolWindow, Ventana fija) o 5 (Sizable ToolWindow, Ventana de tamaño ajustable), las propiedades **MinButton**, **MaxButton** y **ShowInTaskbar** se establecen automáticamente a **False**.

Nota Si para un formulario con un menú se establece a 3 (Fixed Dialog, Diálogo fijo), se mostrará con el borde del valor 1 (Fixed Single, Simple fijo).
En tiempo de ejecución, los formularios son modales o no modales, según especifique con el método **Show**.

SE APLICA A
Contenedor OLE (Control)
Form (Objeto), Forms (Colección)
Frame (Control)

Function (Control, diseñador de informe de datos)
Image (Control)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
Line (Control)
Line (Control, diseñador de informe de datos)
MSChart (Control)
PictureBox (Control)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
UserControl (Objeto)

ClipControls (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si los métodos gráficos en eventos Paint vuelven a dibujar el objeto entero o sólo las áreas recién expuestas. También determina si el entorno operativo Microsoft Windows crea un área de recorte que excluya los controles no gráficos contenidos en el objeto. Es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**ClipControls**

La sintaxis de la propiedad **ClipControls** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica la forma en que se vuelven a dibujar los objetos, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado) Los métodos gráficos de los eventos Paint vuelven a dibujar todo el objeto. Antes del evento Paint se crea un área de recorte alrededor de los controles no gráficos del formulario.
False	Los métodos gráficos de los eventos Paint sólo vuelven a dibujar las partes recién expuestas. Antes del evento Paint no se crea un área de recorte alrededor de los controles no gráficos. Los formularios complejos se cargan normalmente más rápido cuando **ClipControls** está establecida a **False**.

Comentarios
El recorte es el proceso de determinar qué partes de un formulario o contenedor, como un control **Frame** o **PictureBox**, se dibujan cuando se presenta el formulario. En memoria se crea un esquema del formulario y los controles. El entorno operativo Windows usa este esquema para dibujar algunas partes, como el fondo, sin afectar a otras partes, como el contenido de un control **TextBox**. Como el área de recorte se crea en memoria, establecer esta propiedad a **False** puede reducir el tiempo necesario para dibujar o volver a dibujar un formulario.

El área de recorte incluye la mayoría de los controles, pero no recorta alrededor de los controles **Image**, **Label**, **Line** o **Shape**.

Evite anidar controles intrínsecos cuando **ClipControls** sea **True** dentro de un control donde **ClipControls** tenga el valor **False** (por ejemplo, un botón de comando dentro de un cuadro con imagen). Este tipo de anidamiento de controles hace que los controles se vuelvan a dibujar incorrectamente. Para solucionar el problema, establezca a **True** la propiedad **ClipControls** del control contenedor y de los controles anidados.

SE APLICA A
Form (Objeto), Forms (Colección)
Frame (Control)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad ClipControls

Este ejemplo muestra cómo la propiedad **ClipControls** afecta al redibujado de un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario y después presione F5. Observe que el color del formulario cambia cada vez que se le cambia de tamaño o se le superpone otro formulario u otra aplicación. Termine el programa y establezca **ClipControls** a **False**, y después ejecute de nuevo el programa. Observe que sólo se vuelven a dibujar las partes del formulario expuestas últimamente.

```
Private Sub Form_Paint ()
   ' Selecciona un color de fondo aleatorio.
   BackColor = &HFFFFFF * Rnd
End Sub
```

Columns (Propiedad, ListBox)

Vea también Ejemplo Se aplica a
Devuelve o establece un valor que determina si un control **ListBox** se desplaza horizontal o verticalmente y cómo se presentan los elementos en las columnas. Si se desplaza horizontalmente, la propiedad **Columns** determina cuántas columnas se presentan.

Sintaxis
objeto.**Columns** [= *número*]
La sintaxis de la propiedad **Columns** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica cómo se desplaza el control y cuántos elementos se presentan en las columnas, como se describe en Valores.

Valores
Los valores admitidos para *número* son:

Valor	Descripción
0	(Predeterminado) Los elementos se presentan en una única columna y el control **ListBox** se desplaza verticalmente.
1 a *n*	Los elementos se presentan en columnas encadenadas, llenando en primer lugar la primera columna, después la segunda y así sucesivamente. El control **ListBox** se desplaza horizontalmente y presenta el número de columnas especificado.

Comentarios
En los controles **ListBox** de desplazamiento horizontal, el ancho de la columna es igual al ancho del control **ListBox**, dividido entre el número de columnas.
Esta propiedad no puede establecerse a 0 o cambiarse desde 0 en tiempo de ejecución; es decir, no puede convertir en tiempo de ejecución un control **ListBox** de múltiples columnas en un control **ListBox** de una única columna o viceversa. Sin embargo, sí es posible cambiar en tiempo de ejecución el número de columnas de un control **ListBox** de múltiples columnas.

SE APLICA A : CONSULTAR PROPIEDADES METODOS Y EVENTOS DE **ListBox (Control)**

Ejemplo de la propiedad Columns

Este ejemplo ilustra cómo funcionan dos clases diferentes de controles **ListBox** cuando contienen los mismos datos. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga dos controles **ListBox**. Establezca la propiedad **Columns** de List2 a 2, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Load ()
   Dim I    ' Declara una variable.
   List1.Move 50, 50, 2000, 1750     ' Organiza los cuadros
de lista.
   List2.Move 2500, 50, 3000, 1750
```

```
   For I = 0 To Screen.FontCount -1      ' Rellena ambos
cuadros de lista
      List1.AddItem Screen.Fonts(I)      ' con nombres de
fuentes.
      List2.AddItem Screen.Fonts(I)
   Next I
End Sub
```

Container (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el contenedor de un control de un **Form**. No está disponible en tiempo de diseño.

Sintaxis

Set *objeto*.**Container** [= *contenedor*]

La sintaxis de la propiedad **Container** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
contenedor	Una expresión de objeto que da como resultado un objeto que puede servir como contenedor de otros controles, como se describe en Comentarios.

Comentarios

Los controles siguientes pueden contener otros controles:

- Control **Frame**
- Control **PictureBox**
- Control **SSTab**

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)

MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
Multimedia MCI (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
Shape (Control)
TextBox (Control)
VBControlExtender (Objeto)

Ejemplo de la propiedad Container

Este ejemplo demuestra el movimiento de un control **CommandButton** de un contenedor al contenedor del objeto **Form**. Para probar este ejemplo, pegue el código en la sección Declaraciones de cada formulario que contiene un control **Frame**, un control **PictureBox** y un **CommandButton**, y después presione F5.

```
Private Sub Form_Click()
   Static intX As Integer
   Select Case intX
      Case 0
         Set Command1.Container = Picture1
         Command1.Top= 0
         Command1.Left= 0
      Case 1
         Set Command1.Container = Frame1
         Command1.Top= 0
         Command1.Left= 0
      Case 2
         Set Command1.Container = Form1
         Command1.Top= 0
         Command1.Left= 0
   End Select
   intX = intX + 1
End Sub
```

CurrentX, CurrentY (Propiedades)

Vea también Ejemplo Se aplica a

Devuelven o establecen las coordenadas horizontal (**CurrentX**) o vertical (**CurrentY**) para el siguiente método gráfico o de impresión. No están disponibles en tiempo de diseño.

Sintaxis
objeto.**CurrentX** [= *x*]
objeto.**CurrentY** [= *y*]

Las sintaxis de las propiedades **CurrentX** y **CurrentY** constan de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
x	Un número que especifica la coordenada horizontal.
y	Un número que especifica la coordenada vertical.

Comentarios
Las coordenadas se miden a partir de la esquina superior izquierda del objeto. El valor de la propiedad **CurrentX** es 0 en el borde izquierdo de un objeto y el valor de la propiedad **CurrentY** es 0 en su borde superior. Las coordenadas se expresan en twips o en la unidad de medida actual definida por las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft**, **ScaleTop** y **ScaleMode**.
Cuando se usan los siguientes métodos gráficos, los valores de **CurrentX** y **CurrentY** cambian de la forma siguiente:

Este método	Establece CurrentX, CurrentY como
Circle	El centro del objeto.
Cls	0, 0.
EndDoc	0, 0.
Line	El punto final de una línea.
NewPage	0, 0.
Print	La próxima posición de impresión.
Pset	El punto dibujado.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

DataChanged (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece un valor que indica que algún proceso distinto de la recuperación de datos del registro actual ha cambiado los datos del control enlazado. No está disponible en tiempo de diseño.

Sintaxis
objeto.**DataChanged** [= *valor*]
La sintaxis de la propiedad **DataChanged** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Una expresión booleana que indica si los datos han cambiado, como se describe en Valores.

Valores
Los valores admitidos para *valor* son los siguientes:

Valor	Descripción
True	Los datos que hay actualmente en el control no son iguales que los del registro actual.
False	(Predeterminado) Los datos que hay actualmente en el control (si los hay) son iguales que los del registro actual.

Comentarios
Cuando un control de datos se mueve de un registro a otro, pasa datos desde los campos del registro actual a controles enlazados al campo específico o todo el registro. Cuando se muestran datos en los controles enlazados, la propiedad **DataChanged** se establece a **False**. Si el usuario o alguna otra operación cambia el valor del control enlazado, la propiedad **DataChanged** se establece a **True**. Si pasa a otro registro la propiedad **DataChanged** no se ve afectada.

Cuando el control de datos comienza a pasar a otro registro, se produce el evento Validate. Si **DataChanged** es **True** para algún control enlazado, el control de datos invoca automáticamente los métodos **Edit** y **Update** para enviar los cambios a la base de datos.

Si no desea guardar en la base de datos los cambios de un control enlazado, puede establecer la propiedad **DataChanged** a **False** en el evento Validate.
Inspeccione el valor de la propiedad **DataChanged** en el código para el evento Change de un control con el fin de evitar eventos en cascada. Esto se aplica a controles enlazados y no enlazados.

Tipo de datos
Integer (Boolean)

SE APLICA A
Binding (Objeto)
CheckBox (Control)
ComboBox (Control)
Contenedor OLE (Control)
Extender (Objeto)
Image (Control)
Label (Control)
ListBox (Control)
Masked Edit (Control)

PictureBox (Control)
RichTextBox (Control)
TextBox (Control)
VBControlExtender (Objeto)

DataField (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el nombre de un campo al que será enlazado un receptor de datos.

Sintaxis

objeto.**DataField** [= *cadena*]

La sintaxis de la propiedad **DataField** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
cadena	Una expresión de cadena que da como resultado el nombre de uno de los campos del objeto **Recordset** especificado por un origen de datos.

Comentarios

Los controles enlazados proporcionan acceso a datos específicos de la base de datos. Los controles enlazados que administran un único campo suelen mostrar el valor un campo específico del registro actual. La propiedad **DataSource** de un control enlazado especifica un origen de datos válido y la propiedad **DataField** especifica un nombre de campo válido en el objeto **Recordset** creado por el origen de datos. Juntas, estas propiedades especifican qué datos aparecen en el control enlazado.

Cuando utiliza un objeto **QueryDef** o una instrucción SQL que devuelve los resultados de una expresión, el motor de base de datos Microsoft Jet genera automáticamente el nombre de campo . Por ejemplo, cuando codifica una función agregada de SQL o una expresión en su consulta SQL, a menos que asigne un alias a los campos agregados mediante una cláusula AS, los nombres de campo se generarán automáticamente. El nombre del campo de expresión suele ser Expr1 seguido de un número de tres dígitos que comienza por 000. La primera expresión devuelta tendría el nombre Expr1000.
Se recomienda codificar las consultas SQL para asignar alias a columnas de expresión como se muestra a continuación:

```
Adodc1.RecordSource = "Select AVG(Ventas)   " _
   & " AS VentasMedias From TablaVentas"
Text1.DataField = "VentasMedias"
Adodc1.Refresh
```

Nota Compruebe que el valor de la propiedad **DataField** es válido para cada control enlazado. Si cambia el valor de la propiedad **RecordSource** de un control de datos y a continuación utiliza **Refresh**, el objeto **Recordset** identificará el nuevo objeto. Esto puede invalidar los valores de **DataField** de controles enlazados y generar un error interceptable.

SE APLICA A
Binding (Objeto)
CheckBox (Control)
ComboBox (Control)
Contenedor OLE (Control)
Extender (Objeto)
Function (Control, diseñador de informe de datos)
Image (Control)
Label (Control)
ListBox (Control)
Masked Edit (Control)
PictureBox (Control)
RichTextBox (Control)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
VBControlExtender (Objeto)

DataFormat (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el objeto **StdDataFormat** al que está adjunto un objeto vinculado. Lectura y escritura en tiempo de diseño y en tiempo de ejecución.

Sintaxis

objeto.**DataFormat** = *objetoFormato*

La sintaxis de la propiedad **DataFormat** consta de las siguientes partes

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

objetoFormato Necesario. Un objeto **StdDataFormat**.

Comentarios

También se puede establecer la propiedad **DataFormat** mediante una página de propiedades desde la ventana Propiedades. Si el valor del código es diferente del valor de la página de propiedades, se usará el valor de la página de propiedades para el primer registro obtenido; después se usarán los valores del código.

La propiedad **DataFormat** del control **DataCombo** en una propiedad **Extender**. Por lo tanto siempre es visible en la hoja de propiedades y puede configurarse mediante código. Sin embargo el control **DataCombo** sólo da formato al primer elemento de la lista. Esto puede desconcertar al usuario final que ve el primer elemento con formato, pero se le da a seleccionar una lista de elementos sin formato. El elemento con formato también puede llevar a conclusiones erróneas al usuario final que asume que el elemento se introducirá en la base de datos con formato. Por estas razones es aconsejable no establecer la propiedad **DataFormat** cuando se utilice el control **DataCombo**.

SE APLICA A
Binding (Objeto)
CheckBox (Control Ligero)
CheckBox (Control)
ComboBox (Control Ligero)
ComboBox (Control)
DataBinding (Objeto)
Extender (Objeto)
Function (Control, diseñador de informe de datos)
Image (Control)
Label (Control)
ListBox (Control Ligero)
ListBox (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RepeaterBinding (Objeto)
RichTextBox (Control)
TextBox (Control Ligero)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
VBControlExtender (Objeto)

DataMember (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece un miembro de datos especificado entre los varios que ofrece el proveedor de datos.

Sintaxis
objeto.**DataMember** [= *cadena*]

Parte Descripción
objeto Requerido. Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
*cadena*El nombre de un miembro de datos.

Comentarios
Un proveedor de datos puede disponer de múltiples conjuntos de datos entre los que un receptor puede elegir enlazar. Cada conjunto de datos se llama "miembro de datos" y está identificado mediante una cadena única.

Por ejemplo, cuando utiliza un entorno de datos que contiene numerosos objetos **Command** como en el caso de un **DataSource**, la propiedad **DataMember** permite especificar qué objeto **Command** se debe usar.

Cuando utilice un módulo de clase o un control de usuario como origen de datos, use **GetDataMember** en su código para obtener un miembro de datos apropiado. El argumento *miembroDatos* del evento contiene el valor de la propiedad **DataMember**. Al solicitar el argumento, puede determinar qué miembro de datos ha sido solicitado y devolver los datos apropiados a través del argumento *Datos*.

SE APLICA A
BindingCollection (Objeto)
CheckBox (Control)
ComboBox (Control)
DataBinding (Objeto)
DataReport (Objeto)
Extender (Objeto)
Image (Control)
Label (Control)
ListBox (Control)
PictureBox (Control)
RichTextBox (Control)
TextBox (Control)

Ejemplo del objeto BindingCollection de la colección DataMembers

En el ejemplo se utiliza un módulo de clase como origen de datos. Cuando ejecuta el código que establece las propiedades **DataSource** y **DataMember** de dos objetos **Binding**, ocurre el evento Initialize del módulo de clase; durante dicho evento, dos recordsets ADO son creados y sus nombres son agregados a la colección **DataMembers**. El evento GetDataMember y sus argumentos son utilizados para devolver datos al receptor de datos.

Para probar el ejemplo, haga clic en **References** en el menú **Proyecto** y establezca una referencia a **Microsoft Data Binding Collection** y **Microsoft ActiveX Data Objects**. En el menú **Proyecto**, haga clic en **Agregar módulo de clase**. Cambie el nombre de la clase por MiClaseDeDatos y establezca la propiedad **DataSourceBehavior** a **vbDataSource**. Dibuje a continuación dos controles **TextBox** en un formulario. Pegue el código en el módulo de código del objeto **Form**.

```
Option Explicit
' Declara las variables de objeto, una para un módulo de
clase llamado
' MiClaseDeDatos, y dos más para los recordsets respectivos
de cada
' objeto BindingCollection.
```

```
Private clsData As New MiClaseDeDatos                    '
Módulo de clase
Private bndColProductos As New BindingCollection         '
Colección Bindings
Private bndColProveedores As New BindingCollection       '
Colección Bindings

Private Sub Form_Load()
   ' Establece las propiedades DataSource y DataMember de
cada objeto
   ' BindingCollection.
   With bndColProductos
      .DataMember = "Productos"
      Set .DataSource = clsData
      .Add Text1, "Text", "NombreProducto" ' Enlazado con
un TextBox.
   End With

   With bndColProveedores
      .DataMember = "Proveedores"
      Set .DataSource = clsData
      .Add Text2, "Text", "NombreOrganización" ' Enlazado
con un TextBox.
   End With

   ' Cambia la propiedad Caption de Command1
   Command1.Caption = "Siguiente"

End Sub

Private Sub Command1_Click()
   clsData.MoveNext
End Sub
```

Pegue el código siguiente en el módulo de clase de MiClaseDeDatos. La propiedad **DataSourceBehavior** debe tener el valor **vbDataSource** para poder ver el evento GetDataMember. Ejecute el proyecto.

```
Option Explicit
'  Declara variables de objeto para los objetos ADO
Connection y
' Recordset.
Private WithEvents rsProductos As ADODB.Recordset
Private WithEvents rsProveedores As ADODB.Recordset
Private cnNwind As ADODB.Connection
```

```
Private Sub Class_Initialize()
    ' Agrega cadenas a la colección DataMembers.
    With DataMembers
        .Add "Productos"
        .Add "Proveedores"
    End With

    ' Establece objetos Recordset.
    Set rsProductos = New ADODB.Recordset
    Set rsProveedores = New ADODB.Recordset
    Set cnNwind = New ADODB.Connection

    ' Establece los parámetros del objeto Connection.
    With cnNwind
        ' El archivo Nwind.mdb incluido con Visual Basic debe
instalarse en
        ' el equipo, en otro caso el código fallará. En
cualquier caso,
        ' modifique la ruta para buscar el archivo en el
equipo.
        .Provider = "Microsoft.Jet.OLEDB.3.51"
        .Open                    "C:\Archivos                    de
programa\DevStudio\VB\Nwind.mdb"
    End With

    ' Abre los objetos recordset.
    rsProveedores.Open "SELECT * FROM Proveedores", cnNwind,
_
    adOpenStatic, adLockOptimistic
    rsProductos.Open "SELECT * FROM Productos", cnNwind, _
    adOpenStatic, adLockOptimistic

End Sub

' GetDataMember ocurre cuando se establece la propiedad
DataSource
' de un receptor de datos. En este caso, el objeto
BindingCollection es
' el receptor.
Private Sub Class_GetDataMember(MiembroDatos As String,
Datos As Object)
    Select Case MiembroDatos
    Case "Productos"
        Set Datos = rsProductos
    Case "Proveedores"
        Set Datos = rsProvededores
```

```
   Case ""
      ' Proporciona un origen de registro predeterminado si
no se
      ' especifica un miembro de datos.
      Set Datos = rsProductos
   End Select

End Sub

Public Function MoveNext()
   If rsProductos.EOF Then
      rsProductos.MoveFirst
   Else
      rsProductos.MoveNext
   End If
End Function

Private Sub rsProductos_MoveComplete(ByVal adReason As _
ADODB.EventReasonEnum,   ByVal   pError   As   ADODB.Error,
adStatus As _
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
   ' Mantiene sincronizados los dos recordsets. El primer
textbox muestra
   ' el proveedor del producto. Si el valor IdProveedor
para ambos
   ' proveedores es el mismo, no se necesitan más cambios.
En otro caso,
   ' cambia al primer registro y comprueba IdProveedor.
Este ejemplo
   ' sólo tiene fines de demostración, pues este método no
es
   ' el más eficiente.

   If rsProveedores("IdProveedor").Value = _
   pRecordset("IdProveedor").Value Then Exit Sub

   rsProveedores.MoveFirst
   Do While Not rsProveedores.EOF
      If rsProveedores("IdProveedor").Value = _
      pRecordset("IdProveedor").Value Then
         Exit Sub
      Else
         rsProveedores.MoveNext
      End If
   Loop
End Sub
```

DataSource (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el origen de datos mediante el cual un receptor de datos enlaza con una base de datos.

Sintaxis

objeto.**DataSource** [=*datasource*]

La sintaxis de la propiedad **DataSource** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

datasource Una referencia a un objeto que califica como un origen de datos, incluso objetos **Recordset ADO** y clases o controles de usuario definidos como orígenes de datos (propiedad **DataSourceBehavior = vbDataSource**).

Comentarios

Utilice la instrucción **Set** para establecer la propiedad **DataSource**, como se indica a continuación:

```
Set Text1.DataSource = ADODC1
```

Nota Es posible usar como orígenes de datos dos controles antiguos, el control **Data** y el control **RemoteData**, sin embargo no puede establecer la propiedad **DataSource** de otro control u objeto a ninguno de estos controles en tiempo de ejecución. Por ejemplo, el código siguiente no funcionará:

```
Set Text1.DataSource = Data1
'¡Fallará! No puede establecer DataSource  a un control
Data intrínseco en tiempo  de ejecución.
```

Para usar el control **Data** o el control **RemoteData** como origen de datos, puede establecer la propiedad **DataSource** de los controles enlazados tan sólo en tiempo de diseño.

SE APLICA A

BindingCollection (Objeto)
CheckBox (Control)
ComboBox (Control)
DataReport (Objeto)
Extender (Objeto)
Image (Control)
Label (Control)
ListBox (Control)
Masked Edit (Control)
PictureBox (Control)
RichTextBox (Control)

DragIcon (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el icono que se presenta como puntero en una operación de arrastrar y colocar.

Sintaxis

objeto.**DragIcon** [= *icono*]

La sintaxis de la propiedad **DragIcon** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
icono	Cualquier referencia de código cuyo resultado sea un icono válido, como la referencia al icono de un formulario (`Form1.Icon`), una referencia a la propiedad **DragIcon** de otro control (`Text1.DragIcon`) o la función **LoadPicture**.

Valores

Los valores admitidos para *icono* son:

Valor	Descripción
(ninguna)	(Predeterminado) Un puntero de flecha dentro de un rectángulo.
Icono	Un puntero de *mouse* (ratón) personalizado. Para especificar el icono, defínalo mediante la ventana Propiedades en tiempo de diseño. También puede usar la función **LoadPicture** en tiempo de ejecución. El archivo que cargue debe tener la extensión .ico y formato de icono.

Comentarios

Puede usar la propiedad **DragIcon** para proporcionar información visual durante una operación de arrastrar y colocar; por ejemplo, para indicar que el control de origen está sobre el destino apropiado. **DragIcon** tiene efecto cuando el usuario inicia una operación de arrastrar y colocar. Normalmente, **DragIcon** se define como parte de un procedimiento de evento MouseDown o DragOver.

Nota En tiempo de ejecución, a la propiedad **DragIcon** se le pueden asignar los valores de las propiedades **DragIcon** o **Icon** de cualquier objeto, o se le puede asignar un icono devuelto por la función **LoadPicture**.

Cuando establezca la propiedad **DragIcon** en tiempo de ejecución asignando la propiedad **Picture** de un control a la propiedad **DragIcon** de otro control, la propiedad **Picture** debe contener un archivo .ico y no un archivo .bmp.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)

Ejemplo de la propiedad DragIcon

Este ejemplo cambia el valor de la propiedad **DragIcon** cada vez que se arrastra un control **PictureBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **PictureBox**. Establezca la propiedad **DragMode** a 1, y después presione F5 y haga clic y arrastre el control **PictureBox**.

```
Private Sub Form_DragDrop (Source As Control, X As Single,
Y As Single)
   Dim Pic    ' Declara una variable.
   Source.Move X, Y    ' Establece la posición del control.
   Pic = "ICONS\OFFICE\CRDFLE01.ICO"    ' Obtiene el nombre
del archivo de                                          '
iconos.
   If Source.DragIcon = False Then     ' Si no hay ninguna
imagen cargada,
      Source.DragIcon = LoadPicture(Pic)       ' carga la
imagen.
```

```
    Else
        Source.DragIcon = LoadPicture()        ' Descarga la
imagen.
    End If
End Sub
```

DragMode (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si se usa el modo de arrastre manual o automático en una operación de arrastrar y colocar.

Sintaxis

objeto.**DragMode** [= *número*]

La sintaxis de la propiedad **DragMode** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica el modo de arrastre, como se describe en Valores.

Valores

Los valores admitidos para *número* son:

Constante	Valor	Descripción
vbManual	0	(Predeterminado) Manual: requiere el uso del método **Drag** para iniciar una operación de arrastrar y colocar en el control de origen.
vbAutomatic	1	Automático: al hacer clic en el control de origen se inicia automáticamente una operación de arrastrar y colocar. Los controles contenedores **OLE** sólo se arrastran automáticamente cuando no tienen el enfoque.

Comentarios

Cuando **DragMode** se establece a 1 (Automático), el control no responde de la forma habitual a los eventos del *mouse*. Use el valor 0 (Manual) para determinar cuándo empieza o termina una operación de arrastrar y colocar; puede usar este valor para iniciar una operación de arrastrar y colocar como respuesta a un comando de teclado o de menú, o para permitir que el control de origen reconozca un evento MouseDown previo a la operación de arrastrar y colocar.

Hacer clic mientras el puntero del *mouse* se encuentra sobre un objeto o formulario de destino durante una operación de arrastrar y colocar genera un evento DragDrop en el objeto de destino. Esto termina la operación de arrastrar y colocar. Una operación de arrastrar y colocar también puede generar un evento DragOver.

Nota Mientras se arrastra un control, éste no puede reconocer otros eventos del *mouse* o del teclado iniciados por el usuario (KeyDown, KeyPress o KeyUp, MouseDown, MouseMove o MouseUp). Sin embargo, el control puede recibir eventos iniciados por código o por un vínculo DDE.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)

Ejemplo de la propiedad DragMode

Este ejemplo habilita y deshabilita la capacidad de arrastrar un control **CommandButton** cada vez que se hace clic en un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un **CommandButton**, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   ' Comprueba DragMode.
   If Command1.DragMode = vbManual Then
```

```
        ' Lo activa.
        Command1.DragMode = vbAutomatic
    Else
        ' O lo desactiva.
        Command1.DragMode = vbManual
    End If
End Sub
```

DrawMode (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina la apariencia del resultado de un método gráfico o la apariencia de un control **Shape** o **Line**.

Sintaxis

objeto.**DrawMode** [= *número*]

La sintaxis de la propiedad **DrawMode** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica la apariencia, como se describe en Valores.

Valores

Los valores admitidos para *número* son:

Constante	Valor	Descripción
vbBlackness	1	Negro.
vbNotMergePen	2	Inverso de combinar el lápiz y la muestra: inverso del valor 15.
vbMaskNotPen	3	Combinar fondo e inverso del lápiz: combinación de los colores comunes en el color de fondo y el inverso de la pluma.
vbNotCopyPen	4	Inverso de copiar del lápiz: inverso del valor 13.
vbMaskPenNot	5	Combinar el lápiz y el inverso de la muestra: combinación de los colores comunes a la pluma y al inverso de la presentación.
vbInvert	6	Invertir: inverso del color de presentación.
vbXorPen	7	Operación Xor con el lápiz: combinación de los colores de la pluma y del color de presentación, pero no de ambos.
vbNotMaskPen	8	Inverso de combinar el lápiz y los colores comunes: inverso del valor 9.
vbMaskPen	9	Combinación del lápiz y la presentación: combinación de los colores comunes a la pluma y a la presentación.
vbNotXorPen	10	Inverso de la operación Xor con el lápiz: inverso del valor 7.
vbNop	11	Ninguna operación. El resultado permanece sin cambios.

		De hecho, este valor desactiva el dibujo.
vbMergeNotPen	12	Combinación de la muestra y el inverso del lápiz: combinación del color de presentación y el inverso del color de la pluma.
vbCopyPen	13	Copiar del lápiz (predeterminado): color especificado por la propiedad **ForeColor**.
vbMergePenNot	14	Combinación del lápiz y el inverso de la muestra: combinación del color de la pluma y el inverso del color de presentación.
vbMergePen	15	Combinación del lápiz y la muestra: combinación del color de la pluma y el color de presentación.
vbWhiteness	16	Blanco.

Comentarios

Use esta propiedad para producir efectos visuales con controles **Shape** o **Line** o al dibujar con métodos gráficos. Visual Basic compara cada píxel del patrón de dibujo con el píxel correspondiente del fondo existente y después aplica operaciones en el nivel de bit. Por ejemplo, el valor 7 (Operación Xor con el lápiz) usa el operador **Xor** para combinar un píxel del patrón de dibujo con un píxel del fondo.

El efecto exacto de un valor de **DrawMode** depende del modo en que el color de una línea dibujada en tiempo de ejecución se combina con los colores de la pantalla. Los valores 1, 6, 7, 11, 13 y 16 producen los mejores resultados.

SE APLICA A

Form (Objeto), Forms (Colección)
Line (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
Shape (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad DrawMode

Este ejemplo permite dibujar en un formulario arrastrando el puntero del *mouse* (ratón). Cada clic con el *mouse* define un valor diferente de la propiedad **DrawMode**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Load
   DrawWidth = 10    ' Establece DrawWidth.
End Sub
Private Sub Form_Click ()
```

```
   Static M As Integer    ' Valor actual de DrawMode.
   ForeColor = QBColor(Int(Rnd * 15))    ' Elige un color.
   M = ((M + 1) Mod 16) + 1    ' Hace que DrawMode sea 16 o
menor.
   DrawMode = M    ' Establece DrawMode.
End Sub
Private Sub Form_MouseMove (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   If Button Then    ' Mientras se presiona el botón,
      PSet (X, Y)    ' se dibuja un punto grande.
   End If
End Sub
```

DrawStyle (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina el estilo de línea del resultado de métodos gráficos.

Sintaxis

objeto.**DrawStyle** [= *número*]

La sintaxis de la propiedad **DrawStyle** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica el estilo de línea, como se describe en Valores.

Valores

Los valores admitidos para *número* son:

Constante	Valor	Descripción
vbSolid	0	(Predeterminado) Sólido.
vbDash	1	Raya.
vbDot	2	Punto.
vbDashDot	3	Raya-punto.
vbDashDotDot	4	Raya-punto-punto.
vbInvisible	5	Transparente.
vbInsideSolid	6	Interior sólido.

Comentarios

Si **DrawWidth** se establece a un valor mayor que 1, los valores de **DrawStyle** entre 1 y 4 producen una línea sólida (el valor de la propiedad **DrawStyle** no cambia). Si **DrawWidth** se establece a 1, **DrawStyle** produce el efecto descrito para cada valor en la tabla anterior.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad DrawStyle

Este ejemplo dibuja siete líneas en un formulario, cada una de ellas con un valor distinto de la propiedad **DrawStyle**. (Si establece **AutoRedraw** a **True**, el formulario acumula un nuevo conjunto de líneas cada vez que le cambia de tamaño y hace clic en él). Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim I   ' Declara una variable.
   ScaleHeight = 8   ' Divide el alto por 8.
   For I = 0 To 6
      DrawStyle = I   ' Cambia el estilo.
      Line (0, I + 1) - (ScaleWidth, I + 1)    ' Dibuja una
línea nueva.
   Next I
End Sub
```

DrawWidth (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el ancho de línea para el resultado de los métodos gráficos.

Sintaxis

objeto.**DrawWidth** [= *tamaño*]

La sintaxis de la propiedad **DrawWidth** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
tamaño	Una expresión numérica comprendida entre 1 y 32.767 que representa el ancho de la línea en píxeles. El valor predeterminado es 1, es decir, un píxel de ancho.

Comentarios

Incremente el valor de esta propiedad para aumentar el ancho de la línea. Si el valor de la propiedad **DrawWidth** es mayor que 1, los valores de 1 a 4 en la propiedad **DrawStyle** producirán una línea continua (el valor de **DrawStyle** no se modifica). Si establece 1 en **DrawWidth**, **DrawStyle** producirá los resultados mostrados en la tabla de esta propiedad.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad DrawWidth

En este ejemplo se dibuja una línea que va aumentando de grosor gradualmente a través de un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim I    ' Declarar una variable.
   DrawWidth = 1    ' Establecer el ancho inicial del
                    ' pincel.
   PSet (0, ScaleHeight / 2)    ' Establecer el punto
                                ' inicial.
   ForeColor = QBColor(5)    ' Establecer el color
                             ' del pincel.
   For I = 1 To 100 Step 10    ' Iniciar un bucle.
      DrawWidth = I    ' Restablecer el ancho del pincel.
      Line - Step(ScaleWidth / 10, 0)
                             ' Dibujar una línea.
   Next I
End Sub
```

Drive (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece la unidad seleccionada en tiempo de ejecución. No está disponible en tiempo de diseño.

Sintaxis

objeto.**Drive** [= *unidad*]

La sintaxis de la propiedad **Drive** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

unidad Una expresión de cadena que especifica la unidad seleccionada.

Comentarios

Entre las unidades válidas para la propiedad **Drive** se incluyen todas las unidades presentes o conectadas al sistema cuando se crea el control o cuando se actualiza en tiempo de ejecución. El valor predeterminado de la propiedad **Drive** es la unidad actual.

Cuando se lee el valor de esta propiedad, la unidad seleccionada se devuelve en uno de los siguientes formatos:

- Discos flexibles: "a:" o "b:", etc.
- Medio fijo: "c: [*id volumen*]"
- Conexiones de red: "x: \\servidor\recursoCompartido"
- Cuando se establece esta propiedad:
- Sólo es significativo el primer carácter de la cadena (la cadena no distingue entre mayúsculas y minúsculas).
- Cambiar el valor de la propiedad **Drive** invoca un evento Change.
- Seleccionar una unidad no existente produce un error.
- Establecer esta propiedad también vuelve a generar la lista de unidades, lo que proporciona un método para hacer un seguimiento desde el código de las conexiones de red que se han agregado desde que se creó el control.

Si la propiedad **FileName** se establece a una ruta de red completa sin una designación de unidad, el valor de la propiedad **Drive** es una cadena de longitud cero ("), no se selecciona ninguna unidad y el valor de la propiedad **ListIndex** es –1.
Nota La propiedad **Drive** devuelve un valor diferente que la propiedad **ListIndex**, que devuelve la selección del cuadro de lista.

SE APLICA A CONSULTAR LAS PROPOEDADES METODOS Y EVENTOS DE DriveListBox (Control)

Ejemplo de la propiedad Drive

Este ejemplo presenta la lista de archivos del directorio y la unidad actuales. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **DriveListBox**, un control **DirListBox** y un control **FileListBox**, y después presione F5. Use el *mouse* para cambiar la unidad o el directorio.

```
Private Sub Drive1_Change ()
   Dir1.Path = Drive1.Drive    ' Cuando cambie la unidad,
                               '  establecer  la  ruta  del
directorio.
End Sub
Private Sub Dir1_Change ()
```

```
   File1.Path = Dir1.Path            ' Cuando cambie el
directorio,
                                     ' establecer la ruta del
archivo.
End Sub
```

Enabled (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si un formulario o un control puede responder a eventos generados por el usuario.

Sintaxis

objeto.**Enabled** [= *booleano*]

La sintaxis de la propiedad **Enabled** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supondrá que el formulario asociado al módulo del formulario activo es *objeto*.
booleano	Una expresión de tipo Boolean que especifica si *objeto* puede responder a eventos generados por el usuario.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado) Permite que *objeto* responda a eventos.
False	Impide que *objeto* responda a eventos.

Comentarios

La propiedad **Enabled** permite activar o desactivar formularios y controles en tiempo de ejecución. Por ejemplo, puede desactivar objetos que no se aplican al estado actual de la aplicación. También puede desactivar un control empleado fundamentalmente con fines de presentación, como un cuadro de texto que proporciona información de sólo lectura.

Si desactiva un control **Timer** estableciendo **Enabled** a **False** cancelará la cuenta atrás establecida por la propiedad **Interval** del control.

Para un control **Menu**, **Enabled** suele ser de lectura y escritura en tiempo de ejecución. Sin embargo, **Enabled** es de sólo lectura para elementos de menú que Visual Basic expone o suministra a complementos, como el comando **Administrador de complementos** del menú **Complementos**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
TextBox (Control)
Timer (Control)
UserControl (Objeto)

Ejemplo de la propiedad Enabled

Este ejemplo habilita un control **CommandButton** siempre que un control **TextBox** contenga texto. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario con controles **CommandButton** y **TextBox**, y después presione F5 y escriba algo en el cuadro de texto.

```
Private Sub Form_Load ()
   Text1.Text = ""    ' Borra el cuadro de texto.
   Command1.Caption = "Guardar"    ' Coloca el título del
botón.
End Sub

Private Sub Text1_Change ()
   If Text1.Text = "" Then    ' Mira si el cuadro de texto
está vacío.
      Command1.Enabled = False    ' Deshabilita el botón.
   Else
      Command1.Enabled = True    ' Habilita el botón.
   End If
End Sub
```

FillColor (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el color usado para llenar formas; **FillColor** también se usa para llenar círculos y cuadros creados con los métodos gráficos **Circle** y **Line**.

Sintaxis

objeto.**FillColor** [= *valor*]

La sintaxis de la propiedad **FillColor** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

valor Un valor o una constante que determina el color de relleno, como se describe en Valores.

Valores

Los valores admitidos para *valor* son:

Valor Descripción

Colores RGB normales Colores establecidos en el código mediante las funciones **RGB** o **QBColor**.

Colores de sistema Colores especificados con las constantes de color del sistema de la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos. El entorno operativo Microsoft Windows sustituye las opciones del usuario, como se especifica en los valores del Panel de control del usuario.

De forma predeterminada, **FillColor** está establecida a 0 (Negro).

Comentarios

Excepto en el objeto **Form**, cuando la propiedad **FillStyle** se establece al valor predeterminado, 1 (Transparente), el valor de **FillColor** se pasa por alto.

SE APLICA A

Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
Shape (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad FillColor

Este ejemplo crea un círculo en un formulario con valores aleatorios de las propiedades **FillColor** y **FillStyle** al hacer clic con el *mouse*. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, y después presione F5 y haga clic en el formulario.

```
Private Sub Form_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   FillColor = QBColor(Int(Rnd * 15))    ' Elige un valor
aleatorio de                                            '
FillColor.
   FillStyle = Int(Rnd * 8)    ' Elige un valor aleatorio de
FillStyle.
   Circle (X, Y), 250    ' Dibuja un círculo.
End Sub
```

FileName (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece la ruta y el nombre de un archivo seleccionado. No está disponible en tiempo de diseño para el control **FileListBox**.

Sintaxis

objeto.**FileName** [= *nombreRuta*]

La sintaxis de la propiedad **FileName** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

nombreRuta Una expresión de cadena que especifica la ruta y el nombre de archivo.

Comentarios

Cuando crea el control en tiempo de ejecución, la propiedad **FileName** se establece a una cadena de longitud cero ("""), que significa que no hay ningún archivo seleccionado.

En el control **CommonDialog**, puede establecer la propiedad **FileName** antes de abrir un cuadro de diálogo para establecer un nombre de archivo inicial.

La lectura de esta propiedad devuelve el nombre del archivo seleccionado actualmente en la lista. La ruta se devuelve por separado, mediante la propiedad **Path**. El valor es funcionalmente equivalente a **List**(*ÍndiceLista*). Si no hay ningún archivo seleccionado, **FileName** devuelve una cadena de longitud cero.

Al establecer esta propiedad:

- La inclusión de una unidad, ruta o patrón en la cadena provoca el cambio de los valores de las propiedades **Drive**, **Path** y **Pattern**.

- La inclusión del nombre de un archivo existente (sin caracteres comodín) en la cadena selecciona el archivo.

- El cambio del valor de esta propiedad también puede provocar uno o más de estos eventos: PathChange (si cambia la ruta), PatternChange (si cambia el patrón) o DblClick (si asigna un nombre de archivo existente).

- El valor de esta propiedad puede ser un nombre completo de red y de archivo con la siguiente sintaxis
  `\\nombreServidor\recursoCompartido\nombreRuta`

SE APLICA A LAS PROPIEDADES, METODOS Y EVENTOS DE FileListBox

FileListBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos

El control **FileListBox** encuentra y muestra los archivos del directorio especificado por la propiedad **Path** en tiempo de ejecución. Utilice este control para mostrar una lista de los archivos seleccionados por tipo. Puede crear cuadros de diálogo en la aplicación que, por ejemplo, permitan al usuario seleccionar un archivo o un grupo de archivos.

Sintaxis
FileListBox

Comentarios
Establezca las propiedades **List**, **ListCount** y **ListIndex** para permitir al usuario tener acceso a los elementos de la lista. Si además muestra los controles **DirListBox** y **DriveListBox**, podrá escribir código para sincronizarlos con el control **FileListBox** y entre sí.

PROPIEDADES DE FileListBox (Control)

Appearance (Propiedad)
Archive, Hidden, Normal, System (Propiedades)
BackColor, ForeColor (Propiedades)
Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
FileName (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
List (Propiedad)
ListCount (Propiedad)
ListIndex (Propiedad)
Locked (Propiedad)
MouseIcon (Propiedad)

MousePointer (Propiedad)
MultiSelect (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
Parent (Propiedad)
Path (Propiedad)
Pattern (Propiedad)
ReadOnly (Propiedad)
Selected (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
ToolTipText (Propiedad)
TopIndex (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE FileListBox (Control)

Drag (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE FileListBox (Control)

Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEGiveFeedback (Evento)
OLESetData (Evento)

OLEStartDrag (Evento)
PathChange (Evento)
PatternChange (Evento)
Scroll (Evento)
Validate (Evento)

FillStyle (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el patrón usado para llenar controles **Shape**, así como los círculos y los cuadros creados con los métodos gráficos **Circle** y **Line**.

Sintaxis

objeto.**FillStyle** [= *número*]

La sintaxis de la propiedad **FillStyle** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica el estilo de relleno, como se describe en Valores.

Valores

Los valores para *número* son:

Constante	Valor	Descripción
vbFSSolid	0	Sólido.
vbFSTransparent	1	(Predeterminado) Transparente.
vbHorizontalLine	2	Línea horizontal.
vbVerticalLine	3	Línea vertical.
vbUpwardDiagonal	4	Diagonal hacia arriba.
vbDownwardDiagonal	5	Diagonal hacia abajo.
vbCross	6	Cruz.
vbDiagonalCross	7	Cruz diagonal.

Comentarios

Cuando la propiedad **FillStyle** se establece a su valor predeterminado, 1 (Transparente), el valor de **FillColor** se pasa por alto, excepto en el objeto **Form**.

SE APLICA A

Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
Shape (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad FillStyle

Este ejemplo crea un círculo en un formulario con valores aleatorios de las propiedades **FillColor** y **FillStyle** al hacer clic con el *mouse*. Para probar este ejemplo, pegue el código en la sección Declaraciones, y después presione F5 para ejecutar el programa.

```
Private Sub Form_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   FillColor = QBColor(Rnd * 15)      ' Elige un valor
aleatorio para                                        '
FillColor.
   FillStyle = Int(Rnd * 8)    ' Elige un valor aleatorio
para FillStyle.
   Circle (X, Y), 250   ' Dibuja un círculo.
End Sub
```

Font (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve un objeto **Font**.

Sintaxis

objeto.**Font**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Utilice la propiedad **Font** de un objeto para identificar un objeto **Font** específico cuyas propiedades desee usar. Por ejemplo, el código siguiente cambia el valor de la propiedad **Bold** de un objeto **Font** identificado por la propiedad **Font** de un objeto **TextBox**:

```
txtFirstName.Font.Bold = True
```

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Control de datos ADO
Data (Control)
DataReport (Objeto)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Function (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)

ListBox (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
UserControl (Objeto)
UserDocument (Objeto)

FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)

Vea también Ejemplo Se aplica a

Devuelve o establece los estilos de fuente en los siguientes formatos: **Negrita**, *Cursiva*, Tachado y Subrayado.

Nota Las propiedades **FontBold**, **FontItalic**, **FontStrikethru** y **FontUnderline** se incluyen para su uso en el control **CommonDialog** y por compatibilidad con versiones anteriores de Visual Basic. Para obtener funcionalidad adicional, utilice las propiedades del nuevo objeto **Font** (no disponibles para el control **CommonDialog**).

Sintaxis

objeto.**FontBold** [= *booleano*]
objeto.**FontItalic** [= *booleano*]
objeto.**FontStrikethru** [= *booleano*]
objeto.**FontUnderline** [=*booleano*]

La sintaxis de las propiedades **FontBold**, **FontItalic**, **FontStrikethru** y **FontUnderline** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica el estilo de fuente, según se indica en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado para **FontBold**, excepto con el control **CommonDialog**). Habilita el formato en el estilo correspondiente.
False	(Predeterminado para **FontItalic**, **FontStrikethru** y **FontUnderline**, y para **FontBold** con el control **CommonDialog**). Deshabilita el formato en el estilo correspondiente.

Comentarios
Utilice estas propiedades de fuente para dar formato al texto, ya sea en tiempo de diseño mediante la ventana Propiedades o en tiempo de ejecución a través de código. En el caso de los controles **PictureBox** y los objetos **Form** y **Printer**, establecer estas propiedades no afecta a los gráficos ni al texto ya dibujados en el control u objeto. Para todos los demás controles, las modificaciones en la fuente se hacen efectivas en la pantalla inmediatamente.
Para usar estas propiedades con el control **CommonDialog** debe estar activado el indicador **Effects**.

Nota Las fuentes disponibles en Visual Basic varían dependiendo de la configuración del sistema y de los dispositivos de presentación e impresión. En las propiedades relacionadas con las fuentes sólo pueden establecerse valores para los que existan fuentes en el sistema.
En general, deberá modificar la propiedad **FontName** antes de establecer los atributos de tamaño y estilo con las propiedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** y **FontUnderline**. Sin embargo, cuando especifique un tamaño inferior a 8 puntos para una fuente TrueType, debe establecer el tamaño en puntos con la propiedad **FontSize**, después especificar la propiedad **FontName** y después establecer de nuevo el tamaño con **FontSize**. El entorno operativo Microsoft Windows utiliza una fuente distinta para las fuentes TrueType con un tamaño inferior a 8 puntos.

SE APLICA A
AmbientProperties (Objeto)
AxisTitle (Objeto)
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Label (Control)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de las propiedades FontBold, FontItalic, FontStrikethru y FontUnderline

En este ejemplo se escribe texto en un formulario con una de entre dos combinaciones de estilos con cada clic del *mouse* (ratón). Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario, presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   FontStrikethru = Not FontStrikethru     ' Alternar el
tachado.
   FontItalic = Not FontItalic   ' Alternar el estilo de la
fuente.
   Print "¡Ahora es el momento!"   ' Imprimir texto.
End Sub
```

FontName (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece la fuente utilizada para mostrar texto en un control o en una operación de dibujo o impresión en tiempo de ejecución.

Nota La propiedad **FontName** se incluye para su uso con el control **CommonDialog** y por compatibilidad con versiones anteriores de Visual Basic. Para obtener una funcionalidad adicional, utilice las propiedades del nuevo objeto **Font** (no disponibles para el control **CommonDialog**).

Sintaxis

objeto.**FontName** [=*fuente*]

La sintaxis de la propiedad **FontName** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
fuente	Una expresión de cadena que especifica el nombre de la fuente que se va a usar.

Comentarios

El valor predeterminado de esta propiedad lo determina el sistema. Las fuentes disponibles en Visual Basic varían dependiendo de la configuración del sistema y de los dispositivos de presentación e impresión. En las propiedades relacionadas con fuentes sólo pueden establecerse valores para los que exista una fuente.

En general, deberá modificar **FontName** antes de establecer los atributos de tamaño y estilo con las propiedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** y **FontUnderline**.

Nota En tiempo de ejecución puede obtener información acerca de las fuentes disponibles en el sistema mediante las propiedades **FontCount** y **Fonts**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Label (Control)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad FontName

En este ejemplo se escribe el nombre de cada fuente usando como fuente ella misma. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario. Presione F5 para ejecutar el programa y luego haga clic en el formulario. Cada vez que haga clic en el formulario aparecerá el nombre de la fuente.

```
Private Sub Form_Click ()
   Static I    ' Declarar variables.
   Dim OldFont
   OldFont = FontName    ' Conservar la fuente original.
   FontName = Screen.Fonts(I)      ' Cambiar a una nueva
fuente.
   Print Screen.Fonts(I)      ' Imprimir el nombre de la
fuente.
   I = I + 1    ' Incrementar el contador.
   If I = FontCount Then I = 0    ' Comenzar.
   FontName = OldFont    ' Restaurar la fuente original.
End Sub
```

FontSize (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el tamaño de la fuente que se va a usar para el texto mostrado en un control o en una operación de dibujo o impresión en tiempo de ejecución.

Nota La propiedad **FontSize** se incluye para su uso con el control **CommonDialog** y por compatibilidad con versiones anteriores de Visual Basic. Para obtener una funcionalidad adicional, utilice las propiedades del nuevo objeto **Font** (no disponibles para el control **CommonDialog**).

Sintaxis

objeto.**FontSize** [= *puntos*]

La sintaxis de la propiedad **FontSize** consta de las siguientes partes:

Parte Descripción

objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

puntos Una expresión numérica que especifica el tamaño de fuente que se va a usar, en puntos.

Comentarios

Utilice esta propiedad para dar al texto el tamaño que desee. El valor predeterminado está determinado por el sistema. Para cambiar este valor, especifique el nuevo tamaño de la fuente en puntos.

El valor máximo de **FontSize** es 2160 puntos.

Nota Las fuentes disponibles en Visual Basic varían dependiendo de la configuración del sistema y de los dispositivos de presentación e impresión. En las propiedades relacionadas con las fuentes sólo pueden establecerse valores para los que exista una fuente.

En general, deberá modificar **FontName** antes de establecer los atributos de tamaño y estilo con las propiedades **FontSize**, **FontBold**, **FontItalic**, **FontStrikethru** y **FontUnderline**. Sin embargo, cuando especifique un tamaño inferior a 8 puntos para una fuente TrueType, primero deberá establecer el tamaño en puntos con la propiedad **FontSize**, después especificar la propiedad **FontName** y por último establecer de nuevo el tamaño con **FontSize**. El entorno operativo Microsoft Windows utiliza una fuente distinta para las fuentes TrueType con un tamaño inferior a 8 puntos.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)

Label (Control)
ListBox (Control)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad FontSize

En este ejemplo se escribe texto en un formulario con uno de entre dos tamaños en puntos con cada clic del *mouse*. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario. Presione F5 para ejecutar el programa y luego haga clic en el formulario.

```
Private Sub Form_Click ()
   FontSize = 24   ' Establecer FontSize.
   Print "Éste es un tipo de 24 puntos."
         ' Imprimir con el tipo grande.
   FontSize = 8   ' Establecer FontSize.
   Print "Éste es un tipo de 8 puntos."
         'Imprimir con el tipo pequeño.
End Sub
```

FontTransparent (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si el texto y los gráficos de fondo de un objeto **Form** o **Printer**, o de un control **PictureBox**, se muestran en el espacio situado entre los caracteres.

Sintaxis

objeto.**FontTransparent** [= *booleano*]

La sintaxis de la propiedad **FontTransparent** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión de tipo Boolean que especifica el estado del texto y los gráficos de fondo, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado) Permite que los gráficos y el texto de fondo aparezcan entre los caracteres de una fuente.
False	Oculta los gráficos y el texto existentes entre los caracteres de una fuente.

Comentarios
Establezca **FontTransparent** en tiempo de diseño mediante la ventana Propiedades o en tiempo de ejecución a través del código. Cambiar **FontTransparent** en tiempo de ejecución no afecta a los gráficos ni el texto ya dibujados en un objeto **Form**, **Printer** o **PictureBox**.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad FontTransparent

En este ejemplo se imprime texto sobre un gráfico de un control **PictureBox**. Coloque un control **PictureBox** en un formulario, establezca a **True** su propiedad **AutoSize** y cargue en su propiedad **Picture** un archivo de mapa de bits (.bmp). Para probar este ejemplo, pegue el código en la sección Declaraciones del formulario, presione F5 y haga clic dos veces en el formulario.

```
Private Sub Form_Click ()
   ' Alterna la propiedad.
   Picture1.FontTransparent = Not Picture1.FontTransparent
   Picture1.Print "Demostración de la propiedad
FontTransparent."
End Sub
```

Height, Width (Propiedades)

Vea también Ejemplo Se aplica a
Devuelven o establecen las dimensiones de un objeto o el ancho del objeto **Columns** de un control **DataGrid**. Para los objetos **Printer** y **Screen**, no están disponibles en tiempo de diseño.

Sintaxis
objeto.**Height** [= *número*]
objeto.**Width** [= *número*]
La sintaxis de las propiedades **Height** y **Width** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Una expresión numérica que especifica las dimensiones del objeto, según se indica en Valores.

Valores

Las medidas se calculan de la siguiente forma:

- **Form**: alto y ancho externos del formulario, incluyendo los bordes y la barra de título.

- **Control**: se mide desde el centro del borde del control, de forma que los controles con anchos de borde distintos se alineen correctamente. Estas propiedades utilizan las unidades de escala del contenedor de un control.

- Objeto **Printer**: las dimensiones físicas del papel configurado para el dispositivo de impresión. No está disponible en tiempo de diseño. Si se establecen en tiempo de ejecución, los valores de estas propiedades se usarán en lugar de la propiedad **PaperSize**.

- Objeto **Screen**: el alto y el ancho de la pantalla. No están disponibles en tiempo de diseño y son de sólo lectura en tiempo de ejecución.

- Objeto **Picture**: el alto y el ancho de la imagen en unidades HiMetric.

Comentarios

Para los objetos **Form**, **Printer** y **Screen**, estas propiedades se miden siempre en twips. Para los formularios y controles, los valores de estas propiedades cambian cuando el usuario o el código modifican el tamaño del objeto. Los límites máximos de estas propiedades para todos los objetos dependen del sistema.

Si establece las propiedades **Height** y **Width** para un controlador de impresora que no admite su uso, no se producirá ningún error y el tamaño del papel continuará siendo el mismo. Si establece **Height** y **Width** para un controlador de impresora que sólo admite en ellas determinados valores, no se producirá ningún error y en la propiedad se establecerá un valor que el controlador admita. Por ejemplo, puede ocurrir que al intentar establecer 150 en **Height**, el controlador establezca 144.

Utilice las propiedades **Height**, **Width**, **Left** y **Top** para realizar operaciones o cálculos que impliquen el área total de un objeto, como puede ser mover el objeto o cambiar su tamaño. Utilice las propiedades **ScaleLeft**, **ScaleTop**, **ScaleHeight** y **ScaleWidth** para realizar operaciones o cálculos que impliquen solamente el área interna de un objeto, como dibujar o mover objetos dentro de otro objeto.

Nota La propiedad **Height** no puede modificarse en los controles **DriveListBox** ni **ComboBox**, en los cuales el valor de la propiedad **Style** es 0 (cuadro combinado desplegable) o 2 (lista desplegable).

Para el objeto **Columns** del control **DataGrid**, **Width** se especifica en la unidad de medida del objeto que contiene el control **DataGrid**. El valor predeterminado de **Width** es el valor de la propiedad **DefColWidth** del control **DataGrid**.

Para el objeto **Picture**, utilice los métodos **ScaleX** y **ScaleY** para convertir las unidades HiMetric a la escala que necesite.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Function (Control, diseñador de informe de datos)
HScrollBar, VScrollBar (Controles)
Image (Control)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
Line (Control, diseñador de informe de datos)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
Picture (Objeto)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
RemoteData (Control)
Screen (Objeto)
Section (Objeto, diseñador de informe de datos)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de las propiedades Height y Width

En este ejemplo se establece como tamaño de un formulario el 75 por ciento del tamaño de la pantalla y se centra el formulario al cargarlo. Para probar este ejemplo, pegue el

código en la sección Declaraciones de un formulario. Presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Width = Screen.Width * .75    ' Establecer el ancho del
formulario.
   Height = Screen.Height * .75    ' Establecer el alto del
formulario.
   Left = (Screen.Width - Width) / 2      ' Centrar el
formulario
' horizontalmente.
   Top = (Screen.Height - Height) / 2      ' Centrar el
formulario
' verticalmente.
End Sub
```

HelpContextID (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un número de contexto asociado a un objeto. Se utiliza para proporcionar Ayuda interactiva para la aplicación.

Sintaxis

objeto.**HelpContextID** [= *número*]

La sintaxis de la propiedad **HelpContextID** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supondrá que el formulario asociado al módulo de formulario activo es *objeto*.
número	Una expresión numérica que especifica el número de contexto del tema de Ayuda asociado a *objeto*.

Valores

Los valores admitidos para *número* son:

Valor	Descripción
0	(Predeterminado) No se especifica ningún número de contexto.
> 0	Un entero que especifica un número de contexto válido.

Comentarios

Para la Ayuda interactiva de un objeto de la aplicación debe asignar el mismo número de contexto a *objeto* y al tema de Ayuda asociado cuando compile el archivo de Ayuda.

Si ha creado un archivo de Ayuda del entorno operativo Microsoft Windows para su aplicación y establece la propiedad **HelpFile** de la aplicación, cuando el usuario presione la tecla F1, Visual Basic llamará automáticamente a la Ayuda y buscará el tema identificado por el número de contexto actual.

El número de contexto actual es el valor de **HelpContextID** para el objeto que tiene el enfoque. Si **HelpContextID** se establece a 0, Visual Basic buscará en la propiedad **HelpContextID** del contenedor del objeto y, a continuación, en el contenedor de este objeto, y así sucesivamente. Si no se encuentra un número de contexto actual distinto de cero, se pasa por alto la tecla F1.

Para un control **Menu**, **HelpContextID** suele ser de lectura y escritura en tiempo de ejecución. Sin embargo, **HelpContextID** es de sólo lectura para elementos de menú que Visual Basic expone o suministra a complementos, como el comando **Administrador de complementos** del menú **Complementos**.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Error (Objeto, diseñador de informe de datos)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MDIForm (Objeto)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)

Ejemplo de la propiedad HelpContextID

Este ejemplo utiliza temas del archivo de Ayuda de Visual Basic para demostrar cómo se especifican los números de contexto para los temas de Ayuda. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** y un control **Frame** con un control **OptionButton** dentro de él. Presione F5. Una vez que el programa se esté ejecutando, mueva el enfoque a uno de los controles y presione F1.

```
' Números de contexto reales para el archivo de Ayuda de
Visual Basic.
Const winColorPalette = 21004   ' Define constantes.
Const winToolbox = 21001
Const winCodeWindow = 21005

Private Sub Form_Load ()
   App.HelpFile = "VB.HLP"
   Frame1.HelpContextID = winColorPalette
   Text1.HelpContextID = winToolbox
   Form1.HelpContextID = winCodeWindow
End Sub
```

Image (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve un controlador de un gráfico persistente; el controlador lo proporciona el entorno operativo Microsoft Windows.

Sintaxis

objeto.**Image**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

La propiedad **AutoRedraw** de un objeto determina si un objeto se vuelve a dibujar mediante gráficos persistentes o mediante eventos Paint. El entorno operativo Windows identifica el gráfico persistente de un objeto asignándole un controlador; puede usar la propiedad **Image** para obtener este controlador.

El valor de **Image** existe independientemente del valor de la propiedad **AutoRedraw**. Si **AutoRedraw** es **True** y no se ha dibujado nada, la imagen sólo presenta el color definido en la propiedad **BackColor** y el gráfico.

Puede asignar un valor de **Image** a la propiedad **Picture**. La propiedad **Image** también proporciona un valor que se pasa en las llamadas a la API de Windows.

Las propiedades **Image**, **DragIcon** y **Picture** se usan normalmente cuando se asignan valores a otras propiedades, cuando se guarda con la instrucción **SavePicture** o cuando se coloca algo en el Portapapeles. No puede asignar estos valores a una variable temporal, salvo el tipo de datos **Picture**.

La propiedad **AutoRedraw** puede hacer que cambie **Image**, que es un controlador de un mapa de bits. Cuando **AutoRedraw** es **True**, la propiedad **hDC** de un objeto se convierte en un controlador de contexto de dispositivo que contiene el mapa de bits devuelto por **Image**.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad Image

Este ejemplo dibuja un círculo en el primer control **PictureBox** cada vez que hace clic en él. Cuando haga clic en el segundo **PictureBox**, el gráfico del primer **PictureBox** se copia en él. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga dos controles **PictureBox** grandes y de igual tamaño. Presione F5 para ejecutar el programa y después haga clic en los controles **PictureBox**.

```
Private Sub Form_Load ()
   ' Establece AutoRedraw a True.
   Picture1.AutoReDraw = True
End Sub

Private Sub Picture1_Click ()
   ' Declara variables.
   Dim PW, PH
   ' Establece FillStyle a sólido.
   Picture1.FillStyle = vbFSSolid
   ' Elige un color aleatorio.
   Picture1.FillColor = QBColor(Int(Rnd * 15))
   PW = Picture1.ScaleWidth    ' Establece ScaleWidth.
   PH = Picture1.ScaleHeight    ' Establece ScaleHeight.
   ' Dibuja un círculo en una ubicación aleatoria.
   Picture1.Circle (Int(Rnd * PW), Int(Rnd * PH)), 250
End Sub

Private Sub Picture2_Click ()
   ' Copia Image a Picture2.
   Picture2.Picture = Picture1.Image
End Sub
```

Index (Propiedad, matriz de controles)

Vea también Ejemplo Se aplica a

Devuelve o establece el número que identifica un control de forma exclusiva en una matriz de controles. Sólo está disponible si el control forma parte de una matriz de controles.

Sintaxis

objeto[(*número*)].**Index**

La sintaxis de la propiedad **Index** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Una expresión numérica que da como resultado un entero que identifica un control individual en una matriz de controles.

Valores

Los valores admitidos para *número* son:

Valor	Descripción
Ningún valor	(Predeterminado) No forma parte de una matriz de controles.
0 a 32.767	Parte de una matriz. Especifica un entero mayor o igual que 0 que identifica un control en una matriz de controles. Todos los controles de una matriz de controles tienen la misma propiedad **Name**. Visual Basic asigna automáticamente el siguiente entero disponible de la matriz de controles.

Comentarios

Puesto que los elementos de matrices de controles comparten el mismo valor de la propiedad **Name**, deberá usar la propiedad **Index** en código para especificar un determinado control de la matriz. **Index** debe aparecer como un entero (o una expresión numérica que da como resultado un entero) entre paréntesis al lado del nombre de la matriz de controles; por ejemplo, `MisBotones(3)`. También puede usar el valor de la propiedad **Tag** para distinguir un control de otro en una matriz de controles.

Cuando un control de la matriz reconoce que se ha producido un evento, Visual Basic llama al procedimiento de evento de la matriz de controles y pasa el valor de **Index** correspondiente como argumento adicional. Esta propiedad también sirve para crear controles dinámicamente en tiempo de ejecución con la instrucción **Load** o quitarlos con la instrucción **Unload**.

Aunque Visual Basic asigna de forma predeterminada el siguiente entero disponible como valor de **Index** para un nuevo control de una matriz de controles, puede anular este valor asignado y omitir enteros. También puede establecer **Index** a un entero distinto de 0 para el primer control de la matriz. Si hace referencia a un valor **Index** en el código que

no identifica uno de los controles de una matriz de controles, se producirá un error en tiempo de ejecución de Visual Basic.

Nota Para quitar un control de una matriz de controles, cambie el valor de la propiedad **Name** del control y elimine el valor de su propiedad **Index**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
Shape (Control)
TextBox (Control)
Timer (Control)

Ejemplo de la propiedad Index

Este ejemplo comienza con dos controles **OptionButton** y agrega un nuevo **OptionButton** al formulario cada vez que hace clic en un control **CommandButton**. Cuando hace clic en un **OptionButton**, se establece la propiedad **FillStyle** y se dibuja un nuevo círculo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que tenga dos controles **OptionButton**, un **CommandButton** y un control **PictureBox** grande. Establezca la propiedad **Name** a **optButton** para ambos botones para crear una matriz de controles.

```
Private Sub OptButton_Click (Index As Integer)
   Dim H, W    ' Declara variables.
   Picture1.Cls    ' Borra la imagen.
   Picture1.FillStyle = Index    ' Establece FillStyle.
   W = Picture1.ScaleWidth / 2    ' Obtiene el tamaño del
círculo.
   H = Picture1.ScaleHeight / 2
   Picture1.Circle (W, H), W / 2    ' Dibuja el círculo.
End Sub

Private Sub Command1_Click ()
   Static MaxIdx    ' Índice mayor de la matriz.
   If MaxIdx = 0 Then MaxIdx = 1    ' MaxIdx preestablecido.
   MaxIdx = MaxIdx + 1    ' Incrementa el índice.
   If MaxIdx > 7 Then Exit Sub    ' Coloca ocho botones en
el formulario.
   Load OptButton(MaxIdx)    ' Crea nuevos elementos en la
matriz.
   ' Establece la ubicación del nuevo botón de opciones
bajo un botón
   ' previo.
   OptButton(MaxIdx).Top = OptButton(MaxIdx - 1).Top + 360
   OptButton(MaxIdx).Visible = True    ' Hace visible el
nuevo botón.
End Sub
```

LargeChange, SmallChange (Propiedades)

Vea también Ejemplo Se aplica a

- **LargeChange**: devuelve o establece el cambio que se producirá en el valor de la propiedad **Value** de un control de barra de desplazamiento (**HScrollBar** o **VScrollBar**) cuando el usuario haga clic en el área situada entre el cuadro de desplazamiento y la flecha de desplazamiento.

- **SmallChange**: devuelve o establece el cambio que se producirá en el valor de la propiedad **Value** de un control de barra de desplazamiento cuando el usuario haga clic en la flecha de desplazamiento.

Sintaxis

objeto.**LargeChange** [= *número*]

objeto.**SmallChange** [= *número*]

Las sintaxis de las propiedades **LargeChange** y **SmallChange** constan de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

número Un entero que especifica el cambio que se producirá en el valor de la propiedad **Value**.

Comentarios
Para ambas propiedades puede especificar un entero entre 1 y 32.767, ambos inclusive. De forma predeterminada, cada propiedad está establecida a 1.

El entorno operativo Microsoft Windows define automáticamente incrementos proporcionales de desplazamiento para las barras de desplazamiento de los objetos **MDI Form**, controles **ComboBox** y **ListBox** basándose en la cantidad de datos del objeto. Sin embargo, en los controles **HScrollBar** y **VScrollBar**, debe especificar usted estos incrementos. Use **LargeChange** y **SmallChange** para establecer los incrementos de desplazamiento apropiados para el uso concreto de la barra de desplazamiento.

Normalmente, **LargeChange** y **SmallChange** se establecen en tiempo de diseño. También puede restablecerlas en el código en tiempo de ejecución cuando el desplazamiento deba modificarse dinámicamente.

Nota Los intervalos máximo y mínimo de los controles **HScrollBar** y **VScrollBar** se definen mediante las propiedades **Max** y **Min**.

SE APLICA A LAS PROPIEDADES, METODOS Y EVENTOS DE HscrollBar, VScrollBar

HScrollBar, VScrollBar (Controles)

Vea también Ejemplo Propiedades Métodos Eventos
Las barras de desplazamiento permiten explorar fácilmente una larga lista de elementos o una gran cantidad de información. Además, proporcionan una representación análoga de la posición actual. Puede usar una barra de desplazamiento como dispositivo de entrada o como indicador de velocidad o cantidad; por ejemplo, para controlar el volumen de un juego de PC o para ver el tiempo transcurrido en un proceso temporizado.
Sintaxis
HScrollBar
VScrollBar

Comentarios
Cuando utiliza una barra de desplazamiento como indicador de cantidad o velocidad, o como dispositivo de entrada, utilice las propiedades **Max** y **Min** para establecer el intervalo adecuado del control.

Para especificar la cantidad de cambio que va a indicar en una barra de desplazamiento, utilice la propiedad **LargeChange** para hacer clic en la barra de desplazamiento y la propiedad **SmallChange** para hacer clic en las flechas de los extremos de la barra. La propiedad **Value** de la barra de desplazamiento aumenta o disminuye los valores establecidos para las propiedades **LargeChange** y **SmallChange**. Puede colocar el

cuadro de desplazamiento en tiempo de ejecución si establece **Value** entre 0 y 32.767, ambos inclusive.

PROPIEDADES DE **HScrollBar, VScrollBar (Controles)**

Container (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
Index (Propiedad, matriz de controles)
LargeChange, SmallChange (Propiedades)
Left, Top (Propiedades)
Max, Min (Propiedades, barra de desplazamiento)
MouseIcon (Propiedad)
MousePointer (Propiedad)
Name (Propiedad)
Parent (Propiedad)
RightToLeft (Propiedad)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)
Value (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE **HScrollBar, VScrollBar (Controles)**

Drag (Método)
Move (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE **HScrollBar, VScrollBar (Controles)**

Change (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)

KeyPress (Evento)
LostFocus (Evento)
Scroll (Evento)
Validate (Evento)

Left, Top (Propiedades)

Vea también Ejemplo Se aplica a

- **Left**: devuelve o establece la distancia entre el borde interno izquierdo de un objeto y el borde izquierdo de su contenedor.

- **Top**: devuelve o establece la distancia entre el borde interno superior de un objeto y el borde superior de su contenedor.

Sintaxis
objeto.**Left** [= *valor*]
objeto.**Top** [= *valor*]
La sintaxis de las propiedades **Left** y **Top** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Una expresión numérica que especifica la distancia.

Comentarios
Para un formulario, las propiedades **Left** y **Top** se expresan siempre en twips, mientras que para los controles las unidades dependen del sistema de coordenadas del contenedor. Los valores de estas propiedades cambian cuando el usuario o el código desplazan el objeto. En el caso de los controles **CommonDialog** y **Timer**, estas propiedades no están disponibles en tiempo de ejecución.

En cada propiedad puede especificar un número de precisión simple.
Utilice las propiedades **Left**, **Top**, **Height** y **Width** para operaciones que se basen en las dimensiones externas de un objeto, como moverlo o cambiar su tamaño. Utilice las propiedades **ScaleLeft**, **ScaleTop**, **ScaleHeight** y **ScaleWidth** para operaciones basadas en las dimensiones internas del objeto, como dibujar o mover objetos contenidos dentro del objeto. Las propiedades relacionadas con la escala sólo se aplican a los controles **PictureBox** y a los objetos **Form** y **Printer**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)

Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Function (Control, diseñador de informe de datos)
HScrollBar, VScrollBar (Controles)
Image (Control)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
Line (Control, diseñador de informe de datos)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
Timer (Control)

Ejemplo de las propiedades Left y Top

En este ejemplo se establece como tamaño de un formulario el 75 por ciento del tamaño de la pantalla y se centra el formulario al cargarlo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario. Presione F5 y haga clic en el formulario.

```
Private Sub Form_Click ()
   Width = Screen.Width * .75    ' Establecer el ancho del
formulario.
   Height = Screen.Height * .75    ' Establecer el alto del
formulario.
   Left = (Screen.Width - Width) / 2      ' Centrar el
formulario
' horizontalmente.
   Top = (Screen.Height - Height) / 2      ' Centrar el
formulario
' verticalmente.
End Sub
```

List (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece los elementos contenidos en la parte de lista de un control. La lista es una matriz de cadenas cuyos elementos se corresponden con los de la lista. Está disponible en tiempo de diseño para los controles **ListBox** y **ComboBox** a través de la ventana **Propiedades**, es de sólo lectura en tiempo de ejecución para los controles **DirListBox**, **DriveListBox** y **FileListBox**, y es de lectura y escritura en tiempo de ejecución para los controles **ComboBox** y **ListBox**.

Sintaxis

objeto.**List**(*índice*) [= *cadena*]

La sintaxis de la propiedad **List** consta de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	El número de un elemento específico de la lista.
cadena	Una expresión de cadena que especifica el elemento de la lista.

Comentarios

Utilice esta propiedad para tener acceso a los elementos de la lista.

Para todos los controles excepto **DirListBox**, el índice del primer elemento es 0 y el del último es **ListCount**–1.

Para los controles **DirListBox**, la secuencia de los números de índice se basa en los directorios y subdirectorios actuales cuando el control se crea en tiempo de ejecución. El directorio expandido actualmente se representa con el índice –1. Los directorios situados por encima del expandido actualmente se representan con índices negativos y valores absolutos mayores. Por ejemplo, –2 es el directorio primario del directorio expandido actualmente, y –3 es el del nivel superior siguiente. Los directorios situados bajo el directorio expandido actualmente se numeran entre 0 y **ListCount**–1.

Inicialmente, los controles **ComboBox** y **ListBox** contienen una lista vacía. En el caso de los controles de sistema de archivos, la lista se basa en las condiciones existentes en el momento de crear el control en tiempo de ejecución:

- **DirListBox**: contiene una lista de todos los directorios, usando el intervalo de *-n* a **ListCount**–*1*.
- **DriveListBox**: contiene la lista de conexiones de unidad efectivas.
- **FileListBox**: contiene una lista con los archivos del directorio expandido actualmente que coinciden con la propiedad **Pattern**. No se incluye la ruta de acceso.

La propiedad **List** funciona junto con las propiedades **ListCount** y **ListIndex**.

Para todos los controles a los que se aplican estas propiedades, excepto **DirListBox**, al enumerar una lista de 0 a **ListCount** -1 se obtienen todos los elementos de la lista. En el caso de un control **DirListBox**, la enumeración de la lista de –*n* a **ListCount**–1 permite obtener una lista con todos los directorios y subdirectorios visibles desde el directorio expandido actualmente. En este caso, *n* es el número de niveles de directorios por encima del expandido actualmente.

Nota Para especificar los elementos que desea mostrar en un control **ComboBox** o **ListBox**, utilice el método **AddItem**. Para quitar elementos, utilice el método **RemoveItem**. Para mantener elementos ordenados alfabéticamente, establezca la propiedad **Sorted** del control a **True** antes de agregar elementos a la lista.

El uso de la instrucción `Option Base = 1` en la sección Declaraciones no afecta a la enumeración de elementos de los controles de Visual Basic. El primer elemento siempre es el 0.

Cuando el índice se encuentra fuera del intervalo de entradas reales del cuadro de lista, se obtiene una cadena de longitud cero ("). Por ejemplo, `List(-1)` devuelve una cadena de longitud cero cuando se utiliza para un control **ComboBox** o **ListBox**.

SE APLICA A
ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
ListBox (Control)

Ejemplo de la propiedad List

En este ejemplo se carga un control **ComboBox** con una lista de nombres de sándwiches y se muestra el primer elemento de la lista. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ComboBox** y luego presione F5.

```
Private Sub Form_Load ()
   Combo1.AddItem "Sandwich Toledo"        ' Agregar cada
elemento a la lista.
   Combo1.AddItem "Sandwich Buenos Aires"
   Combo1.AddItem "Sandwich Córdoba"
   Combo1.Text = Combo1.List(0)        ' Mostrar el primer
elemento.
End Sub
```

ListCount (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve el número de elementos de la parte de lista de un control.

Sintaxis

objeto.**ListCount**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

ListCount proporciona información específica de cada control:

- Controles **ComboBox** y **ListBox**: el número de elementos de la lista.
- Control **DirListBox**: el número de subdirectorios del directorio actual.
- Control **DriveListBox**: el número de conexiones con unidades de disco.
- Control **FileListBox**: el número de archivos del directorio actual que coinciden con el valor de la propiedad **Pattern**.

Si no hay seleccionado ningún elemento, el valor de la propiedad **ListIndex** es –1. El primer elemento de la lista corresponde a **ListIndex** = 0 y **ListCount** es siempre una unidad mayor que el valor máximo de **ListIndex**.

SE APLICA A

ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
ListBox (Control)

Ejemplo de la propiedad ListCount

En este ejemplo se carga la lista de fuentes de impresora en un control **ComboBox**, se muestra el primer elemento de la lista y se escribe el número total de fuentes. Cada vez que se hace clic en el botón de comando, todos los elementos de la lista cambian a mayúsculas o a minúsculas. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ComboBox** (**Style** = 2) y un control **CommandButton**, presione F5 y haga clic en el control **CommandButton**.

BIBLIOTECA
UNADECA
ALAJUELA COSTA RICA

```
Private Sub Form_Load ()
   Dim I   ' Declarar la variable.
   AutoRedraw = True   ' Establecer AutoRedraw.
   For I = 0 To Printer.FontCount - 1   ' Poner los nombre
de las fuentes                                          ' en
una lista.
      Combo1.AddItem Printer.Fonts(I)
   Next I
   Combo1.ListIndex = 0   ' Establecer el texto del primer
elemento.
   ' Imprimir la información de ListCount en el formulario.
   Print "Número de fuentes de impresora: ";
Combo1.ListCount
End Sub
Private Sub Command1_Click ()
   Static UpperCase
   Dim I   ' Declarar variable.
   For I = 0 To Combo1.ListCount - 1   ' Bucle por la
lista.
      If UpperCase Then
         Combo1.List(I) = UCase(Combo1.List(I))
      Else
         Combo1.List(I) = LCase(Combo1.List(I))
      End If
   Next I
   UpperCase = Not UpperCase   ' Cambiar mayúsculas y
minúsculas.
End Sub
```

ListIndex (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el índice del elemento seleccionado actualmente en el control. No está disponible en tiempo de diseño.

Sintaxis

objeto.**ListIndex** [= *índice*]

La sintaxis de la propiedad **ListIndex** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Una expresión numérica que especifica el índice del elemento actual, según se indica en Valores.

Valores
Los valores admitidos para *índice* son:

Valor	Descripción
–1	(Predeterminado para los controles **ComboBox**, **DirListBox** y **DriveListBox**) Indica que actualmente no hay ningún elemento seleccionado. Para un control **ComboBox** indica que el usuario ha escrito texto nuevo en la parte de cuadro de texto. En el caso de un control **DirListBox**, indica el índice de la ruta de acceso actual. Para un control **DriveListBox**, indica el índice de la unidad actual cuando el control se crea en tiempo de ejecución.
n	(Predeterminado para los controles **FileListBox** y **ListBox**) Número que indica el índice del elemento seleccionado actualmente.

Comentarios
La expresión `List(List1.ListIndex)` devuelve la cadena correspondiente al elemento seleccionado actualmente.
El primer elemento de la lista corresponde a **ListIndex** = 0 y **ListCount** es siembre una unidad mayor que el valor máximo de **ListIndex**.

Para los controles en los que el usuario puede realizar selecciones múltiples, el comportamiento de esta propiedad depende del número de elementos seleccionado. Si se selecciona un único elemento, **ListIndex** devuelve el índice de ese elemento. Cuando hay una selección múltiple, **ListIndex** devuelve el índice del elemento contenido en el rectángulo de enfoque, esté seleccionado o no.

SE APLICA A
ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
ListBox (Control)

Ejemplo de la propiedad ListIndex

En este ejemplo se muestran los nombres de tres jugadores en un control **ListBox** y el sueldo del jugador seleccionado en un control **Label**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ComboBox** y un control **Label**, presione F5 y elija un nombre en el control **ComboBox**.

```
Dim Player(0 To 2)    ' Dimensionar dos matrices.
Dim Salary(0 To 2)
Private Sub Form_Load ()
   Dim I    ' Declarar variable.
   AutoSize = True
   Player(0) = "Daniel Rendich"    ' Introducir los datos en
las matrices.
```

```
    Player(1) = "Federico Couto"
    Player(2) = "Ernesto Méndez"
    Salary(0) = "234.500$"
    Salary(1) = "158.900$"
    Salary(2) = "1.030.500$"
    For I = 0 To 2   ' Agregar los nombres a la lista.
       Combo1.AddItem Player(I)
    Next I
    Combo1.ListIndex = 0    ' Mostrar el primer elemento de
la lista.
End Sub

Private Sub Combo1_Click ()
    ' Mostrar el salario correspondiente al nombre.
    Label1.Caption = Salary(Combo1.ListIndex)
End Sub
```

Locked (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica si un control se puede modificar.

Sintaxis

objeto.**Locked** [= *booleano*]

La sintaxis de la propiedad **Locked** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica si el control se puede modificar, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	Control **TextBox**: el texto del control se puede desplazar y resaltar, pero no se puede modificar. El programa puede cambiar el texto si cambia la propiedad **Text**. Objeto **Column**: los valores de la columna no se pueden modificar. Objeto **ComboBox**: no puede escribir en el cuadro de texto.
False	Control **TextBox**: puede modificar el texto del control. Objeto **Column**: puede modificar los valores de la columna. Objeto **ComboBox**: puede escribir en el cuadro de texto y desplegar su lista.

Comentarios
En el objeto **Column**, el valor predeterminado de **Locked** es el valor de la propiedad **DataUpdatable** del campo subyacente; sin embargo, si **Column** está desvinculado o el origen de datos no acepta **DataUpdatable**, el valor predeterminado es **True**. Si **DataUpdatable** en el campo subyacente es **False**, el establecer esta propiedad a **True** no produce ningún error. Sin embargo, si el control intenta escribir los datos modificados en la base de datos se produce un error.

Para el control **ComboBox**, cuando **Locked** está establecida a **True**, el usuario no puede cambiar ningún dato, pero puede resaltar datos en el cuadro de texto y copiarlos. Esta propiedad no afecta al acceso desde el programa al control **ComboBox**.

SE APLICA A LAS PROPIEADES, METODIS Y EVENTOS DE TextBox(Control)

TextBox (Control)

Vea también Ejemplo Propiedades Métodos Eventos
Un control **TextBox**, también denominado control de campo de edición o control de edición, muestra información introducida en tiempo de diseño, introducida por el usuario o asignada al control en código en tiempo de ejecución.

Sintaxis
TextBox

Comentarios
Para mostrar múltiples líneas de texto en un control **TextBox**, establezca la propiedad **MultiLine** a **True**. Si un **TextBox** de múltiples líneas no tiene una barra de desplazamiento horizontal, el texto se ajustará automáticamente aunque se ajuste el tamaño de **TextBox**. Para personalizar la combinación de barra de desplazamiento en un

TextBox, establezca la propiedad **ScrollBars**.
Las barras de desplazamiento siempre aparecerán en el **TextBox** cuando su propiedad **MultiLine** se haya establecido a **True** y su propiedad **ScrollBars** a cualquier cosa excepto **None** (0).

Si establece la propiedad **MultiLine** a **True**, podrá emplear la propiedad **Alignment** para establecer la alineación del texto dentro del **TextBox**. El texto está justificado a la izquierda de forma predeterminada. Si la propiedad **MultiLine** es **False**, el valor de la propiedad **Alignment** no tiene ningún efecto.
Un control **TextBox** también puede actuar como vínculo de destino en una conversación DDE.

PROPIEDADES DE TextBox (Control)

Alignment (Propiedad)
Appearance (Propiedad)
BackColor, ForeColor (Propiedades)
BorderStyle (Propiedad)
Container (Propiedad)
DataChanged (Propiedad)
DataField (Propiedad)
DataFormat (Propiedad)
DataMember (Propiedad)
DataSource (Propiedad)
DragIcon (Propiedad)
DragMode (Propiedad)
Enabled (Propiedad)
Font (Propiedad)
FontBold, FontItalic, FontStrikethru, FontUnderline (Propiedades)
FontName (Propiedad)
FontSize (Propiedad)
Height, Width (Propiedades)
HelpContextID (Propiedad)
HideSelection (Propiedad)
Index (Propiedad, matriz de controles)
Left, Top (Propiedades)
LinkItem (Propiedad)
LinkMode (Propiedad)
LinkTimeout (Propiedad)
LinkTopic (Propiedad)
Locked (Propiedad)
MaxLength (Propiedad)
MouseIcon (Propiedad)
MousePointer (Propiedad)
MultiLine (Propiedad)
Name (Propiedad)
OLEDragMode (Propiedad)
OLEDropMode (Propiedad)
OLEDropMode (Propiedad, controles ActiveX)
Parent (Propiedad)
PasswordChar (Propiedad)
RightToLeft (Propiedad)
ScrollBars (Propiedad)
SelLength, SelStart, SelText (Propiedades)
TabIndex (Propiedad)
TabStop (Propiedad)
Tag (Propiedad)

Text (Propiedad)
ToolTipText (Propiedad)
Visible (Propiedad)
WhatsThisHelpID (Propiedad)
hWnd (Propiedad)

METODOS DE TextBox (Control)

Drag (Método)
LinkExecute (Método)
LinkPoke (Método)
LinkRequest (Método)
LinkSend (Método)
Move (Método)
OLEDrag (Método)
Refresh (Método)
SetFocus (Método)
ShowWhatsThis (Método)
ZOrder (Método)

EVENTOS DE TextBox (Control)

Change (Evento)
Click (Evento)
DblClick (Evento)
DragDrop (Evento)
DragOver (Evento)
GotFocus (Evento)
KeyDown, KeyUp (Eventos)
KeyPress (Evento)
LinkClose (Evento)
LinkError (Evento)
LinkNotify (Evento)
LinkOpen (Evento)
LostFocus (Evento)
MouseDown, MouseUp (Eventos)
MouseMove (Evento)
OLECompleteDrag (Evento)
OLECompleteDrag (Evento, controles ActiveX)
OLEDragDrop (Evento)
OLEDragOver (Evento)
OLEDragOver (Evento, controles ActiveX)
OLEGiveFeedback (Evento)
OLESetData (Evento)
OLEStartDrag (Evento)
Validate (Evento)

LinkItem (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece los datos pasados a un control de destino en una conversación DDE con otra aplicación.

Sintaxis

objeto.**LinkItem** [= *cadena*]

La sintaxis de la propiedad **LinkItem** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
cadena	Una expresión de cadena que especifica los datos que se pasan al control de destino.

Comentarios

Esta propiedad corresponde al argumento *elemento* de la sintaxis DDE estándar, con *aplicación*, *tema* y *elemento* como argumentos. Para establecer esta propiedad, especifique una unidad de datos reconocible de una aplicación como referencia; por ejemplo, una referencia de celda como "R1C1" en Microsoft Excel.

Use **LinkItem** en combinación con la propiedad **LinkTopic** para especificar el vínculo de datos completo de un control de destino a una aplicación de origen. Para activar este vínculo, establezca la propiedad **LinkMode**.

LinkItem se establece solamente en un control usado como destino. Cuando un formulario de Visual Basic es un origen de una conversación DDE, el nombre de cualquier control **Label**, **PictureBox** o **TextBox** del formulario puede ser el argumento *elemento* de la cadena *aplicación*|*tema*!*elemento* usada por el destino. Por ejemplo, la siguiente sintaxis representa una referencia válida desde Microsoft Excel a una aplicación de Visual Basic:

=*AplicaciónVisBasic*|*MiForm*!*TextBox1*

Podría escribir la sintaxis anterior para una celda de destino en la barra de fórmulas de Microsoft Excel.

Un control DDE puede actuar como destino y como origen al mismo tiempo, provocando un bucle infinito si una pareja destino-origen es al mismo tiempo una pareja origen-destino. Por ejemplo, un control **TextBox** puede ser un origen (a través de su formulario primario) y un destino de la misma celda de Microsoft Excel. Cuando los datos de un control **TextBox** de Visual Basic cambian, se envían datos a Microsoft Excel, la celda de Microsoft Excel cambia, se envían los cambios al **TextBox**, y así sucesivamente, provocando el bucle.

Para evitar estos bucles, use elementos relacionados pero no idénticos como vínculos origen-destino y destino-origen en ambas direcciones entre las aplicaciones. Por ejemplo, en Microsoft Excel, use celdas relacionadas (precedentes o dependientes) para vincular una hoja de cálculo con un control de Visual Basic, evitando usar un único elemento como destino y como origen. Documente todas las parejas *aplicación*|*tema* establecidas si incluye un comando Pegar vínculo para su uso en tiempo de ejecución.

Nota Establecer un vínculo de datos permanente en tiempo de diseño con el comando **Pegar vínculo** del menú **Edición** también establece las propiedades **LinkMode**, **LinkTopic** y **LinkItem**. Esto crea un vínculo que se guarda con el formulario. Cada vez que se carga el formulario, Visual Basic intenta volver a establecer la conversación.

SE APLICA A

Label (Control)
PictureBox (Control)
TextBox (Control)

Ejemplo de las propiedades LinkItem, LinkMode, LinkTopic

En este ejemplo, cada clic del *mouse* hace que una celda de una hoja de cálculo de Microsoft Excel actualice el contenido de un control **TextBox** de Visual Basic. Para probar este ejemplo, inicie Microsoft Excel, abra una nueva hoja de cálculo llamada Hoja1 y escriba algún dato en la primera columna. En Visual Basic, cree un formulario con un control **TextBox**. Pegue el código en la sección Declaraciones y después presione F5 para ejecutar el programa.

```
Private Sub Form_Click ()
   Dim CurRow As String
   Static Row    ' Número de fila de la hoja de cálculo.
   Row = Row + 1    ' Incrementa la fila.
   If Row = 1 Then    ' Sólo la primera vez.
      ' Se asegura que el vínculo no está activo.
      Text1.LinkMode = 0
      ' Establece el nombre de aplicación y de tema.
      Text1.LinkTopic = "Excel|Hoja1"
      Text1.LinkItem = "R1C1"    ' Establece LinkItem.
      Text1.LinkMode  =  1        '  Establece  LinkMode  a
automático.
   Else
      ' Actualiza la fila en el elemento de datos.
      CurRow = "R" & Row & "C1"
      Text1.LinkItem = CurRow    ' Establece LinkItem.
   End If
End Sub
```

LinkMode (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el tipo de vínculo usado en una conversación DDE y activa la conexión de la forma siguiente:

- **Control**: permite que un control de destino de un formulario de Visual Basic inicie una conversación, como se especifica en las propiedades **LinkTopic** y **LinkItem**.

- **Formulario**: permite que una aplicación de destino inicie una conversación con un formulario de origen de Visual Basic, como se especifica en la expresión *aplicación|tema!elemento* de la aplicación de destino.

Sintaxis

objeto.**LinkMode** [= *número*]

La sintaxis de la propiedad **LinkMode** consta de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Un entero que especifica el tipo de conexión, como se describe en Valores.

Valores

En controles usados como destinos en conversaciones DDE, los valores admitidos para *número* son:

Constante	**Valor**	**Descripción**
vbLinkNone	0	(Predeterminado) Ninguno: no hay interacción DDE.
vbLinkAutomatic	1	Automático: el control de destino se actualiza cada vez que los datos vinculados cambian.
vbLinkManual	2	Manual: el control de destino se actualiza sólo cuando se invoca el método **LinkRequest**.
VbLinkNotify	3	Notificar: se produce un evento LinkNotify cuando los datos vinculados cambian, pero el control de destino se actualiza sólo cuando se invoca **LinkRequest**.

En los formularios usados como origen en conversaciones DDE, los valores admitidos para *número* son:

Constante	**Valor**	**Descripción**
vbLinkNone	0	(Predeterminado) Ninguno: no hay interacción DDE. Ninguna aplicación de destino puede iniciar una conversación con el formulario de origen como tema y ninguna aplicación puede llevar datos al formulario. Si **LinkMode** es 0 (Ninguno) en tiempo de diseño, no será posible cambiar a 1 (Origen) en tiempo de ejecución.
vbLinkSource	1	Origen: permite que cualquier control **Label**, **PictureBox** o **TextBox** de un formulario suministre datos a cualquier aplicación de destino que establezca una conversación DDE con el formulario. Si dicho vínculo existe, Visual Basic notifica automáticamente al destino cuando el

contenido de un control ha cambiado. Además, una aplicación de destino puede llevar datos a cualquier control **Label**, **PictureBox** o **TextBox** del formulario. Si **LinkMode** es 1 (Origen) en tiempo de diseño, puede cambiarse a 0 (Ninguno) y a 1 de nuevo en tiempo de ejecución.

Comentarios

Las siguientes condiciones también se aplican a la propiedad **LinkMode**:

- Establecer **LinkMode** a un valor distinto de cero en un control de destino hace que Visual Basic intente iniciar la conversación especificada en las propiedades **LinkTopic** y **LinkItem**. El origen actualiza el control de destino de acuerdo con el tipo de vínculo especificado (automático, manual o notificar).

- Si una aplicación de origen termina una conversación con un control de destino de Visual Basic, el valor de **LinkMode** de dicho control cambia a 0 (Ninguno).

- Si en un formulario deja **LinkMode** establecida de forma predeterminada a 0 (Ninguno) en tiempo de diseño, **LinkMode** no puede cambiarse en tiempo de ejecución. Si quiere que un formulario actúe como origen, debe establecer **LinkMode** a 1 (Origen) en tiempo de diseño. El valor de **LinkMode** no puede cambiarse en tiempo de ejecución.

Nota Cuando establece un vínculo de datos permanente en tiempo de diseño con el comando **Pegar vínculo** del menú **Edición**, también establece las propiedades **LinkMode**, **LinkTopic** y **LinkItem**. Esto crea un vínculo que se guarda con el formulario. Cada vez que se carga el formulario, Visual Basic intenta volver a establecer la conversación.

SE APLICA A

Form (Objeto), Forms (Colección)
Label (Control)
MDIForm (Objeto)
PictureBox (Control)
TextBox (Control)

LinkTimeout (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece la cantidad de tiempo que un control espera una respuesta a un mensaje DDE.

Sintaxis
objeto.**LinkTimeout** [= *número*]
La sintaxis de la propiedad **LinkTimeout** consta de las siguientes partes:

Parte Descripción
objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número Una expresión numérica que especifica el tiempo de espera.

Comentarios
De forma predeterminada, la propiedad **LinkTimeout** se establece a 50 (equivalente a 5 segundos). Puede especificar otros valores en décimas de segundo.
El tiempo de respuesta DDE desde las aplicaciones de origen varía. Use esta propiedad para ajustar el tiempo que un control de destino espera la respuesta de una aplicación de origen. Si usa **LinkTimeout**, puede evitar la generación de un error de Visual Basic si una aplicación de origen tarda mucho en responder.

Nota La duración mayor de tiempo que un control puede esperar es 65.535 décimas de segundo, es decir, 1 hora y 49 minutos. Al establecer **LinkTimeout** a –1 se indica al control que espere la respuesta en una conversación DDE durante el intervalo máximo de tiempo. El usuario puede forzar que el control deje de esperar si presiona la tecla ESC.

SE APLICA A
Label (Control)
PictureBox (Control)
TextBox (Control)

LinkTopic (Propiedad)

Vea también Ejemplo Se aplica a
Para un control de destino, devuelve o establece la aplicación de origen y el tema correspondiente (el agrupamiento de datos fundamental utilizado en esa aplicación). Utilice **LinkTopic** con la propiedad **LinkItem** para especificar el vínculo de datos completo.

Para un formulario de origen, devuelve o establece el tema al que el formulario de origen responde en una conversación DDE.

Sintaxis
objeto.**LinkTopic** [= *valor*]
La sintaxis de la propiedad **LinkTopic** consta de las siguientes partes:

Parte Descripción
objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor Una expresión de cadena que especifica un elemento de la sintaxis de DDE.

Comentarios
La propiedad **LinkTopic** consiste en una cadena que proporciona parte de la información necesaria para establecer un vínculo de destino o de origen. La cadena que utilice dependerá de si se trabaja con un control de destino o con un formulario de origen. Cada cadena corresponde a uno o más elementos de la sintaxis estándar de DDE, como son *aplicación*, *tema* y *elemento*.

Nota Aunque la definición estándar de un vínculo DDE incluye los elementos *aplicación*, *tema* y *elemento*, la sintaxis real utilizada en las aplicaciones para un vínculo de destino con una aplicación de origen puede variar ligeramente. Por ejemplo, en Microsoft Excel se utiliza la sintaxis siguiente:
aplicación|tema!elemento

En Microsoft Word para Windows, la sintaxis es:
aplicación tema elemento

(No utilice el carácter de canalización [|] ni el signo de admiración [!].)
En una aplicación de Visual Basic, la sintaxis es:
aplicación|tema

El signo de admiración de *tema* está implícito.
Control de destino Para establecer la propiedad **LinkTopic** de un control de destino, utilice una cadena con la sintaxis *aplicación|tema*, según se indica a continuación:

- *aplicación* es el nombre de la aplicación de la que se solicitan los datos y normalmente es el nombre del archivo ejecutable sin la extensión. Por ejemplo, Excel (para Microsoft Excel).

- El carácter de canalización (|, o el código de carácter 124) separa la aplicación del tema.

- *tema* es el agrupamiento fundamental de datos utilizado en la aplicación de origen. Por ejemplo, una hoja de cálculo en Microsoft Excel.

Además, y sólo para los controles de destino, es necesario establecer la propiedad relacionada **LinkItem** para especificar el componente *elemento* del vínculo. Una referencia a una celda, como L1C1, corresponderá a un elemento de una hoja de cálculo de Microsoft Excel.

Formulario de origen Para establecer la propiedad **LinkTopic** de un formulario de origen, establezca *valor* a un identificador adecuado para el formulario. La aplicación de destino usará esta cadena como argumento *tema* al establecer un vínculo DDE con el formulario. Aunque esta cadena es todo lo que necesita para establecer la propiedad **LinkTopic** de un formulario de origen de Visual Basic, la aplicación de destino también debe especificar lo siguiente:

- El componente *aplicación* que utiliza la aplicación de destino, que será el nombre de proyecto de Visual Basic sin la extensión .vbp (si la aplicación se está ejecutando en el entorno de desarrollo de Visual Basic) o el nombre de archivo de la aplicación de Visual Basic sin la extensión .exe (si se ejecuta como un archivo ejecutable). La propiedad **EXEName** del objeto **App** proporciona esta cadena en el código de Visual Basic, a menos que el usuario haya cambiado el nombre del archivo. (**EXEName** siembre devuelve el nombre de archivo real de la aplicación en el disco y DDE utiliza siempre el nombre original especificado en el cuadro de diálogo **Propiedades del proyecto**.)

- El componente *elemento* que utiliza la aplicación de destino, que corresponde al valor de la propiedad **Name** del control **Label**, **PictureBox** o **TextBox** del formulario de origen.

La sintaxis siguiente es un ejemplo de una referencia válida desde Microsoft Excel a una aplicación de Visual Basic que actúa como origen:
=AplicaciónVisBasic|FormN!TextBox1
Esta referencia podría especificarse en una celda de destino a través de la barra de fórmulas de Microsoft Excel.

Para activar el vínculo de datos establecido con **LinkTopic**, establezca la propiedad **LinkMode** al valor distinto de cero que corresponda al tipo de vínculo que desee. Como norma general, debe establecer **LinkMode** después de **LinkTopic**. Para un control de destino, al modificar **LinkTopic** se interrumpe un vínculo existente y se termina la conversación DDE. Para un formulario de origen, al modificar **LinkTopic** se interrumpen todos los vínculos de destino que utilizan ese tema. Por estas razones, debe establecer siempre a 0 la propiedad **LinkMode** antes de modificar **LinkTopic**. Una vez modificada la propiedad **LinkTopic** de un control de destino, deberá establecer **LinkMode** a los valores 1 (automático), 2 (manual) o 3 (con notificación) para iniciar una conversación con el nuevo tema.

Nota Al crear un vínculo de datos permanente en tiempo de diseño con el comando **Pegar vínculo** del menú **Edición**, también se establecen las propiedades **LinkMode**, **LinkTopic** y **LinkItem**. De esta forma se crea un vínculo que se guarda con el formulario. Cada vez que se cargue el formulario, Visual Basic intentará volver a establecer la conversación.

SE APLICA A
Form (Objeto), Forms (Colección)
Label (Control)
MDIForm (Objeto)
PictureBox (Control)
TextBox (Control)

Ejemplo de las propiedades LinkItem, LinkMode, LinkTopic

En este ejemplo, cada clic del *mouse* hace que una celda de una hoja de cálculo de Microsoft Excel actualice el contenido de un control **TextBox** de Visual Basic. Para probar este ejemplo, inicie Microsoft Excel, abra una nueva hoja de cálculo llamada Hoja1 y escriba algún dato en la primera columna. En Visual Basic, cree un formulario con un control **TextBox**. Pegue el código en la sección Declaraciones y después presione F5 para ejecutar el programa.

```
Private Sub Form_Click ()
   Dim CurRow As String
   Static Row    ' Número de fila de la hoja de cálculo.
   Row = Row + 1    ' Incrementa la fila.
   If Row = 1 Then    ' Sólo la primera vez.
      ' Se asegura que el vínculo no está activo.
      Text1.LinkMode = 0
      ' Establece el nombre de aplicación y de tema.
      Text1.LinkTopic = "Excel|Hoja1"
      Text1.LinkItem = "R1C1"    ' Establece LinkItem.
      Text1.LinkMode  =  1        '  Establece  LinkMode  a
automático.
   Else
      ' Actualiza la fila en el elemento de datos.
      CurRow = "R" & Row & "C1"
      Text1.LinkItem = CurRow    ' Establece LinkItem.
   End If
End Sub
```

Max, Min (Propiedades, barra de desplazamiento)

Vea también Ejemplo Se aplica a

- **Max**: devuelve o establece el valor máximo de la propiedad **Value** de una barra de desplazamiento cuando el cuadro de desplazamiento se encuentra en el extremo inferior o derecho. Para el control **ProgressBar**, devuelve o establece su valor máximo.

- **Min**: devuelve o establece el valor mínimo de la propiedad **Value** de una barra de desplazamiento cuando el cuadro de desplazamiento se encuentra en el extremo superior o izquierdo. Para el control **ProgressBar**, devuelve o establece su valor mínimo.

Sintaxis

objeto.**Max** [= *valor*]
objeto.**Min** [= *valor*]

Las sintaxis de las propiedades **Max** y **Min** constan de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Una expresión numérica que especifica el valor máximo o mínimo de la propiedad **Value**, según se describe en Valores.

Valores

En cada propiedad puede especificar un número entero entre -32.768 y 32.767, inclusive. Los valores predeterminados son:

- **Max**: 32767.
- **Min**: 0.

Comentarios

El entorno operativo Microsoft Windows establece automáticamente los intervalos para las barras de desplazamiento de los formularios, controles **ComboBox** y controles **ListBox**, haciéndolos proporcionales a su contenido. Sin embargo, para un control de barra de desplazamiento (**HScrollBar** o **VScrollBar**), debe especificar estos intervalos. Utilice **Max** y **Min** para establecer un intervalo adecuado al uso de la barra de desplazamiento, por ejemplo como dispositivo de entrada o como indicador de velocidad o cantidad.

Normalmente, **Max** y **Min** se establecen en tiempo de diseño. También puede establecerlas en el código en tiempo de ejecución si el intervalo de desplazamiento deba cambiar dinámicamente, como al agregar registros a una base de datos por la que sea posible desplazarse. Los incrementos de desplazamiento máximo y mínimo de una barra de desplazamiento se establecen mediante las propiedades **LargeChange** y **SmallChange**.

Nota Si en **Max** establece un valor menor que el de **Min**, el valor máximo se establecerá en el extremo izquierdo en el caso de una barra horizontal, o en el superior en el caso de una barra de desplazamiento vertical. La propiedad **Max** de un control **ProgressBar** debe ser siempre mayor que la propiedad **Min**, y ésta debe ser siempre mayor o igual que cero.

Las propiedades **Max** y **Min** definen el intervalo de un control. El valor predeterminado de la propiedad **Min** de un control **ProgressBar** es 0 y el de la propiedad **Max** es 100, representando el porcentaje de duración de una operación.

SE APLICA A VER: LOS METODOS PROPIEDADES Y EVETNOS DE HScrollBar, VScrollBar (Controles)

MouseIcon (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un icono personalizado de *mouse* (ratón).

Sintaxis
objeto.**MouseIcon = LoadPicture(***ruta***)**
objeto.**MouseIcon [=** *imagen***]**
La sintaxis de la propiedad **MouseIcon** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
ruta	Una expresión de cadena que especifica la ruta y el nombre del archivo que contiene el icono personalizado.
Imagen	la propiedad **Picture** de un objeto **Form**, de un control **PictureBox** o de un control **Image**.

Comentarios
La propiedad **MouseIcon** proporciona un icono de *mouse* personalizado que se utiliza cuando el valor de la propiedad **MousePointer** es 99.

La propiedad **MouseIcon** proporciona al programa un acceso sencillo a cursores personalizados de cualquier tamaño, con la zona activa en cualquier ubicación deseada. Visual Basic no carga archivos de cursores animados (.ani), aunque las versiones de 32 bits de Windows permitan estos cursores.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
MSFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RichTextBox (Control)
Screen (Objeto)

TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad MouseIcon

Este ejemplo ilustra la forma en que la propiedad **MouseIcon** establece un icono de *mouse* (ratón) personalizado. Para probarlo, cree un control **ListBox** en un formulario y establezca a 1 ó 2 la propiedad **MultiSelect**. En tiempo de ejecución, seleccione uno o más elementos. Aparecerán iconos distintos dependiendo de si ha seleccionado múltiples elementos o uno solo.

```
Private Sub Form_Load ()
   ' Pone algunos elementos en el cuadro de lista.
   List1.AddItem "Selección 1"
   List1.AddItem "Selección 2"
   List1.AddItem "Selección 3"
   List1.AddItem "Selección 4"
   List1.AddItem "Selección 5"
End Sub

Private Sub List1_MouseDown (Button As Integer, Shift As
Integer, X As Single, Y As Single)
   ' Establece el icono de mouse personalizado para
múltiples elementos.
   If List1.SelCount > 1 Then
      List1.MouseIcon =
LoadPicture("ICONS\COMPUTER\MOUSE04.ICO")
      List1.MousePointer = 99
   Else    ' Establece el icono personalizado de mouse para
un único                ' elemento.
      List1.MouseIcon =
LoadPicture("ICONS\COMPUTER\MOUSE02.ICO")
      List1.MousePointer = 99
   End If
End Sub
```

MousePointer (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica el tipo de puntero de *mouse* que aparece cuando se sitúa el *mouse* sobre un área determinada de un objeto en tiempo de ejecución.

Sintaxis

objeto.**MousePointer** [= *valor*]

La sintaxis de la propiedad **MousePointer** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Un número entero que especifica el tipo de puntero de *mouse* que aparece, según se describe en Valores.

Valores

Los valores admitidos para *valor* son:

Constante	Valor	Descripción
vbDefault	0	(Predeterminado) Forma determinada por el objeto.
vbArrow	1	Flecha.
vbCrosshair	2	Cruz (puntero en forma de cruz).
vbIbeam	3	Forma de I.
vbIconPointer	4	Icono (pequeño cuadrado dentro de otro cuadrado).
vbSizePointer	5	Tamaño (flecha de cuatro puntas: norte, sur, este y oeste).
VbSizeNESW	6	Tamaño NE SO (flecha doble que apunta al nordeste y al sudoeste).
vbSizeNS	7	Tamaño N S (flecha doble que apunta al norte y al sur).
vbSizeNWSE	8	Tamaño NO SE (flecha doble que apunta al noroeste y al sudeste).
vbSizeWE	9	Tamaño O E (flecha doble que apunta al oeste y al este).
vbUpArrow	10	Flecha hacia arriba.
vbHourglass	11	Reloj de arena (espera).
vbNoDrop	12	No colocar.
vbArrowHourglass	13	Flecha y reloj de arena.
vbArrowQuestion	14	Flecha y signo de interrogación.
vbSizeAll	15	Ajustar todo.
vbCustom	99	Icono personalizado especificado en la propiedad **MouseIcon**.

Comentarios

Puede usar esta propiedad cuando desee indicar los cambios en la funcionalidad al desplazar el puntero del *mouse* por los controles de un formulario o cuadro de diálogo. La opción de reloj de arena (11) es útil para indicar que el usuario debe esperar a que termine un proceso u operación.

Nota Si la aplicación llama a DoEvents, la propiedad **MousePointer** puede variar temporalmente para un componente ActiveX.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)

FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
Screen (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad MousePointer

En este ejemplo se cambia el puntero del *mouse* por un reloj de arena mientras se dibujan círculos en la pantalla y al final del procedimiento el puntero adopta la forma que tenía originalmente. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario. Presione F5 para ejecutar el programa y luego haga clic en el formulario.

```
Private Sub Form_Click ()
   Dim I    ' Declara una variable.
   ' Cambia el puntero del mouse por un reloj de arena.
   Screen.MousePointer = vbHourglass
   ' Establece un color aleatorio y dibuja círculos en el
formulario.
   For I = 0 To ScaleWidth Step 50
      ForeColor = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
      Circle (I, ScaleHeight * Rnd), 400
   Next
   ' Vuelve al puntero normal.
   Screen.MousePointer = vbDefault
End Sub
```

MultiSelect (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica si el usuario puede realizar selecciones múltiples en un control **FileListBox** o **ListBox**, y la forma de llevarlas a cabo. Es de sólo lectura en tiempo de ejecución.

Sintaxis
objeto.**MultiSelect**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Valores
Los valores admitidos para la propiedad **MultiSelect** son:

Valor	Descripción
0	(Predeterminado) No se permiten selecciones múltiples.
1	Selección múltiple simple. Al hacer clic con el *mouse* o presionar la BARRA ESPACIADORA se selecciona o se desactiva un elemento de la lista. (Las teclas de dirección desplazan el enfoque.)
2	Selección múltiple extendida. Al presionar MAYÚS y hacer clic, o al presionar MAYÚS y una de las teclas de dirección (FLECHA ARRIBA, FLECHA ABAJO, FLECHA IZQUIERDA o FLECHA DERECHA) se extiende la selección desde el elemento seleccionado anteriormente hasta el elemento actual. Al presionar CTRL y hacer clic con el *mouse* se selecciona o desactiva un elemento de la lista.

SE APLICA A
FileListBox (Control)
ListBox (Control)

Ejemplo de la propiedad MultiSelect

En este ejemplo se llena un control **ListBox** con los nombres de las fuentes de pantalla y se ilustra la forma en que la propiedad **MultiSelect** afecta al comportamiento de un control **ListBox**. Para probar este ejemplo, cree dos controles **ListBox** y un control **CommandButton** en un formulario. Establezca a 1 ó 2 la propiedad **MultiSelect** del primer control **ListBox**. En tiempo de ejecución, seleccione varios elementos en el primer control **ListBox** y haga clic en el control **CommandButton**. En el segundo **ListBox** se muestran todos los elementos seleccionados. Ejecute el ejemplo varias veces con distintos valores en la propiedad **MultiSelect**. Pegue el código en la sección Declaraciones y presione F5 para ejecutar el programa.

```
Private Sub Form_Load ()
   Dim I    ' Declara una variable.
   ' Llena el cuadro de lista con los nombres de las
fuentes de pantalla.
   For I = 0 To Screen.FontCount - 1
      List1.AddItem Screen.Fonts(I)
   Next I
End Sub

Private Sub Command1_Click ()
```

```
   Dim I    ' Declara una variable.
   ' Borra todos los elementos de la lista.
   List2.Clear
   ' Si un elemento está seleccionado, se agrega a List2.
   For I = 0 To List1.ListCount - 1
      If List1.Selected(I) Then
         List2.AddItem List1.List(I)
      End If
   Next I
End Sub
```

Name (Propiedad)

Vea también Ejemplo Se aplica a

- Devuelve el nombre utilizado en el código para identificar un formulario, un control o un objeto de acceso a datos. Es de sólo lectura en tiempo de ejecución.

- Devuelve o establece el nombre de un objeto fuente.

Sintaxis

objeto.**Name**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a. Si se omite *objeto*, se supondrá que el formulario asociado al módulo del formulario activo es *objeto*.

Comentarios

El nombre predeterminado para objetos nuevos es el tipo de objeto y un entero único. Por ejemplo, el primer objeto **Form** nuevo será Form1, un objeto **MDIForm** nuevo será MDIForm1 y el tercer control **TextBox** que cree en un formulario será Text3.

La propiedad **Name** de un objeto debe comenzar por una letra y puede tener un máximo de 40 caracteres. Puede incluir números y caracteres de subrayado (_), pero no signos de puntuación ni espacios en blanco. Un formulario no puede tener el mismo nombre que otro objeto público como **Clipboard**, **Screen** o **App**. Aunque el valor de la propiedad **Name** puede ser una palabra clave, un nombre de propiedad o el nombre de otro objeto, esto puede crear conflictos en el código.

Puede usar la propiedad **Name** de un formulario con la instrucción **Dim** en tiempo de ejecución para crear otras instancias del formulario. No puede haber dos formularios con el mismo nombre en tiempo de diseño.

Puede crear un conjunto de controles del mismo tipo si establece la propiedad **Name** al mismo valor. Por ejemplo, cuando establece el nombre de todos los botones de opción en

un grupo como MiOpc, Visual Basic asigna valores únicos a la propiedad **Index** de cada control con el fin de distinguir unos de otros en la matriz. Dos controles de distinto tipo no pueden tener el mismo nombre.

Nota Aunque Visual Basic usa a menudo el valor de la propiedad **Name** como valor predeterminado de las propiedades **Caption**, **LinkTopic** y **Text**, el cambio de una de estas propiedades no afecta a las otras.

Cambiar mayúsculas/minúsculas del valor de la propiedad **Name** para un Formulario u otro módulo sin cambiar el nombre puede producir un mensaje de error del tipo "Conflicto de nombres" la próxima vez que se cargue el proyecto que contiene el formulario o el módulo. Por ejemplo, cambiar "Form1" a "form1" producirá el error; cambiar "Form1" a "formX" no lo hará.

El error es causado por la forma en que se guardan los nombres de los módulos en el archivo del proyecto; el procedimiento para cambiar nombres dentro del archivo del proyecto no distingue mayúsculas de minúsculas, mientras que el procedimiento para leer nombres al cargar el proyecto sí las distingue.

SE APLICA A
CheckBox (Control)
Clase
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
EventInfo (Objeto)
EventParameter (Objeto)
Extender (Objeto)
FileListBox (Control)
Font (Objeto)
Form (Objeto), Forms (Colección)
Frame (Control)
Function (Control, diseñador de informe de datos)
HScrollBar, VScrollBar (Controles)
Image (Control)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)

Line (Control)
Line (Control, diseñador de informe de datos)
ListBox (Control)
MAPIMessages (Control)
MAPISession (Control)
MDIForm (Objeto)
Masked Edit (Control)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
PictureClip (Control)
PropertyPage (Objeto)
RichTextBox (Control)
Section (Objeto, diseñador de informe de datos)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)
Timer (Control)
UserControl (Objeto)
UserDocument (Objeto)

Negotiate (Propiedad)

Vea también Ejemplo Se aplica a

Establece un valor que determina si un control que puede alinearse se muestra cuando un objeto activo del formulario muestra una o más barras de herramientas. No está disponible en tiempo de ejecución.

Valores

La propiedad **Negotiate** tiene estos valores:

Valor	Descripción
True	Si el control se alinea en el formulario (la propiedad **Align** se establece a un valor distinto de cero), el control permanecerá visible cuando un objeto activo del formulario muestre una barra de herramientas.
False	(Predeterminado) El control no se muestra cuando un objeto activo del formulario muestra una barra de herramientas. La barra de herramientas del objeto activo se muestra en lugar del control.

Comentarios

La propiedad **Negotiate** existe para todos los controles con una propiedad **Align**. Utilice la propiedad **Align** para alinear el control dentro de un objeto **Form** o **MDIForm**; sin embargo, la negociación de barra de herramientas sólo se produce en el **MDIForm**. El control alineado debe estar en el **MDIForm**.

Si la propiedad **NegotiateToolbars** se establece a **False**, el valor de la propiedad **Negotiate** no tendrá ningún efecto.

SE APLICA A: VER PROPIEDADES, METODOS Y EVENTOS DE PictureBox (Control)

OLEDragMode (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica si las operaciones OLE de arrastrar y colocar las controla el componente o el programador.

Sintaxis

objeto.**OLEDragMode** = *modo*

La sintaxis de la propiedad **OLEDragMode** consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
modo	Entero que especifica el método con el que un componente realiza las operaciones OLE de arrastrar y colocar, según se describe en Valores.

Valores

Los valores de *modo* son:

Constante	Valor	Descripción
vbOLEDragManual	0	(Predeterminado) Manual. El programador controla todas las operaciones OLE de arrastrar y colocar.
vbOLEDragAutomatic	1	Automático. El componente controla todas las operaciones OLE de arrastrar y colocar.

Comentarios

Cuando el valor de **OLEDragMode** es **Manual**, para iniciar el arrastre debe llamar al método **OLEDrag**, que desencadena el evento **OLEStartDrag**.

Cuando la propiedad **OLEDragMode** está establecida a **Automatic**, el componente de origen puebla el objeto **DataObject** con los datos que contiene y establece el parámetro *efecto* antes de iniciar el evento OLEStartDrag (además del evento OLESetData y otros eventos de la operación OLE de arrastrar y colocar a nivel del origen) cuando el usuario intenta arrastrar algo fuera del control. Esto le permite controlar la operación de arrastre e intervenir si agrega otros formatos o si anula o desactiva los datos y formatos automáticos mediante los métodos **Clear** y **SetData**.

Si la propiedad **OLEDragMode** del origen está establecida a **Automatic** y no se cargan datos en el evento OLEStartDrag, o si *efectosPosteriores* se establece a **0**, no se produce la operación OLE de arrastrar y colocar.

Nota Si la propiedad **DragMode** de un control está establecida a **Automatic**, se pasa por alto el valor de **OLEDragMode**, ya que tienen preferencia los eventos normales de arrastre y colocación de Visual Basic.

SE APLICA A
ComboBox (Control)
DirListBox (Control)
FileListBox (Control)
Image (Control)
ListBox (Control)
PictureBox (Control)
TextBox (Control)

OLEDropMode (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece el modo en que un componente de destino realiza las operaciones de colocación.

Sintaxis
objeto.**OLEDropMode** [= *modo*]
La sintaxis de la propiedad **OLEDropMode** consta de las partes siguientes:

Parte	Descripción
objeto	Expresión de objeto que da como resultado un objeto de la lista Se aplica a.
modo	Entero enumerado que especifica el método mediante el que un componente realiza las operaciones OLE de arrastrar y colocar, según se describe en Valores.

Valores
Los valores de *modo* son:

Constante	Valor	Descripción
vbOLEDropNone	0	(Predeterminado) Ninguno. El componente de destino no admite operaciones OLE de colocar y muestra el cursor "No colocar".
vbOLEDropManual	1	Manual. El componente de destino desencadena los eventos OLE de colocar, lo que permite al programador controlar la operación en el código.
vbOLEDropAutomatic	2	Automático. El componente de destino acepta automáticamente las operaciones OLE de colocar si el objeto **DataObject** contiene datos en un formato que reconoce. No se producen en el destino eventos del *mouse* ni eventos OLE de arrastrar y colocar cuando en **OLEDropMode** está establecido **vbOLEDropAutomatic**.

Comentarios
Nota El componente de destino examina lo que se está arrastrando sobre él para determinar si debe provocar los eventos OLE de arrastrar y colocar o los de Visual Basic. No hay conflicto posible entre los componentes ni confusión sobre los eventos a desencadenar, ya que sólo es posible arrastrar un objeto de un tipo al mismo tiempo.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
Image (Control)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Parent (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve el formulario, objeto o colección que contiene un control u otro objeto o colección.

Sintaxis
objeto.**Parent**
El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios
Utilice la propiedad **Parent** para tener acceso a las propiedades, los métodos o los controles del primario de un objeto. Por ejemplo:

```
MiBotón.Parent.MousePointer = 4
```

La propiedad **Parent** es útil en una aplicación en la que se pasan objetos como argumentos. Por ejemplo, podría pasar una variable de control a un procedimiento

general en un módulo y usar la propiedad **Parent** para tener acceso a su formulario primario.
No hay ninguna relación entre la propiedad **Parent** y la propiedad **MDIChild**. Sin embargo, hay una relación primario-secundario entre un objeto **MDIForm** y cualquier objeto **Form** que tenga su propiedad **MDIChild** establecida a **True**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)
MAPIMessages (Control)
MAPISession (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
PictureClip (Control)
RemoteData (Control)
RichTextBox (Control)
Shape (Control)
TextBox (Control)
Timer (Control)
UserDocument (Objeto)

Ejemplo de la propiedad Parent

Este ejemplo pasa un control de un formulario que no tiene el enfoque a un procedimiento de un módulo y después muestra el estado del control del formulario

primario. Para probar este ejemplo, cree tres formularios: Form1, que contenga un control **CommandButton**, y Form2 y Form3, cada uno de los cuales contiene un control **CheckBox**. También debe crear un nuevo módulo (haga clic en el comando **Agregar módulo** del menú **Proyecto**). Pegue el código en la sección Declaraciones de los respectivos formularios o módulos y después presione F5 para ejecutar el programa.

```
' Escriba este código en Form1.
Private Sub Form_Load ()
   Form2.Show    ' Muestra todos los formularios.
   Form3.Show
   Form2.AutoRedraw = True
   Form3.AutoRedraw = True
End Sub

Private Sub Command1_Click ()
   ReadCheckBox Form2.Check1    ' Llama al procedimiento del
otro módulo
   ReadCheckBox Form3.Check1    ' y envía el control como
argumento.
End Sub

' Escriba este código en Module1.
Sub ReadCheckBox (Source As Control)
   If Source.Value Then
      Source.Parent.Cls    ' Borra el formulario primario.
      Source.Parent.Print "CheckBox está activado."    ' Lo
muestra en el
                                                        '
formulario primario.
   Else
      Source.Parent.Cls    ' Borra el formulario primario.
      Source.Parent.Print "CheckBox está desactivado."    '
Lo muestra en
                                                        '    el
formulario primario.
   End If
End Sub
```

Path (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece la ruta de acceso actual. No está disponible en tiempo de diseño. Para el objeto **App**, es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**Path** [= *nombreRuta*]

La sintaxis de la propiedad **Path** consta de las siguientes partes:

Parte Descripción
objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
nombreRuta Una expresión de cadena que da como resultado el nombre de la ruta de acceso.

Comentarios
El valor de la propiedad **Path** es una cadena que indica una ruta de acceso, como C:\Ob o C:\Windows\System. Para un control **DirListBox** o **FileListBox**, el valor predeterminado es la ruta de acceso actual cuando el control se crea en tiempo de ejecución. Para el objeto **App**, **Path** especifica la ruta de acceso del archivo .vbp de proyecto cuando se ejecuta la aplicación desde el entorno de desarrollo, o la ruta de acceso del archivo .exe cuando se ejecuta la aplicación como un archivo ejecutable.
Use esta propiedad para crear capacidades de manipulación y examen de archivos de una aplicación. Establecer la propiedad **Path** tiene sobre un control un efecto similar al comando **chdir** de MS-DOS; se permiten rutas de acceso relativas con o sin especificación de unidad. Si especifica una unidad seguida de un signo de dos puntos (:) se seleccionará el directorio actual de esa unidad.
La propiedad **Path** también puede establecerse como una ruta de red cualificada sin una conexión de unidad mediante la sintaxis siguiente:
\\nombreServidor\nombreRecursoCompartido\rutaAcceso

La sintaxis anterior cambia la propiedad **Drive** a un cadena de longitud cero ("").
El cambio del valor de **Path** tiene estos efectos:

- Para un control **DirListBox** se genera un evento Change.
- Para un control **FileListBox** se genera un evento PathChange.

Nota Para **DirListBox**, el valor que devuelve **Path** es distinto del de `List(ListIndex)`, que sólo devuelve la selección.

SE APLICA A
App (Objeto)
DirListBox (Control)
FileListBox (Control)

Ejemplo de la propiedad Path

Este ejemplo muestra una lista de los archivos para la unidad y el directorio seleccionados. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga controles **DriveListBox**, **DirListBox** y **FileListBox**. Presione F5. Utilice el *mouse* (ratón) para cambiar la unidad o el directorio.

```
Private Sub Drive1_Change ()
   Dir1.Path = Drive1.Drive      ' Establece la ruta del
directorio.
```

```
End Sub

Private Sub Dir1_Change ()
   File1.Path = Dir1.Path      ' Establece la ruta del
archivo.
End Sub
```

Pattern (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica los nombres de archivo mostrados en un control **FileListBox** en tiempo de ejecución.

Sintaxis

objeto.**Pattern** [= *valor*]

La sintaxis de la propiedad **Pattern** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Una expresión de cadena que indica una especificación de archivo, como "*.*" o "*.FRM". La cadena predeterminada es "*.*", que devuelve una lista de todos los archivos. Además de usar caracteres comodín, también puede especificar varios modelos, separándolos con caracteres de punto y coma (;). Por ejemplo, "*.exe; *.bat" devolvería una lista de todos los archivos ejecutables y todos los archivos por lotes de MS-DOS.

Comentarios

La propiedad **Pattern** desempeña un papel clave en el diseño de las capacidades de examen y manipulación de archivos de una aplicación. Puede usar **Pattern** junto con otras propiedades de control de archivos para proporcionar al usuario formas de explorar archivos o grupos de archivos similares. Por ejemplo, en una aplicación dedicada a iniciar otros programas, podría hacer que sólo apareciesen los archivos .exe en el cuadro de lista de archivos (*.exe). Otras propiedades importantes para el control de archivos son **Drive**, **FileName** y **Path**.

Al modificar el valor de la propiedad **Pattern** se genera un evento PatternChange.

SE APLICA A :CONSULTAR LOS METODOS PROPIEDADES Y EVENTOS DE FileListBox (Control)

Ejemplo de la propiedad Pattern

En este ejemplo se actualiza un control **TextBox** con el nuevo patrón seleccionado en un control **FileListBox**. Los controles se configuran de manera que cuando el usuario escribe en el control **TextBox** un patrón, como *.txt, éste se refleja en el control **FileListBox**, de forma similar a la interacción que se produce en los cuadros de diálogo **Abrir** archivos de las aplicaciones para Windows. Si se especifica una ruta de acceso completa, como C:\Bin*.exe, en el control **TextBox**, el control **FileListBox** analizará

automáticamente la cadena y separará los componentes de ruta y de modelo. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga los siguientes controles: un **DirListBox**, un **FileListBox**, un **TextBox** y un **CommandButton**. Presione F5 y escriba un patrón de archivos válido en el control **TextBox**.

```
Private Sub Form_Load ()
   Command1.Default = True        ' Establece la propiedad
Default.
End Sub

Private Sub Command1_Click ()
   ' El texto se divide en sus componentes de ruta y
patrón.
   File1.Filename = Text1.Text
   Dir1.Path = File1.Path        ' Establece la ruta del
directorio.
End Sub

Private Sub File1_PatternChange ()
   Text1.Text = File1.Pattern    ' Establece como texto el
nuevo patrón.
End Sub

Private Sub Dir1_Change
   File1.Path = Dir1.Path        ' Establece la ruta en el
cuadro de lista.
End Sub
```

ReadOnly (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que determina si un control **FileListBox** contiene archivos con atributos de sólo lectura.

Sintaxis

objeto.**ReadOnly** [= *booleano*]

La sintaxis de la propiedad **ReadOnly** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica si el control muestra archivos con atributos de sólo lectura, como se describe en Valores.

Valores
Los valores admitidos para *booleano* son:

Valor	Descripción
True	El control incluye en la lista archivos de sólo lectura.
False	No aparecen en la lista del control los archivos de sólo lectura.

Comentarios
Use la propiedad **ReadOnly** con un control **FileListBox** para especificar si los archivos con atributos de sólo lectura se muestran o no en la lista de archivos.

SE APLICA A : CONSULTAR LOS METODOS PROPIEDADES Y EVENTOS DE FileListBox (Control)

Selected (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece el estado de selección de un elemento de un control **FileListBox** o **ListBox**. Esta propiedad es una matriz de valores booleanos con el mismo número de elementos que la propiedad **List**. No está disponible en tiempo de diseño.

Sintaxis
objeto.**Selected**(*índice*) [= *booleano*]
La sintaxis de la propiedad **Selected** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	El número de índice del elemento en el control.
booleano	Una expresión de tipo Boolean que especifica si el elemento está seleccionado, como se describe en Valores.

Valores
Los valores admitidos para *booleano* son:

Valor	Descripción
True	El elemento está seleccionado.
False	(Predeterminado) El elemento no está seleccionado.

Comentarios
Esta propiedad es especialmente útil cuando el usuario puede realizar selecciones múltiples. Con ella puede comprobar rápidamente qué elementos de una lista están seleccionados. También puede usarla para seleccionar o desactivar desde el código elementos de una lista.

Si la propiedad **MultiSelect** se establece a 0, puede usar la propiedad **ListIndex** para obtener el índice del elemento seleccionado. Sin embargo, en una selección múltiple, la

propiedad **ListIndex** devuelve el índice del elemento contenido en el rectángulo de enfoque, esté seleccionado o no.

Si la propiedad **Style** del control **ListBox** está establecida a 1 (casillas de verificación), la propiedad **Selected** devuelve **True** sólo para aquellos elementos cuya casilla de verificación está activada. La propiedad **Selected** no devolverá **True** para aquellos elementos que sólo estén resaltados.

SE APLICA A
FileListBox (Control)
ListBox (Control)

Picture (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un gráfico que se mostrará en un control. Para el control contenedor **OLE**, no está disponible en tiempo de diseño y es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**Picture** [= *imagen*]

La sintaxis de la propiedad **Picture** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
imagen	Una expresión de cadena que especifica un archivo que contiene un gráfico, según se indica en Valores.

Valores

Los valores admitidos para *imagen* son:

Valor	Descripción
(Ninguno)	(Predeterminado) Ninguna imagen.
(Mapa de bits, icono, metarchivo, GIF, JPEG)	Especifica un gráfico que puede cargar mediante la ventana Propiedades en tiempo de diseño. En tiempo de ejecución también puede establecer esta propiedad, utilizando la función **LoadPicture** con un mapa de bits, un icono o un metarchivo.

Comentarios

En tiempo de diseño, puede transferir un gráfico con el Portapapeles mediante los comandos **Copiar**, **Cortar** y **Pegar** del menú **Edición**. En tiempo de ejecución puede usar métodos del Portapapeles, como **GetData**, **SetData** y **GetFormat**, con las constantes del Portapapeles **vbCFBitmap**, **vbCFMetafile** y **vbCFDIB**, que aparecen en la biblioteca de objetos de Visual Basic (VB), en el Examinador de objetos.

Al establecer la propiedad **Picture** en tiempo de diseño, el gráfico se guarda y se carga con el formulario. Si crea un archivo ejecutable, la imagen estará contenida en él. Los

gráficos cargados en tiempo de ejecución no se guardan con la aplicación. Utilice la instrucción **SavePicture** para guardar un gráfico de un formulario o de un cuadro de imagen en un archivo.

Nota En tiempo de ejecución puede establecer la propiedad **Picture** al valor de la propiedad **DragIcon**, **Icon**, **Image** o **Picture** de cualquier otro objeto y también se le puede asignar un gráfico devuelto por la función **LoadPicture**.
Nota La compañía Unisys tiene una patente que pretende abarcar ciertos aspectos de la compresión GIF-LZW , que utilizan los controles **PictureBox** e **Image**. Microsoft Corporation obtuvo una licencia de la patente de Unisys LZW en Septiembre de 1996. La licencia de Microsoft, sin embargo, no se extiende a programadores de software o a terceras partes que utilicen cualquier conjunto de herramientas, lenguaje de desarrollo o productos de sistemas operativos para proporcionar la posibilidad de leer o escribir archivos GIF y/o cualquier otra característica LZW en sus propios productos (por ejemplo, mediante DLLs y APIs).

Si su aplicación comercial utiliza uno de estos controles (y, por lo tanto, la tecnología LZW), puede conseguir una opinión legal independiente acerca de los efectos de la patente o bien puede ponerse en contacto con Unisys USA en http://www.unisys.com/ para obtener más información.

SE APLICA A
CheckBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Form (Objeto), Forms (Colección)
Image (Control)
Image (Control, diseñador de informe de datos)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad Picture

En este ejemplo se cargan iconos de la biblioteca de iconos de Visual Basic en dos controles **PictureBox**. Al hacer clic en el formulario, el tercer control **PictureBox** se utiliza para intercambiar los otros dos. Puede usar dos iconos cualesquiera. Pegue el código en la sección Declaraciones de un formulario que contenga tres controles **PictureBox** (para `Picture3`, establezca **Visible** = **False**). Presione F5 para ejecutar el programa y luego haga clic en el formulario.

```
Private Sub Form_Load ()
   ' Cargar los iconos.
   Picture1.Picture =
LoadPicture("ICONOS\EQUIPO\PAPEL02A.ICO")
   Picture2.Picture =
LoadPicture("ICONOS\EQUIPO\PAPEL02B.ICO")
End Sub

Private Sub Form_Click ()
   ' Cambiar los iconos.
   Picture3.Picture = Picture1.Picture
   Picture1.Picture = Picture2.Picture
   Picture2.Picture = Picture3.Picture
   ' Borrar la tercera imagen (no es necesario si no está
visible).
   Picture3.Picture = LoadPicture()
End Sub
```

En este ejemplo se pega un mapa de bits desde el Portapapeles a un control **PictureBox**. Para averiguar el valor de las constantes de formato del Portapapeles (empezando por **vbCF**), vea la biblioteca de objetos de Visual Basic (VB) en el Examinador de objetos. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **PictureBox**. Presione F5 y en otra aplicación, copie un icono al Portapapeles. Pase a Visual Basic y haga clic en el formulario.

```
Private Sub Form_Click ()
   Picture1.Picture = Clipboard.GetData(vbCFDIB)
End Sub
```

RightToLeft (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve un valor booleano que indica el sentido de presentación del texto y controla la apariencia visual en un sistema bidireccional.

Sintaxis

objeto.**RightToLeft**

El marcador de posición *objeto* es una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Valores

Los posibles valores booleanos devueltos por la propiedad **RightToLeft** son:

Valor	Descripción
True	El control se ejecuta en una plataforma bidireccional, como Windows 95 en Árabe o en Hebreo, y el texto va de derecha a izquierda. El control debe modificar su comportamiento, como la posición de las barras de desplazamiento a la izquierda de los cuadros de texto o de lista, la colocación de las etiquetas a la derecha de los cuadros de texto, etc.
False	El control tiene que actuar como si se ejecutara en una plataforma no bidireccional, como Windows 95 en Inglés, y el texto va de izquierda a derecha. Si el contenedor no implementa esta propiedad de entorno, éste será el valor predeterminado.

SE APLICA A

AmbientProperties (Objeto)
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Data (Control)
Form (Objeto), Forms (Colección)
HScrollBar, VScrollBar (Controles)
Label (Control)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

ScaleHeight, ScaleWidth (Propiedades)

Vea también Ejemplo Se aplica a

Devuelven o establecen el número de unidades de medida horizontal (**ScaleWidth**) y vertical (**ScaleHeight**) del interior de un objeto al usar métodos gráficos o al colocar controles. Para los objetos **MDIForm**, no está disponible en tiempo de diseño y es de sólo lectura en tiempo de ejecución.

Sintaxis

objeto.**ScaleHeight** [= *valor*]
objeto.**ScaleWidth** [= *valor*]

Las sintaxis de las propiedades **ScaleHeight** y **ScaleWidth** constan de las partes siguientes:

Parte Descripción
objeto Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor Una expresión numérica que especifica la medida horizontal o vertical.

Comentarios
Con estas propiedades puede crear una escala de coordenadas personalizada para dibujar o imprimir. Por ejemplo, la instrucción `ScaleHeight = 100` cambia la unidad de medida del alto interior real del formulario. En lugar de comenzar el alto en *n* unidades (twips, píxeles, etc.), el alto será de 100 unidades definidas por el usuario. Por tanto, una distancia de 50 unidades es la mitad del ancho o alto del objeto y una distancia de 100 unidades estará situada fuera del objeto en una unidad.

Utilice la propiedad **ScaleMode** para definir una escala basada en una unidad de medida estándar, como twips, puntos, píxeles, caracteres, pulgadas, milímetros o centímetros.
Al establecer valores positivos en estas propiedades, las coordenadas se incrementan de arriba abajo y de izquierda a derecha. Si se establecen valores negativos, las coordenadas se incrementan de abajo arriba y de derecha a izquierda.

Utilizando estas propiedades y las relacionadas **ScaleLeft** y **ScaleTop**, puede configurar un sistema de coordenadas completo, con coordenadas positivas y negativas. Estas cuatro propiedades de escala se relacionan con la propiedad **ScaleMode** de la siguiente forma:

- Al establecer un valor en cualquier otra propiedad de escala, **ScaleMode** se establece automáticamente a 0. Un valor 0 de **ScaleMode** está definido por el usuario.

- Al establecer **ScaleMode** a un número mayor que 0, **ScaleHeight** y **ScaleWidth** cambian a la nueva unidad de medida, y **ScaleLeft** y **ScaleTop** se establecen a 0. Además, los valores de **CurrentX** y **CurrentY** cambian para reflejar las nuevas coordenadas del punto actual.

También puede usar el método **Scale** para establecer las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** en una única instrucción.
Nota Las propiedades **ScaleHeight** y **ScaleWidth** son distintas de las propiedades **Height** y **Width**.

Para los objetos **MDIForm**, **ScaleHeight** y **ScaleWidth** sólo se refieren al área del formulario no cubierta por controles **PictureBox**. Evite el uso de estas propiedades para cambiar el tamaño de un control **PictureBox** en el evento Resize de un objeto **MDIForm**.

SE APLICA A
Form (Objeto), Forms (Colección)
MDIForm (Objeto)
PictureBox (Control)

Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de las propiedades ScaleHeight, ScaleWidth

En este ejemplo se utilizan las propiedades **ScaleHeight** y **ScaleWidth** para cambiar las unidades de medida vertical y horizontal de un formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario y presione F5. Para ver su efecto, haga clic en el formulario, cambie su tamaño y vuelva a hacer clic en él.

```
Private Sub Form_Click ()
   Dim Radius As Integer   ' Declara una variable.
   ScaleHeight = 100   ' Establece las unidades de alto.
   ScaleWidth = 100   ' Establece las unidades de ancho.
   For Radius = 5 to 50 Step 5
      FillStyle = 1
      Circle (50, 50), Radius   ' Dibuja un círculo.
   Next Radius
End Sub
```

ScaleLeft, ScaleTop (Propiedades)

Vea también Ejemplo Se aplica a

Devuelven o establecen las coordenadas horizontal (**ScaleLeft**) y vertical (**ScaleTop**) de los bordes izquierdo y superior de un objeto al usar métodos gráficos o al situar controles.

Sintaxis

objeto.**ScaleLeft** [= *valor*]
objeto.**ScaleTop** [= *valor*]

Las sintaxis de las propiedades **ScaleLeft** y **ScaleTop** constan de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Una expresión numérica que especifica la coordenada horizontal o vertical. El valor predeterminado es 0.

Comentarios

Con estas propiedades y las relacionadas **ScaleHeight** y **ScaleWidth**, puede configurar un sistema de coordenadas completo, con coordenadas positivas y negativas. Estas cuatro propiedades de escala se relacionan con la propiedad **ScaleMode** de la siguiente forma:

- Al establecer un valor en cualquier otra propiedad de escala, **ScaleMode** se establece automáticamente a 0. Un valor 0 de **ScaleMode** está definido por el usuario.

- Al establecer **ScaleMode** a un número mayor que 0, **ScaleHeight** y **ScaleWidth** cambian a la nueva unidad de medida, y **ScaleLeft** y **ScaleTop** se establecen a 0. Los valores de **CurrentX** y **CurrentY** cambian para reflejar las nuevas coordenadas del punto actual.

También puede usar el método **Scale** para establecer las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** en una única instrucción.

Nota Las propiedades **ScaleLeft** y **ScaleTop** son distintas de las propiedades **Left** y **Top**.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de las propiedades ScaleLeft, ScaleTop

En este ejemplo se crea una cuadrícula en un control **PictureBox** y se establecen como coordenadas de la esquina superior izquierda -1, -1, en lugar de 0, 0. Cada 0,25 segundos se dibujan puntos al azar desde la esquina superior izquierda a la inferior derecha. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **PictureBox** grande y un control **Timer**, y luego presione F5.

```
Private Sub Form_Load ()
   Timer1.Interval = 250      ' Establece el intervalo de
cronómetro.
   Picture1.ScaleTop = -1      ' Establece la escala del
extremo superior.
   Picture1.ScaleLeft = -1     ' Establece la escala del
extremo izquierdo.
   Picture1.ScaleWidth = 2    ' Establece la escala (de -1 a
1).
   Picture1.ScaleHeight = 2
   Picture1.Line (-1, 0)-(1, 0)      ' Dibuja una línea
horizontal.
   Picture1.Line (0, -1)-(0, 1)      ' Dibuja una línea
vertical.
End Sub

Private Sub Timer1_Timer ()
   Dim I    ' Declara una variable.
   ' Dibuja puntos al azar dentro de un intervalo.
```

```
    For I = -1 To 1 Step .05
        Picture1.PSet (I * Rnd, I * Rnd)    ' Dibuja un punto.
    Next I
End Sub
```

ScaleMode (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica la unidad de medida de las coordenadas de un objeto al usar métodos gráficos o al situar controles.

Sintaxis

objeto.**ScaleMode** [= *valor*]

La sintaxis de la propiedad **ScaleMode** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Un número entero que especifica la unidad de medida, como se describe en Valores.

Valores

Los valores admitidos para *valor* son:

Constante	Valor	Descripción
vbUser	0	Indica que una o más de las propiedades **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop** tienen valores personalizados.
vbTwips	1	(Predeterminado) Twip (1440 twips por pulgadalógica; 567 twips por centímetro lógico).
vbPoints	2	Punto (72 puntos por pulgada lógica).
vbPixels	3	Píxel (la unidad mínima de la resolución del monitor o la impresora).
vbCharacters	4	Carácter (horizontal = 120 twips por unidad; vertical = 240 twips por unidad).
vbInches	5	Pulgada.
vbMillimeters	6	Milímetro.
vbCentimeters	7	Centímetro.
VbHimetric	8	HiMetric
VbContainerPosition	9	Unidades usadas por el contenedor del control para determinar la posición del control.
VbContainerSize	10	Unidades usadas por el contenedor del control para determinar el tamaño del control.

Comentarios

Utilizando las propiedades relacionadas **ScaleHeight**, **ScaleWidth**, **ScaleLeft** y **ScaleTop**, puede configurar un sistema de coordenadas completo, con coordenadas positivas y negativas. Estas cuatro propiedades de escala se relacionan con la propiedad

ScaleMode de la siguiente forma:

- Al establecer un valor en cualquier otra propiedad de escala, **ScaleMode** se establece automáticamente a 0. Un valor 0 de **ScaleMode** está definido por el usuario.

- Al establecer **ScaleMode** a un número mayor que 0, **ScaleHeight** y **ScaleWidth** cambian a la nueva unidad de medida, y **ScaleLeft** y **ScaleTop** se establecen a 0. Además, los valores de **CurrentX** y **CurrentY** cambian para reflejar las nuevas coordenadas del punto actual.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad ScaleMode

En este ejemplo se muestra cómo cambian el tamaño de un círculo distintos valores de la propiedad **ScaleMode**. Para probarlo, pegue el código en la sección Declaraciones de un formulario, presione F5 y haga clic en el formulario. Al hacer clic, la unidad de medida cambiará al siguiente valor de **ScaleMode** y se dibujará un círculo en el formulario.

```
Private Sub Form_Click ()
   ' Recorrer cada uno de los siete valores de ScaleMode.
   ScaleMode = ((ScaleMode + 1) Mod 7) + 1
   ' Dibuja un círculo con radio 2 en el centro del
formulario.
   Circle (ScaleWidth / 2, ScaleHeight / 2), 2
End Sub
```

TabIndex (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece el orden de tabulación de la mayoría de los objetos dentro de su formulario primario.

Sintaxis
objeto.**TabIndex** [= *índice*]
La sintaxis de la propiedad **TabIndex** consta de las siguientes partes:

Parte	**Descripción**
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
índice	Un número entero entre 0 y (*n*–1), donde *n* es el número de controles del formulario que tienen la propiedad **TabIndex**. Al asignar a **TabIndex** un valor inferior a 0 se produce un error.

Comentarios
De forma predeterminada, Visual Basic asigna un orden de tabulación a los controles cuando se van agregando a un formulario, con la excepción de los controles **Menu**, **Timer**, **Data**, **Image**, **Line** y **Shape**, que no se incluyen en el orden de tabulación. En tiempo de ejecución, los controles invisibles o deshabilitados y los controles que no pueden recibir el enfoque (controles **Frame** y **Label**) permanecen en el orden de tabulación pero se pasan por alto al tabular.

Cada control nuevo se sitúa el último en el orden de tabulación. Si modifica el valor de la propiedad **TabIndex** de un control para cambiar el orden predeterminado, Visual Basic renumerará automáticamente la propiedad **TabIndex** de los controles restantes para reflejar las inserciones y eliminaciones. Puede efectuar los cambios en tiempo de diseño mediante la ventana Propiedades o en tiempo de ejecución a través de código.
La propiedad **TabIndex** no se ve afectada por el método **ZOrder**.

Nota El orden de tabulación de un control no afecta a su tecla de acceso asociada. Si presiona la tecla de acceso de un control **Frame** o **Label**, el enfoque pasará al siguiente control del orden de tabulación que pueda recibir el enfoque.

Al cargar formularios guardados como texto ASCII, los controles con la propiedad **TabIndex** que no aparezcan en la descripción del formulario recibirán automáticamente un valor **TabIndex**. Si los valores de **TabIndex** de los controles cargados posteriormente entran en conflicto con valores ya asignados, dichos controles recibirán automáticamente valores distintos.

Cuando elimine uno o más controles puede usar el comando **Deshacer** para restaurar los controles y todas sus propiedades excepto **TabIndex**, que no se puede restaurar. **TabIndex** se restablece al final del orden de tabulación cuando se usa **Deshacer**.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO

CoolBar (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Label (Control)
ListBox (Control)
MSChart (Control)
MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RichTextBox (Control)
TextBox (Control)

Ejemplo de la propiedad TabIndex

En este ejemplo se invierte el orden de tabulación de un grupo de botones modificando la propiedad **TabIndex** de una matriz de botones de comando. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga cuatro controles **CommandButton**. Establezca **CommandX** en la propiedad **Name** de cada botón para crear la matriz de controles, y luego presione F5 y haga clic en el formulario para invertir su orden de tabulación.

```
Private Sub Form_Click ()
   Dim I, X   ' Declarar variables.
   ' Invertir el orden de tabulación estableciendo el valor
inicial de X.
   If CommandX(0).TabIndex = 0 Then X = 4 Else X = 1
      For I = 0 To 3
         CommandX(I).Caption = X   ' Establecer el título.
         CommandX(I).TabIndex = X - 1     ' Establecer el
orden de                                                  '
tabulación.
         If CommandX(0).TabIndex = 3 Then
            X = X - 1   ' Disminuir X.
         Else
            X = X + 1   ' Incrementar X.
         End If
      Next I
End Sub
```

TabStop (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica si el usuario puede usar la tecla TAB para llevar el enfoque a un objeto.

Sintaxis

objeto.**TabStop** [= *booleano*]

La sintaxis de la propiedad **TabStop** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión de tipo Boolean que especifica si el objeto es un tabulador, como se describe en Valores.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado) Designa el objeto como un tabulador.
False	Hace que se salte el objeto en la tabulación, aunque el objeto mantiene su lugar en el orden de tabulación real, determinado en la propiedad **TabIndex**.

Comentarios

Esta propiedad permite agregar o quitar un control del orden de tabulación de un formulario. Por ejemplo, si utiliza un control **PictureBox** para dibujar un gráfico, establezca a **False** su propiedad **TabStop** para que el usuario no pueda situarse en el control **PictureBox**.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MSChart (Control)

MSFlexGrid (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RichTextBox (Control)
TextBox (Control)

Tag (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece una expresión que almacena cualquier dato adicional que necesite el programa. Al contrario de lo que sucede con otras propiedades, Visual Basic no utiliza el valor de **Tag**. Puede usar esta propiedad para identificar objetos.

Sintaxis

objeto.**Tag** [= *expresión*]

La sintaxis de la propiedad **Tag** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
expresión	Una expresión de cadena que identifica el objeto. El valor predeterminado es una cadena de longitud cero (""").

Comentarios

Puede usar esta propiedad para asignar una cadena de identificación a un objeto sin afectar al resto de sus propiedades ni causar efectos secundarios. La propiedad **Tag** es útil cuando necesita comprobar la identidad de un control u objeto **MDIForm** que se pasa como variable a un procedimiento.

Sugerencia Cuando cree una nueva instancia de un formulario, asigne un valor único a la propiedad **Tag**.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)

HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
Line (Control)
ListBox (Control)
MDIForm (Objeto)
MSChart (Control)
Menu (Control)
OLEObject (Objeto)
OptionButton (Control)
PictureBox (Control)
PropertyPage (Objeto)
RemoteData (Control)
Shape (Control)
TextBox (Control)
Timer (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad Tag

En este ejemplo se muestra un icono único para cada control que se arrastra. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga tres controles **PictureBox**. Establezca a 1 la propiedad **DragMode** de Picture1 y Picture2, y luego presione F5. Utilice el *mouse* para arrastrar Picture1 o Picture2 sobre Picture3.

```
Private Sub Form_Load ()
    Picture1.Tag = "ICONOS\FLECHAS\PUNTO03.ICO"
    Picture2.Tag = "ICONOS\FLECHAS\PUNTO04.ICO"
End Sub

Private Sub Picture3_DragOver (Source As Control, X As
Single, Y As Single, State As Integer)
   If State = vbEnter Then
        ' Seleccionar basándose en la propiedad Name de
cada PictureBox.
      Select Case Source.Name
      Case "Picture1"
         ' Cargar el icono para Picture1.
         Source.DragIcon = LoadPicture(Picture1.Tag)
      Case "Picture2"
         ' Cargar el icono para Picture2.
         Source.DragIcon = LoadPicture(Picture2.Tag)
      End Select
   ElseIf State = vbLeave Then
```

```
        ' Cuando el origen no está sobre Picture3,
descargar el icono.
      Source.DragIcon = LoadPicture()
   End If
End Sub
```

ToolTipText (Propiedad)

Vea también Ejemplo Se aplica a
Devuelve o establece Información sobre herramientas.

Sintaxis
objeto.**ToolTipText** [= *cadena*]
La sintaxis de la propiedad **ToolTipText** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
cadena	Una cadena asociada con un objeto de la lista Se aplica a que aparece durante un segundo aproximadamente en un pequeño rectángulo debajo del objeto cuando el cursor del usuario pasa por encima del objeto en tiempo de ejecución.

Comentarios
Si usa únicamente una imagen para identificar un objeto, puede usar esta propiedad para explicar con pocas palabras cada uno de esos objetos.
En tiempo de diseño puede establecer la cadena de la propiedad **ToolTipText** en el cuadro de diálogo de propiedades del control.
En los controles **Toolbar** y **TabStrip** debe establecer la propiedad **ShowTips** a **True** para mostrar Información sobre herramientas.

SE APLICA A
CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
Image (Control)
Label (Control)

ListBox (Control)
MSChart (Control)
MSFlexGrid (Control)
Masked Edit (Control)
Multimedia MCI (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)

TopIndex (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que especifica el elemento de un control **ComboBox**, **DirListBox**, **DriveListBox**, **FileListBox** o **ListBox** que se muestra en el extremo superior del control. No está disponible en tiempo de diseño.

Sintaxis

objeto.**TopIndex** [= *valor*]

La sintaxis de la propiedad **TopIndex** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	El número del elemento de la lista que se muestra en el extremo superior. El valor predeterminado es 0, correspondiente al primer elemento de la lista.

Comentarios

Puede usar esta propiedad para desplazarse por un control sin seleccionar ningún elemento.

Si el valor de la propiedad **Columns** del control **ListBox** es 0, el elemento aparecerá en el extremo superior si hay bajo él suficientes elementos como para llenar la parte visible de la lista.

Si el valor de la propiedad **Columns** del control **ListBox** es mayor que 0, la columna del elemento pasará a la posición situada más a la izquierda, sin cambiar su posición dentro de la columna.

SE APLICA A

ComboBox (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
ListBox (Control)

Ejemplo de la propiedad TopIndex

En este ejemplo se llena un control **ListBox** con los nombres de las fuentes de pantalla y luego se produce un desplazamiento por la lista al hacer clic en el formulario. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **ListBox**, presione F5 y haga clic en el formulario.

```
Private Sub Form_Load ()
   Dim I   ' Declara una variable.
   For I = 0 To Screen.FontCount -1    ' Llena el cuadro de
lista con
      List1.AddItem Screen.Fonts(I)    ' los nombres de las
fuentes.
   Next I
End Sub

Private Sub Form_Click ()
   Dim X   ' Declara una variable.
   X = List1.TopIndex   ' Obtiene el índice actual.
   List1.TopIndex = List1.TopIndex + 5    ' Restablece el
primer elemento.
   If List1.TopIndex = X Then List1.TopIndex = 0
End Sub
```

Value (Propiedad)

Vea también Ejemplo Se aplica a

- Controles **CheckBox** y **OptionButton**: devuelve o establece el estado del control.
- Control **CommandButton**: devuelve o establece un valor que indica si se ha elegido el botón. No está disponible en tiempo de diseño.
- Objeto **Field**: devuelve o establece el contenido de un campo. No está disponible en tiempo de diseño.
- Controles **HScrollBar** y **VScrollBar** (barras de desplazamiento horizontal y vertical): devuelve o establece la posición actual de la barra de desplazamiento, cuyo valor devuelto se encuentra siempre entre los valores de las propiedades **Max** y **Min**, inclusive.

Sintaxis

objeto.**Value** [= *valor*]

La sintaxis de la propiedad **Value** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
valor	Un valor que especifica el estado, contenido o posición de un control, según se indica en Valores.

Valores

Los valores admitidos para *valor* son:

- Control **CheckBox**: 0 es desactivado (predeterminado), 1 es activado y 2 es atenuado.

- Control **CommandButton**: **True** indica que se ha elegido el botón y **False** (predeterminado) indica que no se ha elegido. Al establecer a **True** la propiedad **Value** se invoca el evento Click del botón.

- Objeto **Field**: restringido solamente por el tipo de datos del campo.

- Controles **HScrollBar** y **VScrollBar**: establece valores entre –32.768 y 32.767 para situar el cuadro de desplazamiento.

- Control **OptionButton**: **True** indica que el botón está seleccionado y **False** (predeterminado) que no lo está.

Comentarios

Se supone la propiedad predeterminada de un objeto y no es necesario que se especifique en el código. Por ejemplo, **Field** es la propiedad predeterminada de los objetos **Recordset** y **Value** es la propiedad predeterminada de los objetos **Field**. Ello hace que las dos instrucciones siguientes sean equivalentes:

```
Dn.Fields("IDPub").Value = X
Dn.("IDPub") = X
```

La primera instrucción *especifica* las propiedades predeterminadas, mientras que la segunda las asume.

SE APLICA A

AsyncProperty (Objeto)
CheckBox (Control)
CommandButton (Control)
EventParameter (Objeto)
HScrollBar, VScrollBar (Controles)
OptionButton (Control)

Ejemplo de la propiedad Value

En este ejemplo se muestra el valor numérico de un control **HScrollBar** (barra de desplazamiento horizontal) en un control **TextBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga un control **TextBox** y un control **HScrollBar**. Presione F5 para ejecutar el programa y haga clic en la barra de desplazamiento.

```
Private Sub Form_Load ()
   HScroll1.Min  =  0        '  Inicializar  la  barra  de
desplazamiento.
   HScroll1.Max = 1000
   HScroll1.LargeChange = 100
   HScroll1.SmallChange = 1
End Sub

Private Sub HScroll1_Change ()
   Text1.Text = Format (HScroll1.Value)
End Sub
```

Value (Propiedad, controles ActiveX)

Vea también Ejemplo Se aplica a

Establece o devuelve el valor de un objeto.

Sintaxis

objeto.**Value** [= *entero*]

La sintaxis de la propiedad **Value** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
entero	Para un control **Slider**, un entero largo que especifica la posición actual del control deslizante. Para el control **ProgressBar**, un entero que especifica el valor del control **ProgressBar**. Para los demás controles, vea Valores.

Valores

Para el objeto **Button**, los valores de *entero* son:

Constante	Valor	Descripción
tbrUnpressed	0	(Predeterminado). El botón no se está presionando ni se está verificando en el momento.
tbrPressed	1	El botón se está presionando o se está verificando ahora.

Comentarios

- Control **Slider**: establece o devuelve la posición actual del control deslizante. Para un control **Slider**, **Value** tiene siempre un valor comprendido entre los valores de las propiedades **Max** y **Min**, inclusive.

- Control **ProgressBar**: establece o devuelve un valor que indica la evolución aproximada de una operación hacia su terminación. Incrementar la propiedad **Value** no modifica la apariencia del control **ProgressBar** en el valor exacto de la propiedad **Value**. **Value** tiene siempre un valor que está entre los valores de las propiedades **Max** y **Min**, inclusive. No está disponible en tiempo de diseño.

Ejemplo de la propiedad Value (Controles ActiveX)

Este ejemplo usa la propiedad **Value** para determinar qué icono de un control **ImageList** asociado se va a presentar en el control **Toolbar**. Para probar el ejemplo, coloque un control **Toolbar** en un formulario y pegue el código en la sección Declaraciones del formulario. Luego, ejecute el ejemplo.

```
Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
   ' Usa el valor Key para determinar en qué botón se ha
hecho clic.
   Select Case Button.Key

   Case "Realizado"   ' Botón de comprobación.
      If Button.Value = vbUnchecked Then
         ' El botón no está verificado.
         Button.Value = vbChecked   ' Comprueba el botón.
         ' Suponiendo que hay un objeto ListImage con
         ' la clave "abajo".
         Button.Image = "abajo"
      Else   ' Deshace la comprobación del botón.
         Button.Value = vbUnchecked
         ' Suponiendo que hay un objeto ListImage con
         ' la clave "arriba".
         Button.Image = "arriba"
      End If

   ' Hay más casos posibles.
   End Select
End Sub
```

Visible (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un valor que indica si un objeto es visible o está oculto.

Sintaxis

objeto.**Visible** [= *booleano*]

La sintaxis de la propiedad **Visible** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
booleano	Una expresión booleana que especifica si el objeto es visible o si está oculto.

Valores

Los valores admitidos para *booleano* son:

Valor	Descripción
True	(Predeterminado) El objeto es visible.
False	El objeto está oculto.

Comentarios

Para ocultar un objeto al inicio, establezca la propiedad **Visible** a **False** en tiempo de diseño. Si establece esta propiedad en el código puede ocultar y volver a mostrar posteriormente un control en tiempo de ejecución como respuesta a un evento determinado.

Nota El uso del método **Show** o **Hide** en un formulario equivale a establecer **True** o **False**, respectivamente, en su propiedad **Visible** a través del código.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
Control de datos ADO
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
Extender (Objeto)
FileListBox (Control)
Form (Objeto), Forms (Colección)

Frame (Control)
Function (Control, diseñador de informe de datos)
HScrollBar, VScrollBar (Controles)
Image (Control)
Image (Control, diseñador de informe de datos)
Label (Control)
Label (Control, diseñador de informe de datos)
Line (Control)
Line (Control, diseñador de informe de datos)
ListBox (Control)
MDIForm (Objeto)
Menu (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
Section (Objeto, diseñador de informe de datos)
Shape (Control)
Shape (Control, diseñador de informe de datos)
TextBox (Control)
TextBox (Control, diseñador de informe de datos)

Ejemplo de la propiedad Visible

En este ejemplo se crea una animación mediante dos controles **PictureBox**. Para probar este ejemplo, pegue el código en la sección Declaraciones de un formulario que contenga dos controles **PictureBox** de tamaño de icono. Establezca a **FileCab** la propiedad **Name** de los dos controles **PictureBox** para crear una matriz, presione F5 y haga clic en la imagen para ver la animación.

```
Private Sub Form_Load ()
   Dim I    ' Declara la variable.
   FileCab(0).BorderStyle = 0    ' Establecer BorderStyle.
   FileCab(1).BorderStyle = 0
   ' Cargar los iconos en los cuadros de imagen.
   FileCab(1).Picture                                    =
LoadPicture("ICONOS\OFFICE\ARCH03B.ICO")
   FileCab(0).Picture                                    =
LoadPicture("ICONOS\OFFICE\ARCH03A.ICO")
   For I = 0 To 1
      FileCab(I).Move 400, 400    ' Colocar los gráficos en
el mismo                                          ' punto.
   Next I
   FileCab(1).Visible  =  False            '  Establecer  a
invisible.
   FileCab(0).Visible = True       ' Establecer a visible.
End Sub
```

```
Private Sub FileCab_Click (Index As Integer)
   Dim I    ' Declarar variable.
   For I = 0 To 1
      ' Cambiar la visibilidad para los dos gráficos.
      FileCab(I).Visible = Not FileCab(I).Visible
   Next I
End Sub
```

WhatsThisHelpID (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve o establece un número de contexto asociado a un objeto. Se utiliza para dotar a las aplicaciones de Ayuda interactiva con el menú emergente ¿Qué es esto? de la Ayuda de Windows 95.

Sintaxis

objeto.**WhatsThisHelpID** [= *número*]

La sintaxis de la propiedad **WhatsThisHelpID** consta de las siguientes partes:

Parte	Descripción
objeto	Una expresión de objeto que da como resultado un objeto de la lista Se aplica a.
número	Una expresión numérica que especifica un número de contexto de Ayuda, según se describe en Valores.

Valores

Los valores admitidos para *número* son:

Valor	Descripción
0	(Predeterminado) No se especifica ningún número de contexto.
>0	Un entero que especifica el número de contexto válido para el tema ¿Qué es esto? asociado al objeto.

Comentarios

Windows 95 usa el botón ¿Qué es esto? situado en la esquina superior derecha de la ventana para iniciar la Ayuda de Windows y cargar el tema identificado por la propiedad **WhatsThisHelpID**.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)

Contenedor OLE (Control)
Control de datos ADO
CoolBar (Control)
Data (Control)
DataRepeater (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Frame (Control)
HScrollBar, VScrollBar (Controles)
Image (Control)
Label (Control)
ListBox (Control)
MSChart (Control)
MSHFlexGrid (Control)
Masked Edit (Control)
OptionButton (Control)
PictureBox (Control)
RemoteData (Control)
RichTextBox (Control)
TextBox (Control)

hDC (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve un controlador proporcionado por el entorno operativo Microsoft Windows para el contexto de dispositivo de un objeto.

Sintaxis

objeto.**hDC**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

Esta propiedad es un controlador de contexto de dispositivo del entorno operativo Windows. El entorno operativo Windows administra la presentación del sistema asignando un contexto de dispositivo al objeto **Printer** y a cada formulario y control **PictureBox** de la aplicación. Puede usar la propiedad **hDC** para referirse al controlador del contexto de dispositivo de un objeto. Así se obtiene un valor que se pasa en las llamadas a la API de Windows.

En un control **CommonDialog**, esta propiedad devuelve un contexto de dispositivo para la impresora seleccionada en el cuadro de diálogo **Imprimir** cuando el indicador **cdlReturnDC** está activado o un contexto de información cuando el indicador **cdlReturnIC** está activado.

Nota El valor de la propiedad **hDC** puede cambiar mientras se ejecuta un programa, por lo que no debe almacenar el valor en una variable. En su lugar, use la propiedad **hDC** cada vez que lo necesite.

La propiedad **AutoRedraw** puede hacer que el valor de la propiedad **hDC** cambie. Si **AutoRedraw** se establece a **True** en un formulario o en un contenedor **PictureBox**, **hDC** actúa como controlador de contexto de dispositivo del gráfico persistente (equivalente a la propiedad **Image**). Cuando **AutoRedraw** es **False**, **hDC** es el valor real de **hDC** de la ventana Formulario o el contenedor **PictureBox**. El valor de la propiedad **hDC** puede cambiar durante la ejecución del programa independientemente del valor de **AutoRedraw**.

Si la propiedad **HasDC** se establece a **False**, el sistema creará un nuevo contexto de dispositivo y el valor de la propiedad **hDC** cambiará cada vez que se le llame.

SE APLICA A
Form (Objeto), Forms (Colección)
PictureBox (Control)
Printer (Objeto, colección Printers)
PropertyPage (Objeto)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad hDC

Este ejemplo dibuja un triángulo y después usa una función de Microsoft Windows para llenarlo con un color. Para probar este ejemplo, cree un nuevo módulo mediante el comando **Agregar módulo** del menú **Proyecto**. Pegue la instrucción **Declare** en la sección Declaraciones del módulo nuevo, asegurándose de que la instrucción se encuentra en una única línea. Después pegue el procedimiento **Sub** en la sección Declaraciones de un formulario. Presione F5 y haga clic en el formulario.

```
' Declaración de una rutina de Windows. Esta instrucción es
para el
' módulo.
Declare Function FloodFill Lib "GDI32" Alias "FloodFill"
 (ByVal hDC As Long, ByVal X As Long, ByVal Y As Long,
ByVal Color As Long) As Long
' Posiciona el siguiente código en el formulario.
Private Sub Form_Click ()
   ScaleMode = vbPixels        ' Windows dibuja en píxeles.
   ForeColor = vbBlack        ' Dibuja líneas negras.
   Line (100, 50)-(300, 50) ' Dibuja un triángulo.
   Line -(200, 200)
   Line -(100, 50)
```

```
   FillStyle = vbFSSolid          ' Establece el valor de
FillStyle a sólido.
   FillColor = RGB(128, 128, 255)          ' Establece
FillColor.
   ' Llama a la API de Windows para llenarlo.
   FloodFill hDC, 200, 100, ForeColor
End Sub
```

hWnd (Propiedad)

Vea también Ejemplo Se aplica a

Devuelve el controlador de un formulario o un control.

Nota Esta propiedad no es compatible con el control contenedor **OLE**.

Sintaxis

objeto.**hWnd**

El marcador de posición *objeto* representa una expresión de objeto que da como resultado un objeto de la lista Se aplica a.

Comentarios

El entorno operativo Microsoft Windows identifica cada formulario y control de una aplicación asignándole un controlador, o **hWnd**. La propiedad **hWnd** se usa en las llamadas a la API de Windows. Muchas funciones del entorno operativo Windows requieren el **hWnd** de la ventana activa como argumento.

Nota Como el valor de esta propiedad puede cambiar durante la ejecución del programa, no almacene el valor de **hWnd** en una variable.

Si la propiedad **Windowless** de **User Control** se establece a **True**, la propiedad **hWnd** devolverá 0.

SE APLICA A

CheckBox (Control)
ComboBox (Control)
CommandButton (Control)
Contenedor OLE (Control)
DirListBox (Control)
DriveListBox (Control)
FileListBox (Control)
Form (Objeto), Forms (Colección)
Frame (Control)
HScrollBar, VScrollBar (Controles)
ListBox (Control)
MDIForm (Objeto)
OptionButton (Control)

PictureBox (Control)
PictureClip (Control)
PropertyPage (Objeto)
RichTextBox (Control)
TextBox (Control)
UserControl (Objeto)
UserDocument (Objeto)

Ejemplo de la propiedad hWnd

Este ejemplo fuerza que un formulario permanezca siempre visible. Para probar este ejemplo cree un formulario (no un formulario MDI secundario), cree un menú para este formulario llamado Principal. Inserte en él un submenú llamado **Siempre visible** y establezca **Name** a mnuTopmost. Cree un nuevo módulo mediante el comando **Agregar módulo** del menú **Proyecto**. Pegue la instrucción **Declare** en la sección Declaraciones del módulo nuevo, asegurándose de que se encuentra en una única línea. Después pegue el procedimiento **Sub** en la sección Declaraciones del formulario y presione F5.

```
' Declaración de una rutina de Windows.
' Debe poner esta instrucción en el módulo.
Declare Function SetWindowPos Lib "user32" Alias
"SetWindowPos" (ByVal hwnd As Long, ByVal hWndInsertAfter
As Long, ByVal x As Long, ByVal y As Long, ByVal cx As
Long, ByVal cy As Long, ByVal wFlags As Long) As Long

' Establece algunos valores de constantes (de
WIN32API.TXT).
Const conHwndTopmost = -1
Const conHwndNoTopmost = -2
Const conSwpNoActivate = &H10
Const conSwpShowWindow = &H40

Private Sub mnuTopmost_Click ()
   ' Agrega o quita la marca de verificación del menú.
   mnuTopmost.Checked = Not mnuTopmost.Checked
   If mnuTopmost.Checked Then
      ' Activa el atributo TopMost.
      SetWindowPos hWnd, conHwndTopmost, 0, 0, 0, 0,
conSwpNoActivate Or conSwpShowWindow
   Else
      ' Desactiva el atributo TopMost.
      SetWindowPos hWnd, conHwndNoTopmost, 0, 0, 0, 0,
conSwpNoActivate Or conSwpShowWindow
   End If
End Sub
```

Este ejemplo despliega automáticamente la parte de lista de un control **ComboBox** cuando el **ComboBox** recibe el enfoque. Para probar este ejemplo, cree un formulario nuevo que contenga un control **ComboBox** y un control **OptionButton** (usado solamente para recibir el enfoque). Cree un módulo nuevo mediante el comando **Agregar módulo** del menú **Proyecto**. Pegue la instrucción **Declare** en la sección Declaraciones del módulo nuevo, asegurándose de que se encuentra en una única línea. Después pegue el procedimiento **Sub** en la sección Declaraciones del formulario y presione F5. Use la tecla TAB para mover el enfoque hacia y desde el **ComboBox**.

```
Declare Function SendMessage Lib "user32" Alias
"SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long,
ByVal wParam As Long, lParam As Long) As Long

Private Sub Combo1_GotFocus ()
   Const CB_SHOWDROPDOWN = &H14F
   Dim Tmp
   Tmp = SendMessage(Combo1.hWnd, CB_SHOWDROPDOWN, 1, ByVal
0&)
End Sub
```

REFERENCIAS

REFERENCIAS DE PROPIEDADES, METODOS, EVENTOS Y CONTROLES DE LA LIBRERÍA DE VISUAL BASIC.

Archivo ejecutable

Una aplicación para Windows que se puede ejecutar fuera del entorno de programación. Un archivo ejecutable tiene una extensión .EXE.

API de Windows

La API (Interfaz de programación de aplicaciones) de Windows consta de las funciones, mensajes, estructuras de datos, tipos de datos e instrucciones que puede usar para crear aplicaciones que se ejecuten bajo Microsoft Windows. Las partes de la API que más se utilizan son los elementos de código incluidos para llamar a funciones de la API desde Windows. Estos incluyen declaraciones de procedimientos (para las funciones de Windows), definiciones de tipos definidos por el usuario (para estructuras de datos pasadas a esas funciones) y declaraciones de constantes (para valores pasados y retornados desde esas funciones).

Aplicación

Una colección de código y elementos visuales que funcionan conjuntamente como un único programa. Los programadores pueden generar y ejecutar aplicaciones dentro del entorno de programación, mientras que los usuarios normalmente ejecutan las aplicaciones como archivos ejecutables fuera del entorno de programación.

Comando Pegar vínculo

Un comando que inserta el contenido del Portapapeles copiado desde una aplicación basada en Windows a otra. Este comando crea un vínculo entre las aplicaciones de manera que los datos copiados se actualizan automáticamente.

Contexto de dispositivo

Un vínculo entre una aplicación para Windows, un controlador de dispositivo y un dispositivo de salida como una pantalla, una impresora o un trazador gráfico.

Controlador

Valor entero único definido por el entorno operativo y utilizado por el programa para identificar un objeto y cambiar a él, por ejemplo un control o un formulario.

Cuadro de lista

Un control (**ListBox**) que muestra una lista de elecciones.

Destino

En el vínculo activo de un intercambio de datos dinámicos (DDE), la aplicación, el formulario o el control que solicita datos de la aplicación de origen.

Evento
Una acción reconocida por un objeto, como hacer clic con el *mouse* o presionar una tecla, para la que se puede escribir código de respuesta. Los eventos pueden producirse como resultado de una acción del usuario o del código, o pueden estar desencadenados por el sistema.

Eventos en cascada
Una secuencia de eventos causada por un procedimiento de evento que se llama a sí mismo directa o indirectamente; también se llama cascada de eventos o recursividad. Los procedimientos de eventos en cascada normalmente producen errores en tiempo de ejecución, como un desbordamiento de pila.

Formulario primario
Un formulario que contiene controles.

Formulario MDI
Una ventana que compone el segundo plano de una aplicación con interfaz de múltiples documentos (MDI). El formulario MDI es el contenedor de cualquier formulario secundario MDI de la aplicación.

Gráfico persistente
El resultado de un método gráfico que se almacena en memoria. Los gráficos persistentes se conservan automáticamente cuando se producen ciertos tipos de eventos de pantalla, por ejemplo cuando se vuelve a mostrar un formulario tras haberse ocultado detrás de otra ventana. Los gráficos son persistentes si se dibujan cuando la propiedad **AutoRedraw** se ha establecido a **True**.

Indicador
Una variable que se usa para hacer un seguimiento de una condición en una aplicación. Un indicador se puede establecer mediante una constante o una combinación de constantes.

Indice
Un número que identifica a un elemento en una matriz, en una matriz de controles o en una colección.

Información sobre herramientas
Texto que describe la función de un botón de la barra de herramientas o cualquier otra herramienta. La información sobre herramientas aparece al dejar el puntero del *mouse* (ratón) sobre un objeto.

Instancia

Cualquiera de un conjunto de objetos que comparten la misma clase. Por ejemplo, múltiples instancias de una clase **Form** comparten el mismo código y se cargan con los mismos controles con los que se diseñó la clase **Form**. En tiempo de ejecución, las propiedades individuales de los controles de cada instancia se pueden establecer a valores diferentes.

Mapa de bits

Una imagen representada por píxeles y almacenada como una colección de bits en la que uno o más bits de información de color corresponde a cada píxel. Los mapas de bits tienen normalmente un nombre de archivo con extensión .BMP.

Máscara de bits

Un valor utilizado con operadores de bit (**And**, **Eqv**, **Imp**, **Not**, **Or**, **Xor**) para comprobar, establecer o restablecer el estado de los bits individuales en un valor de campo de bits.

Metarchivo

Un archivo que almacena una imagen como objetos gráficos (líneas, círculos, polígonos) en lugar de píxeles. Existen dos tipos de metarchivos, estándar y mejorado. Los metarchivos estándar normalmente tienen una extensión .WMF. Los metarchivos mejorados suelen tener una extensión .EMF. Los metarchivos conservan una imagen más precisa que los píxeles cuando la imagen cambia de tamaño.

Método gráfico

Un método que opera sobre un objeto como un **Form**, **PictureBox** o **Printer**, y efectúa operaciones de dibujo en tiempo de ejecución como una animación o simulación. Los métodos gráficos son **Circle**, **Cls**, **Line**, **PaintPicture**, **Point**, **Print** y **PSet**.

Objeto vinculado

Un objeto que se crea en otra aplicación y se vincula a una aplicación de Visual Basic. Cuando vincula un objeto, inserta un marcador de posición en la aplicación. A diferencia de un objeto incrustado, los datos de un objeto vinculado se almacenan realmente en la aplicación que lo creó y están administrados por dicha aplicación.

Objeto incrustado

Un objeto que se crea en otra aplicación y después se incrusta en una aplicación de Visual Basic. Cuando crea un objeto incrustado, todos los datos asociados con el objeto se copian y están contenidos en la aplicación de Visual Basic.

Origen

La aplicación, formulario o control que envía información y comandos cuando dos o más programas que admiten Intercambio dinámico de datos (DDE) se están ejecutando simultáneamente.

Orden de tabulación
El orden en que se desplaza el enfoque entre un campo o un objeto al siguiente a medida que presiona la tecla TAB o Mayús-Tab.

Píxel
Abreviatura de 'elemento de imagen'; un punto que representa la menor unidad gráfica de medida de una pantalla. Un píxel depende de la pantalla; es decir, las dimensiones de los elementos de la pantalla varían con la pantalla y la resolución.

Poke
En Intercambio de datos dinámico (DDE), enviar datos desde una aplicación de destino a una aplicación de origen. La propiedad **LinkMode** de un formulario debe establecerse a **Source** (origen) para permitir que una aplicación de destino envíe datos a los controles del formulario.

Punto
En tipografía, un punto es 1/72 de pulgada. El tamaño de las fuentes se expresa habitualmente en puntos.

Sintaxis
El orden y puntuación prescritos para poner los elementos del lenguaje de programación en instrucciones que son significativas para Visual Basic.

Submenú
Un menú abierto desde otro menú que muestra su propia lista de comandos.

Tecla de acceso
Una tecla presionada mientras se mantiene presionada la tecla ALT que permite al usuario abrir un menú, llevar a cabo un comando, seleccionar un objeto o desplazarse a un objeto. Por ejemplo, Alt-A abre el menú **Archivo**.

Tecla de edición
Las teclas INS, SUPR o RETROCESO.

Tecla de función
Cualquiera de las teclas etiquetadas F1 a F12. Las teclas de función normalmente proporcionan teclas de método abreviado para comandos y acciones que se realizan con frecuencia. Puede asignar una tecla de función como una tecla de método abreviado.

Twip
Una unidad independiente de la pantalla utilizada para asegurar que la colocación y la proporción de los elementos de la pantalla de la aplicación son los mismos en todos los sistemas de pantallas. Un twip es una unidad de medida de la pantalla igual a 1/20 de un punto de impresora. Existen aproximadamente 1440 twips en una pulgada lógica o 567

twips en un centímetro lógico (la longitud de un elemento de pantalla que mide una pulgada o un centímetro al imprimirse).

Ventana activa

La ventana que aparece en primer plano con una barra de título o un borde resaltado para distinguirla de otras ventanas visibles.

Ventana secundaria MDI

Un formulario contenido dentro de un formulario MDI en una aplicación con interfaz de múltiples documentos (MDI). Para crear un formulario secundario, establezca su propiedad **MDIChild** a **True**.

CRYSTAL REPORTS

Crystal Reports Paso a Paso

Cómo asociar un generador de informes corriente a su base de datos D3

Por Duane Gilger

Con más de un millón de usuarios en todo el mundo, Crystal Reports de Seagate Software Information Management Group se ha convertido en un producto a tener en cuenta en el mercado de los generadores de informes en entorno Windows. Por otro lado, software como Microsoft Access Crystal Reports como motor de sus funciones de generador de informes, lo cual da buena cuenta de la utilidad del producto.
Ya se ha tratado de la utilización de funciones de edición de informes de Access en la adaptación de los informes generados con los productos Pick. A continuación le mostraremos cómo asociar Crystal Report 5.0 y D3 ODBC sacando así el máximo partido de los datos Pick, dándoles el aspecto más atractivo al producir informes en Windows. El siguiente ejemplo le indicará los pasos que deberá usted seguir para crear un *file listing* a partir de una base de datos Pick.

Nos inspiramos aquí en el trabajo realizado por uno de nuestros VARs Tower Systems Inc. situada en Tustin, California. Tower es responsable del soporte y desarrollo de tres mercados verticales, a saber: la distribución, el sector médico y las bases de datos a partir de un sistema Advanced Pick/ Pro. Esta compañía comercializa igualmente un programa capaz de recopilar la información de las bases para generar a continuación los informes necesarios para las actividades de las diversas agencias gubernamentales responsables de la financiación de centros de tratamiento de drogodependencia en el estado de California. Este programa denominado DRS recurre a Crystal Reports para generar los informes. El primer contacto con ODBC puede resultar un tanto desafiante, así que para evitar cualquier problema le invitamos a seguir paso a paso las etapas precisas que a continuación le indicamos.

- 1. Declare un servidor ODBC haciendo clic dos veces sobre el icono *ODBC* en el *Panel de configuración* de Windows NT.(figura 1).

- 2. Ahí puede encontrar la referencia de un **Servidor ODBC por defecto** instalado en el momento de la instalación de D3.(figura 2).

- 3. Debe incorporar un servidor capaz de establecer una conexión con los datos Pick. Para mostrar cómo generar INFORMES serán necesarios tres ficheros: **admitrep, clinic** y **pdr.** Estos ficheros se encuentran en el directorio **DRS.** Cree un servidor que corresponda a estos ficheros haciendo clic en **Insertar** en la ventana superior. Aparecerá entonces una ventana titulada *Crear una nueva fuente de datos.* (figura 3).

- 4. Seleccione el **piloto D3 ODBC** y haga clic a continuación sobre **Terminar**. La (figura 4) ilustra un ejemplo de parámetros de configuración de la fuente de datos D3.

- 5. Ojo: **Nombre de la fuente de datos Data Source Name** designa la base de datos Pick que contiene sus ficheros de datos. **Description Text** puede recibir la información que usted desee. **Host and Virtual Machine** corresponde al nombre NT de su aparato. **Port Number** debe conservar por defecto el valor declarado en el momento de la instalación de D3. El balance no precisa de mayores explicaciones.

- 6. Haga dos veces clic sobre el icono **Servicios** del *Panel de configuración* (figura 5).

- 7. Compruebe que se ha puesto en marcha correctamente el **Servidor D3 ODBC** (figura 6).

Para preparar una tabla SQL que corresponda a su fichero tenga en cuenta en su sesión Telnet el texto que sigue.

```
To DRS (connects to database containing your report files)
Sql-create-table admitrep
```

El estado de California utiliza por ejemplo un INFORME que presenta una contabilidad del conjunto de los pacientes ingresados en el curso de un mes. El fichero **admitrep** es por tanto una tabla que se puede crear al seleccionar el conjunto de los pacientes hospitalizados durante un mes. Ejecute Crystal Reports y seleccione **Nuevo informe.** Seleccione entonces **Drill Down exploration** en el menú Galería de informes (figura 7).

En Create Reports Expert, seleccione **SQL/ODBC** (figura 8) y a continuación seleccione el servidor ODBC que ya había creado para los ficheros (figura 9) y haga clic sobre **OK.**

Los ficheros (las tablas) presentes en su base de datos aparecerán en este momento (figura 10). Haga doble clic sobre la tabla que desea seleccionar para este informe, admitrep, la cual será transferida automáticamente a la ventana Create Report Expert. Haga clic sobre **Done** y vuelva a la ventana Create Report Expert donde deberá hacer clic sobre **Siguiente** (figura 11). Se halla usted entonces sobre la pestaña (Select) Fields con la cual puede seleccionar los campos del/de los ficheros que desea que figuren en su INFORME. Haga clic sobre **All** si desea seleccionar todos los campos o bien pulse sobre todos aquellos campos que desee seleccionar, en el orden de aparición elegido y haga clic sobre **Insertar** (figura 12). Una vez terminada la selección de los campos haga clic sobre **Siguiente** para cambiar a la pestaña **Sort** (figura 13).

Seleccione los Report Fields en el orden en el que usted quiera clasificar e introdúzcalos en el Group Fields. Haga clic sobre **Siguiente** para cambiar a la pestaña **Totals.** (figura 14).

Seleccione el campo en el que quiere que figure la cantidad y entonces seleccione el método. (El ejemplo nos muestra un total de las entradas). Haga clic sobre **Preview Report**, las otras pestañas no son necesarias para generar este informe. Se trata de un informe poco elaborado, sin embargo contiene todos los campos, los grupos y las cantidades seleccionados.

Seleccione la pestaña Design de Crystal Reports (figura 15), modifique la anchura del campo, sustituya los encabezamientos de campo existentes y suprima el campo **pgnam** de la "detail section". Incorpore un encabezamiento atractivo y el logotipo del cliente. Coloque el campo fecha en el "trailer section" y añada el número de página. El producto final de tal manera elaborado atraerá poderosamente la atención de sus lectores. El informe presentará el aspecto de la (figura 16). Podrá completar su presentación con ayuda de gráficos y coloreando los diferentes campos.

Hasta aquí lo esencial de las operaciones que debe realizar para explotar al máximo sus bases de datos D3 utilizando Crystal Reports. Así que ya sabe, a partir de ahora y gracias a estas técnicas, usted podrá conferir un aspecto más actual a la presentación de sus aplicaciones.

14 de agosto de 1997

Figure 2

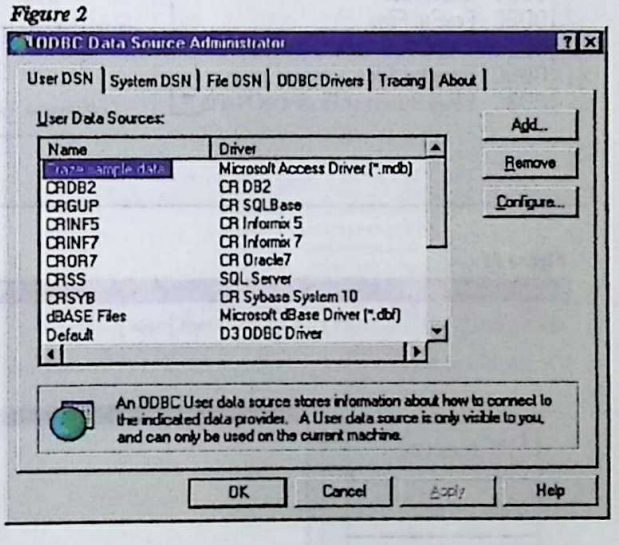


Figure 3

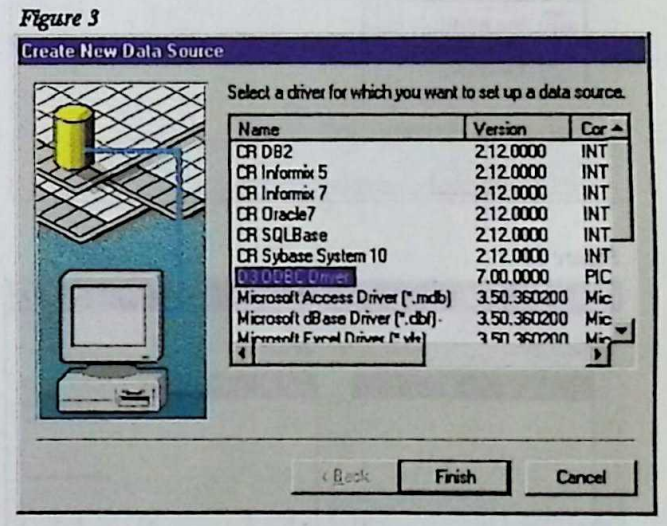


Figure 5

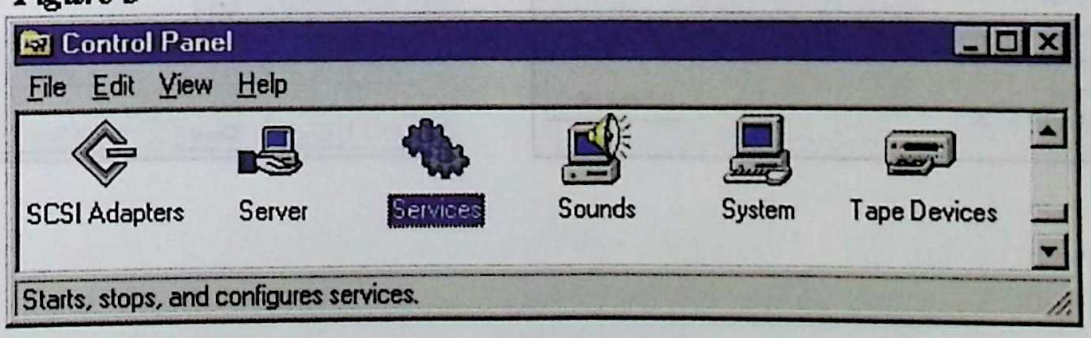


Figure 6

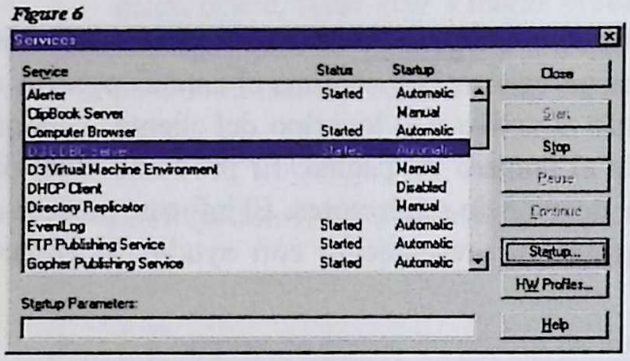


Figure 7

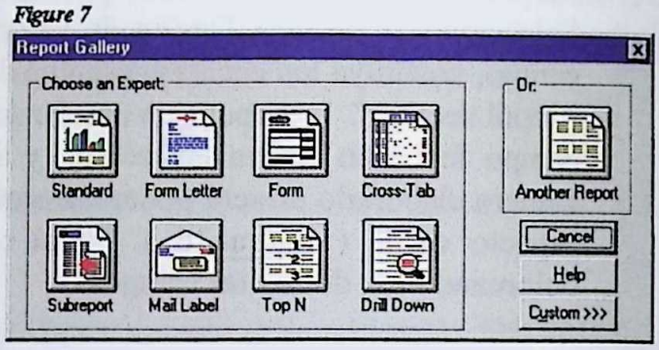


Figure 8

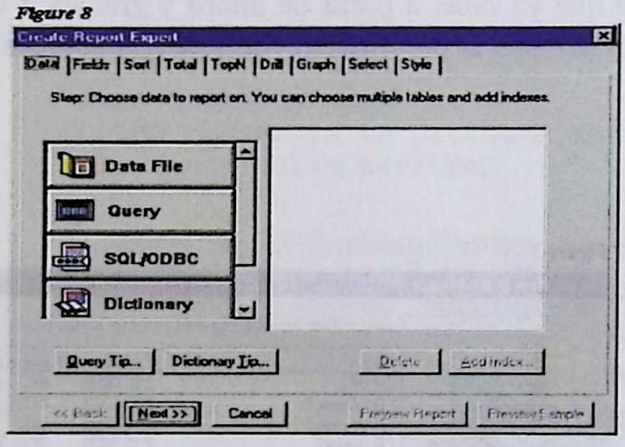


Figure 9

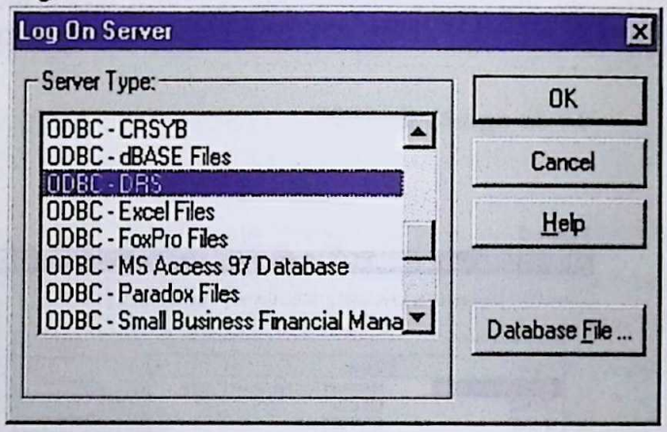


Figure 10

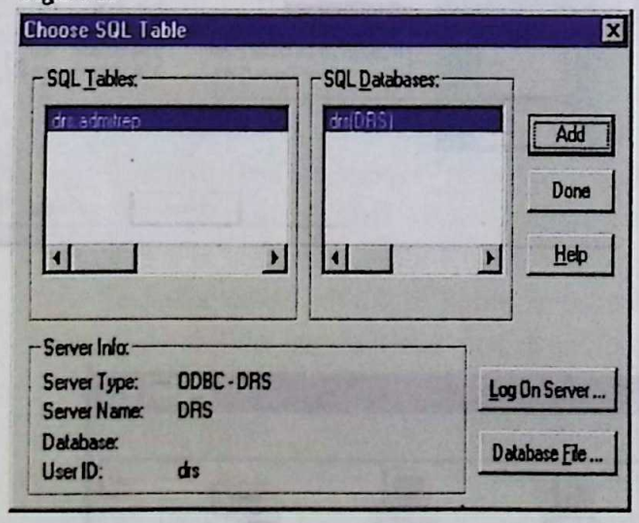
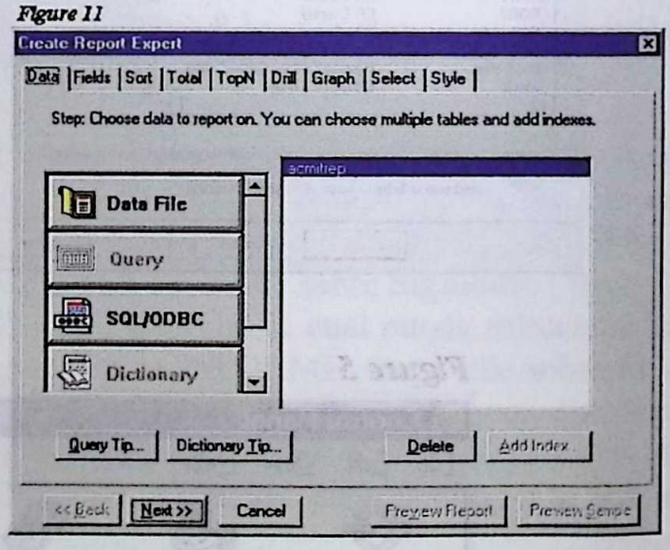


Figure 11



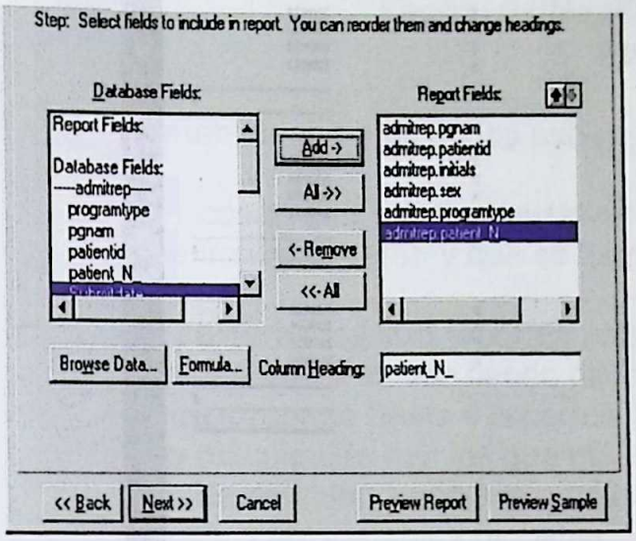


Figure 13

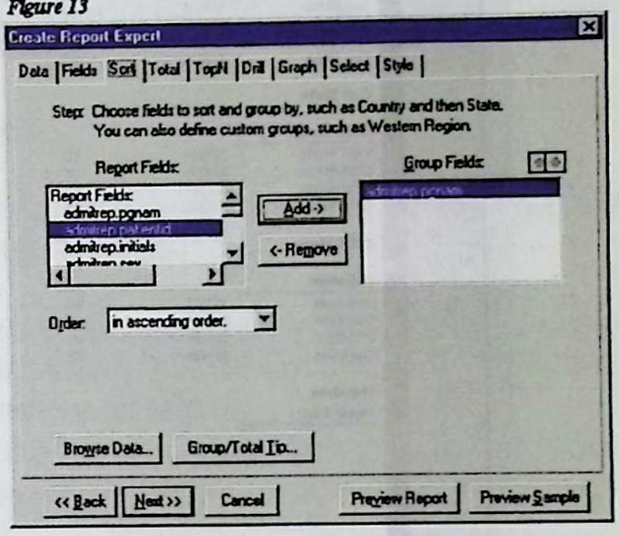


Figure 14

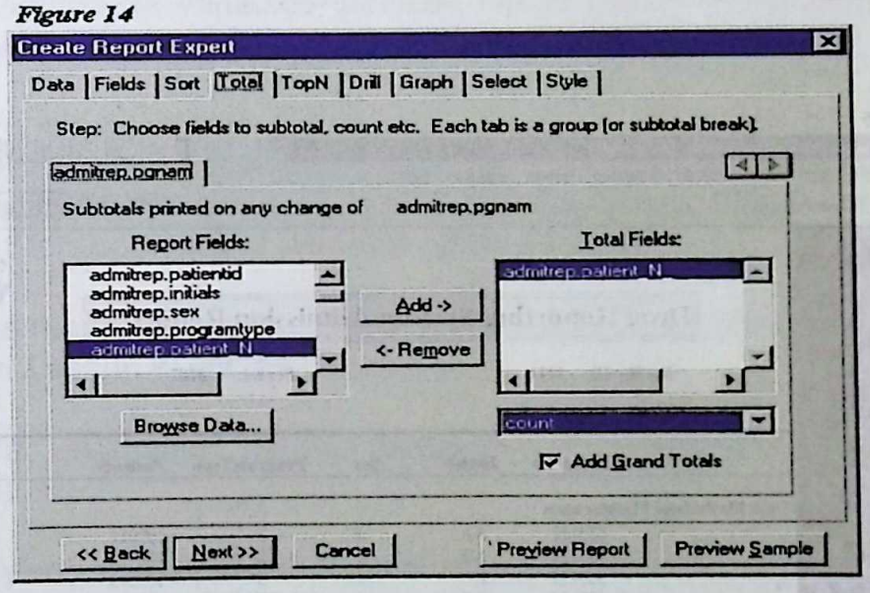


Figure 15

Crystal Reports Professional - [Report1]

File Edit View Insert Format Database Report Window Help

Design | Preview

Today 1:56 1 of 1

8/12/97

program	patientid	initials	sex	programtype	patient N
Cash Metha					
Cash Metha	555114	AD	1	S	100014
Cash Metha	555113	AC	1	S	100013
Cash Metha	555112	AB	1	S	100012
Cash Metha	555111	AA	1	S	100011
Cash Metha					4
DM/C Metha					
DM/C Metha	555134	CD	2	X	100034
DM/C Metha	555133	CC	2	X	100033
DM/C Metha	555132	CB	2	X	100032
DM/C Metha	555131	CA	2	X	100031
DM/C Metha					4
Outpatient					
Outpatient	555121	BA	1	P	100021
Outpatient	555124	BD	1	P	100024
Outpatient	555123	BC	1	P	100023
Outpatient	555122	BB	1	P	100022
Outpatient					4
Grand Total:					12

Figure 16

Crystal Reports Professional - [Admit.rpt]

File Edit View Insert Format Database Report Window Help

Design | Preview

7/15/97 1:23 1 of 1

TOWER systems inc.

Drug Reporting System Admission Report

Facility ID 1111 **Period** 9707

	PatientID	Initials	Sex	ProgramType	Patient#
Cash Methadone Maintenance					
	555111	AA	1	S	100011
	555112	AB	1	S	100012
	555113	AC	1	S	100013
	555114	AD	1	S	100014
Sub Total					4
DM/C Methadone Maintenance					
	555131	CA	2	X	100031
	555132	CB	2	X	100032
	555133	CC	2	X	100033
	555134	CD	2	X	100034
Sub Total					4
Outpatient Detoxification					
	555121	BA	1	P	100021
	555122	BB	1	P	100022
	555123	BC	1	P	100023

Artículos, trucos y comentarios sobre el Crystal...
Crystal Report 5.0
Variables Globales
Fecha: 04/May/98 (recibido: 29/04/98)
Autor: JJS

Uso de variables globales en los sub-reportes

Una de las cosas que más me gusto del Crytal Report en la versión 5.0 es una nueva opción que implementaron y que se llama sub reportes.

En pocas palabras los sub reportes permiten agrupar bajo un mismo reporte principal, varios reportes que a su vez tienen diferentes títulos, toman datos de tablas diferentes, etc. (Yo he agrupado hasta 4 reportes diferentes en uno solo).
Una de las dificultades con las que me encontré es que un subreporte utilizaba valores (resultado de cálculos) que a su vez quería utilizar en otro sub-reporte. Y bueno para eso se hicieron las variables globales.

Para guardar valores en variables globales debes utilizar la siguiente instrucción (Primero debes crear un formula y allí digitar lo siguiente).

Para guardar Variables: (según el tipo)
StoreStringVar (variable name,string), StoreDateVar (variable name,date) StoreCurrencyVar (variable name,currency),StoreNumberVar (variable name,number),StoreBooleanVar (variable name,boolean)

Ahora si quieres recuperar los valores de las variables en un sub-reporte debes utilizar esta instrucción.(según el tipo de variable)

FetchNumberVar (variable name),FetchStringVar (variable name),FetchDateVar (variable name)

FetchCurrencyVar (variable name),FetchBooleanVar (variable name)

Nota : Estas instrucciones no venían con el Crytal 5.0 que yo compre, por lo tanto para averiguarlo hable con la gente de soporte de Crystal y estos a su vez me explicaron y me enviaron un.zip que yo tambien té anexo para que lo utilices, ahora estas instrucciones solo las puedes utilizar en la version 5.0 y 6.0.

Generación de informes con Crystal Reports 8.0 - Orientado al usuario final.

Danyform - Area de formación de Danysoft

Generación de informes con Crystal Reports 8.0

Objetivos: Presentar las técnicas fundamentales de utilización de Crystal Reports 8.0, de manera que los usuarios sean capaces de desarrollar sus propios informes e introducir las características avanzadas que soporta Crystal Reports, con vistas a su futura utilización.

Requisitos: Experiencia de usuario con alguna herramienta de gestión de bases de datos. Familiaridad con Windows 95/98/ME.

Duración propuesta: 25 horas

Crystal Reports es la herramienta más utilizada hoy en día para la generación de informes para Windows, con más de un millón de usuarios registrados en todo el mundo. Entre sus **ventajas fundamentales** podemos citar: Entorno de diseño muy potente a la vez que fácil de utilizar - Posibilidad de acceso a bases de datos creadas con prácticamente todos los sistemas más populares del mercado, tanto locales como Cliente/Servidor - Motor de Impresión de libre distribución, para su uso en aplicaciones a la medida.

Temario

Introducción

· Introducción general a Crystal Reports.
· Arquitectura. Componentes principales.
· Acceso a datos.
· *Instalación y configuración.
· Base de datos de ejemplo.

Técnicas básicas de diseño de informes

· Utilización del Asistentes para Informes.
· Enlace visual de las tablas. Opciones de enlace.
· Técnicas principales de utilización del entorno de diseño. Objetos de texto, campos y formato.
· Informes estándar. Especificación de secciones, agrupación y totalización.
· Selección básica de registros.
· Incorporación de gráficos, mapas y tablas cruzadas (cross-tabs) a partir de los datos del informe.
· Campos especiales y formato avanzado. Experto de resaltado (Highlighting). Drill Down.
· Almacenamiento de informes.

· Exportación a otros formatos: Microsoft Word, Microsoft Excel, HTML.

Utilización de fórmulas en Crystal Reports

· Tipos de datos que reconoce Crystal Reports. Configuración de la interpretación y presentación de los mismos.
· Sintaxis del lenguaje de fórmulas Crystal y Visual Basic.
· Aplicaciones de las fórmulas: Campos calculados - Cálculo de porcentajes - Formateado e impresión condicionales - Selección de datos por registros y grupos.
· Informes con parámetros.
· Agregación de informes: los subinformes. Subinformes parametrizados. Hipervínculos.

Elementos avanzados

· Acceso a datos con driver nativo (cliente servidor) y con ODBC. Proceso en dos pasadas.
· Otras opciones de configuración.
· Otras fuentes de datos: Add-Ins de Access y Excel. - Creación y utilización de consultas SQL mediante Crystal Query. - Creación de diccionarios de datos con Crystal Dictionary. Su importancia práctica.
· Integración con servidor web. Visores de Internet.

Creación de Informes por Impresora.

A pesar de que podamos realizar tareas de impresión directamente desde VB. Imprimiendo incluso sobre el puerto PRN o LPT1 como si fuera un fichero de texto. Esto conlleva un espectacular incremento sobre las tareas a realizar dentro de la programación y todas sus rutinas.

Esto se puede complicar aun más cuando tratamos BB.DD. formadas por múltiples tablas y se requiere un conocimiento amplio del manejo de SQL.

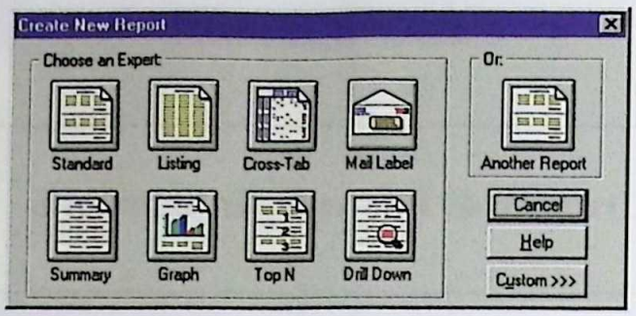
Podemos solventarlo rápido y fácilmente mediante la utilización de Crystal Reports.

Esta Herramienta se utiliza con Visual Basic a pesar de ser desarrollada por otra empresa y nos recuerda en algunas ocasiones al diseño de informes de Access mezclado con una antiguo dBase.

Utilización de Crystal Report

Creando un Nuevo Informe

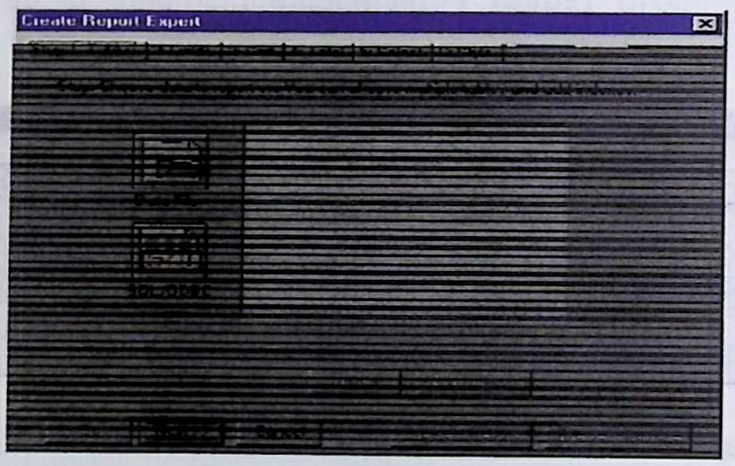
La primera labor es crear un nuevo informe para lo cual podemos utilizar el Bóton del folio para realizar esta tarea o (File/New). Presentando la siguiente Pantalla:



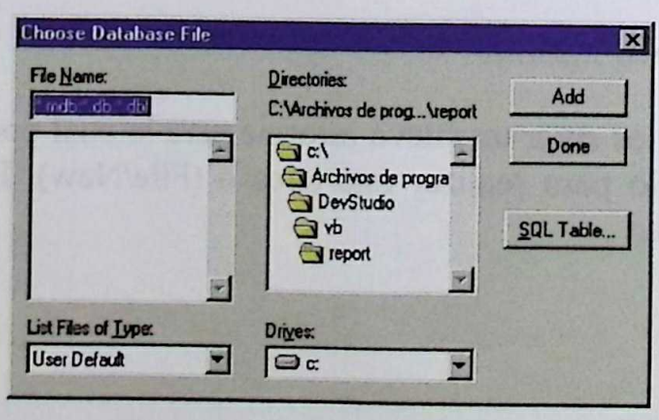
Permitiéndonos seleccionar uno u otro estilo de pendiendo de nuestra necesidades:

Standard

No guiará a través de un asistente el cual nos permitirá de una forma cómoda poder realizar las labores más o menos tediosas del informe:

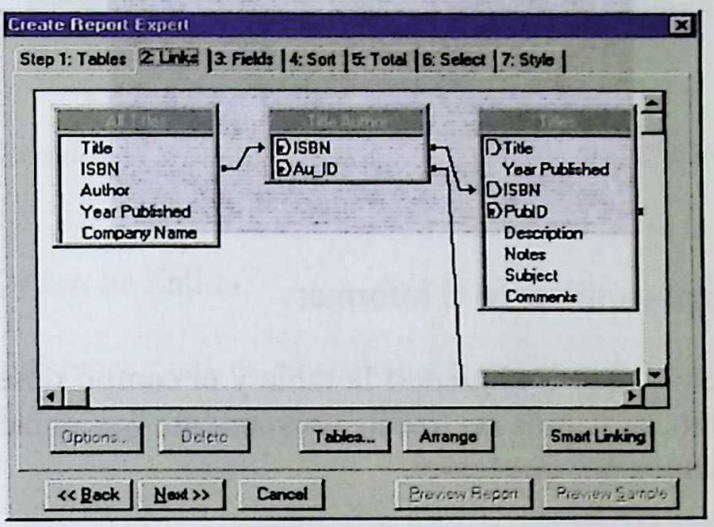


En el primero de los pasos podremos indicarle el nombre la base de datos pudiendo elegir entre múltiples tipos de bases de datos, como Access, dbase, Paradox, etc.



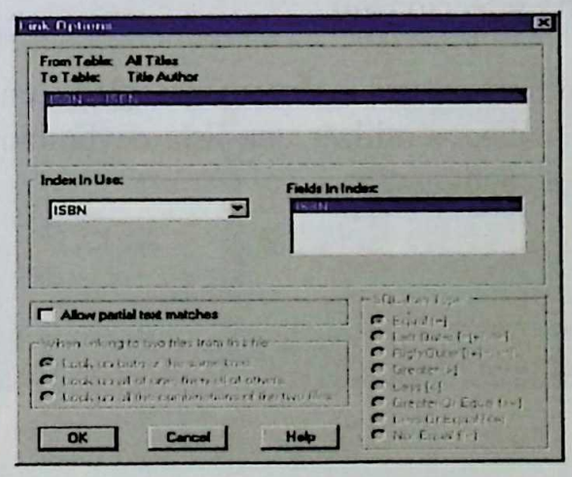
Solamente tenemos que indicarle cual es la unidad ruta y nombre de archivo en de la cual deseamos realizar el informe.

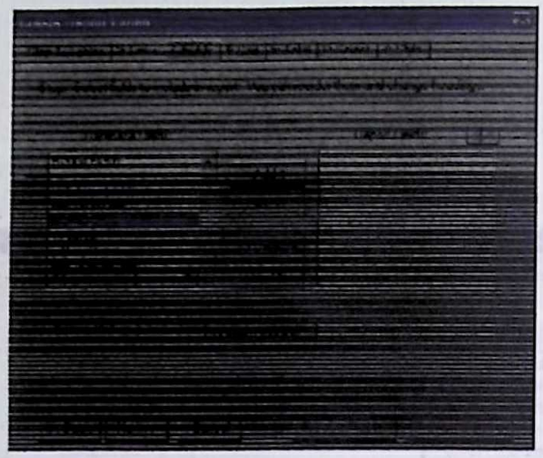
Una vez seleccionada la Base de Datos o la consulta de SQL o EL ODBC creado para el acceso a la base de datos por ejemplo SQL Server u Oracle. Pasaremos a la definición de los Link (Relaciones entre tablas)



Pudiendo eliminar , añadir o modificar el sistema de relaciones entre tablas estas serán vitales para el buen funcionamiento posterior de los informes ya que en caso contrario podemos encontrarnos verdaderos desastres en la salida del informe.

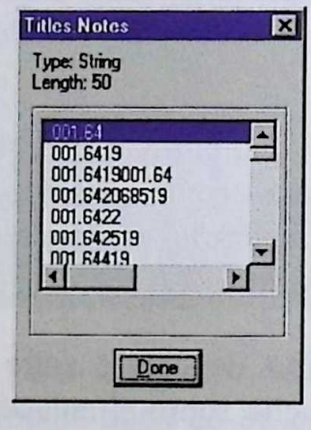
Para poder modificar el tipo de enlace entre las tablas solamente es necesario realizar un doble clic sobre el enlace apareciendo la siguiente pantalla en la que podremos modificar el tipo de acceso realizado.





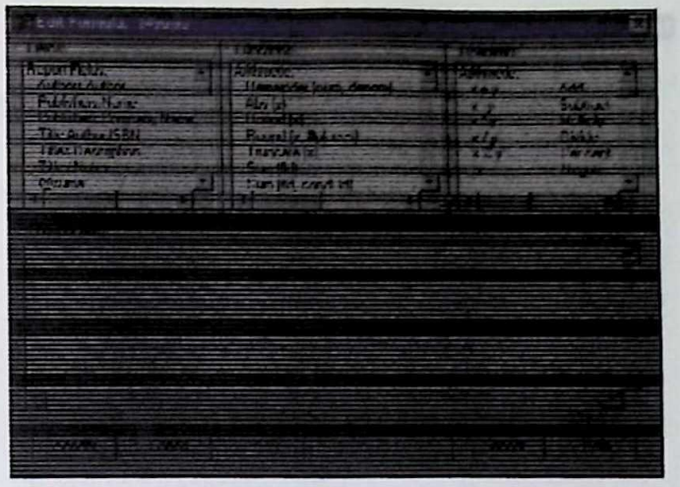
Elegir los campos a utilizar en el informe:

Solamente tenemos que ir eligiendo la tabla y el campo deseado de cada tabla en cuestión, si hemos realizado previamente los enlaces de forma correcta no tenemos por que temer.



Browse Data nos permite visualizar el contenido de un campo antes de poder utilizarlo como seleccionado

Formula permite introducir una columna calculada en función de una formula en la cual podemos utilizar una lista de funciones como la que podemos visualizar a continuación



Para más información utilice la guía de referencia técnica de crystal Report.

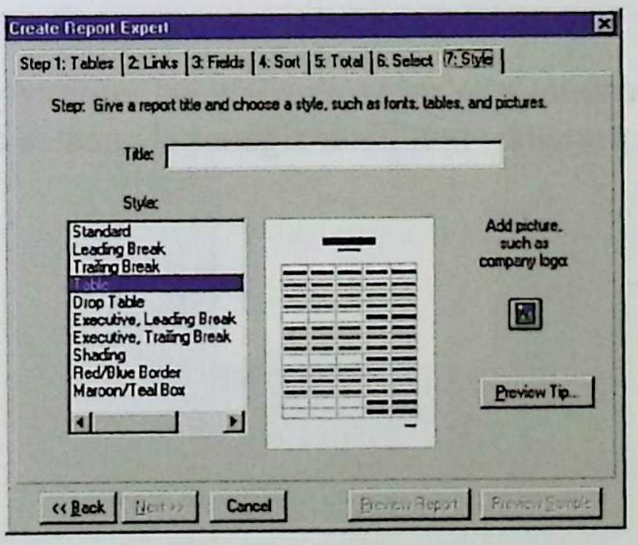
Indicando el Orden de Salida:

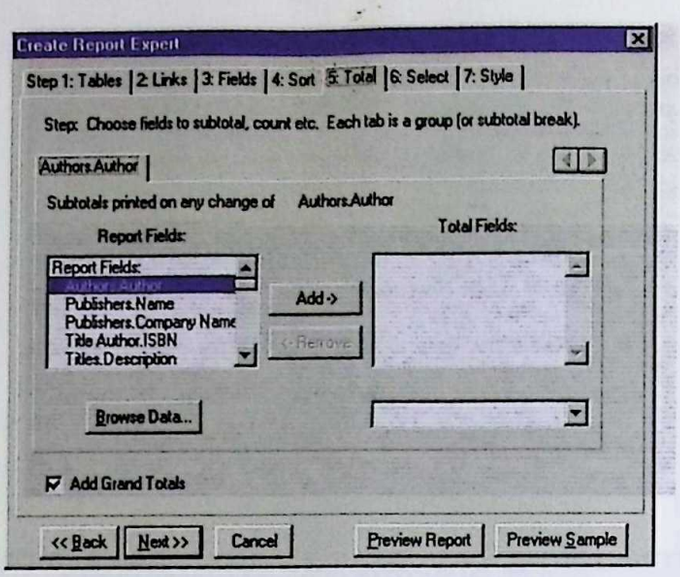
Para poder establecer cual es el orden de salida que deseamos realizar basta con establecer los campos deseados y el orden en que deseamos visualizarlos:

Totales:

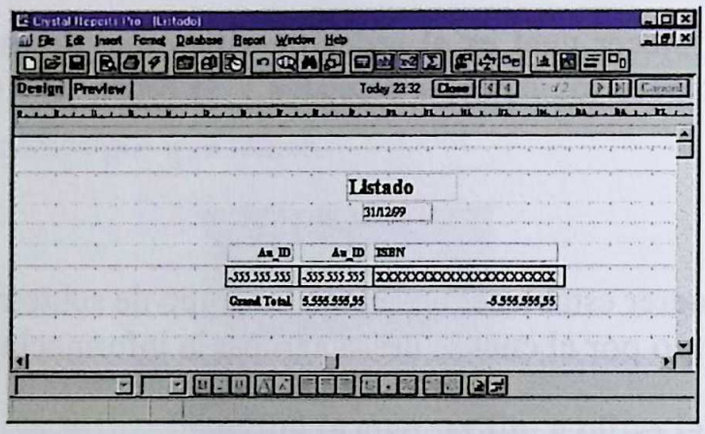
En el caso de querer establecer un subtotal o grupo de subtotales podemos indicarle el campo por el cual se desea agrupar la información.

Estableciendo el estilo a utilizar para el Informe:



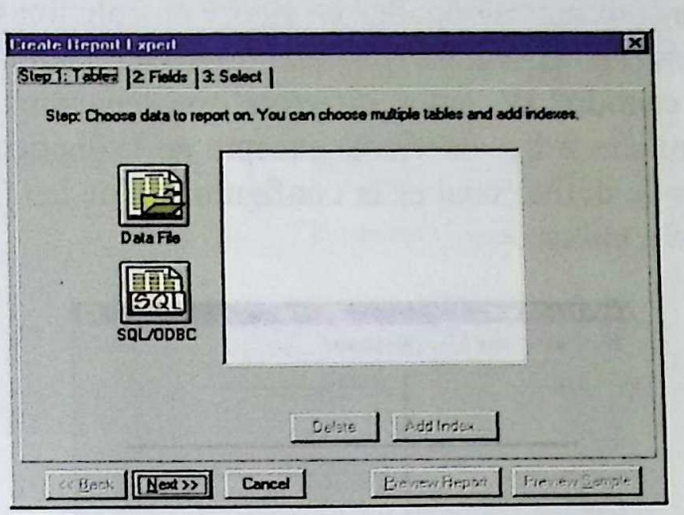


Podemos incluir una imagen que identifique el anagrama de la empresa así como la fisonomía general de informe.



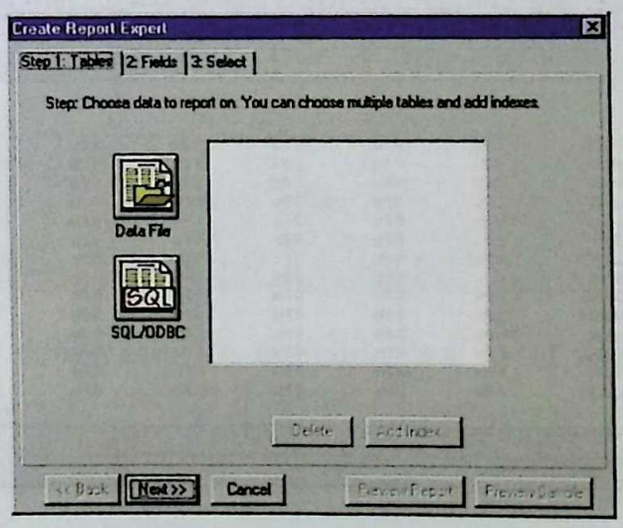
Y Finalmente podremos realizar un Preview del informe y poder realizar alguna prueba en pantalla o modificar algunos elementos del listado.

Listado Simple



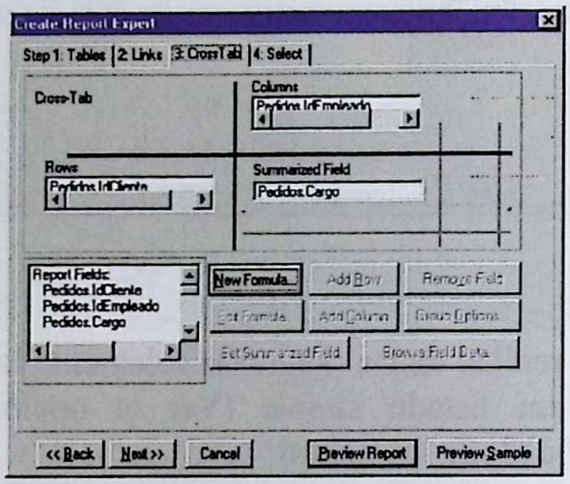
En esta ocasión no realiza tantas preguntas solamente las necesarias para la realización de un listado simple (Ver la opción standard para comprobar el significadode las fichas)

El Resultado es un informe Tabular

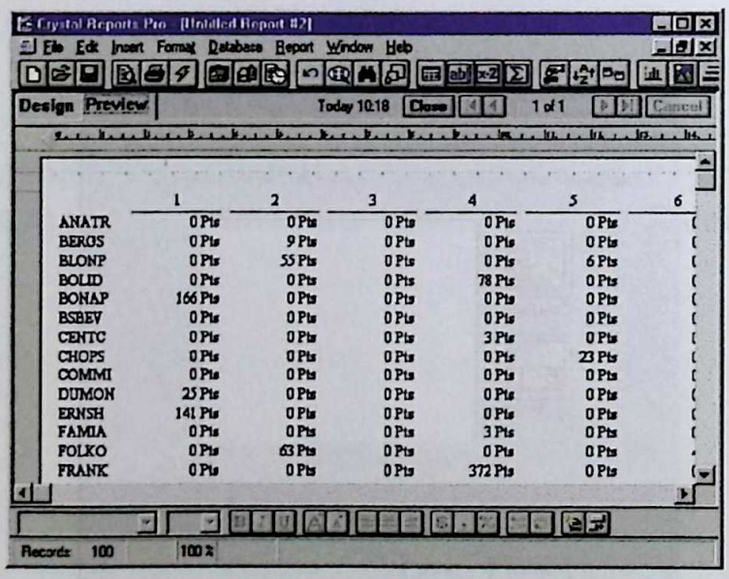


zada

Permite realizar informes agrupados en filas y en columnas de una forma cómoda además las columnas pueden dilatarse automáticamente en función de la cantidad de datos diferetes que tengamos. Los pasos a realizar son similares a los anteriores excepto en la siguiente pantalla en la que debemos de definir cual es la configuración de las filas, columnas y series de datos a utilizar.

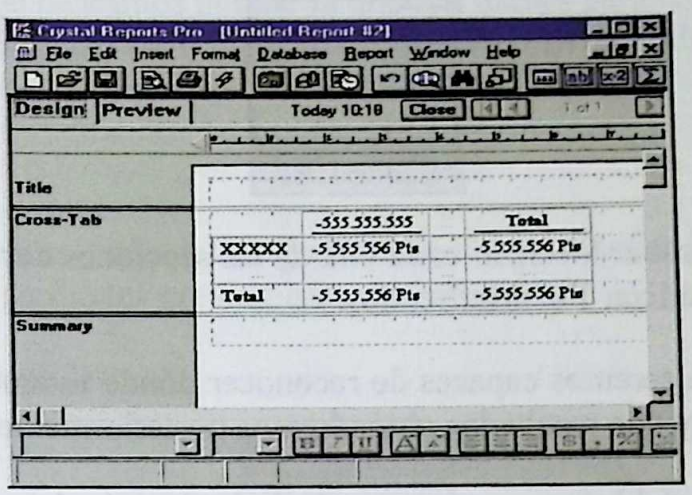


La salida sería similar a :



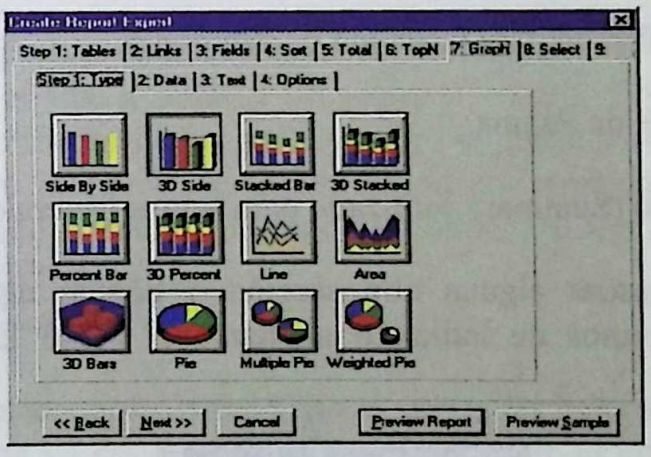
	1	2	3	4	5	6
ANATR	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
BERGS	0 Pts	9 Pts	0 Pts	0 Pts	0 Pts	
BLONP	0 Pts	55 Pts	0 Pts	0 Pts	6 Pts	
BOLID	0 Pts	0 Pts	0 Pts	78 Pts	0 Pts	
BONAP	166 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
BSBEV	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
CENTC	0 Pts	0 Pts	0 Pts	3 Pts	0 Pts	
CHOPS	0 Pts	0 Pts	0 Pts	0 Pts	23 Pts	
COMMI	0 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
DUMON	25 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
ERNSH	141 Pts	0 Pts	0 Pts	0 Pts	0 Pts	
FAMIA	0 Pts	0 Pts	0 Pts	3 Pts	0 Pts	
FOLKO	0 Pts	63 Pts	0 Pts	0 Pts	0 Pts	
FRANK	0 Pts	0 Pts	0 Pts	372 Pts	0 Pts	

Y la configuración del diseño sería :



Gráfico

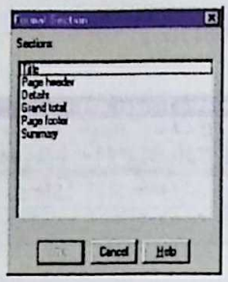
Este tipo de informe nos permite Mostrar de una forma sencillas un gráfico en nuestra base de datos para ello deberemos de seguir en gran parte los mismos pasos anteriores solamente difiren en algunas pantallas.



En esta pantalla deberemos de indicarle cual es el tipo de gráfico que deseamos visualizar en nuestro informe

El resto de los tipos solamente difieren en pequeños detalles respecto a los anteriores

Modificación del diseño del Informe



Lo primero es saber localizar cada una de las secciones del Informe para ello podemos utilizar **Format/Section**

De esta manera seremos capaces de reconocer donde estamos insertando la información y que resultados obtendremos posteriormente.:

Titulo: El título del Informe solo aparederá al principio del mismo.

Encabezados de Página (Page Header) utilizado para insertar toda aquella infromación que queremos que aparezca una sola vez y al principio de la página, por ejemplo el nombre de la tabla o el número de la Página o la fecha.

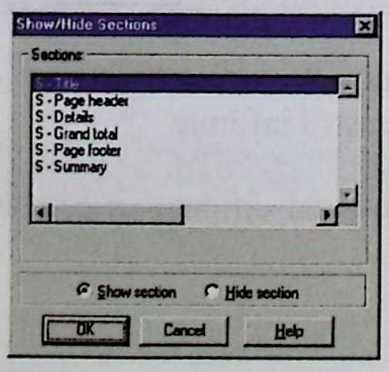
Detalles (Details) En esta zona se incorpora todos aquellos campos que serán posteriormente la información que veamos en la salida impresa como líneas de Registros.

Grand Total: Utilizado para inserción de subtotales.

Page footer : Pies de Página

Resumen General (Summary)utilizado para sumas generales o totales.

Si deseamos mostrar alguna otra sección o ocultar alguna existente solamente deberemos de indicarlo mediante el menú Edit/**ShowHide Section**

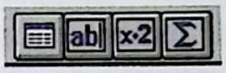


Aquí es donde podremos establecer cuales son aquellas secciones que realmente desamos visualizar y cuales no.

Por ejemplo que deseamos ocultar la línea de totales para el grupo bastará con seleccionar Grand total y posterioremente pulsa en Hide sectión Ello hará que desaparezca del informe tal zona.

También podemos utilizar la opción Edit Delete Section para eliminar algunas de las secciones previamente ocultadas.

Herramientas de Diseño



Para modificar un informe creado previamente podremos utilizar diferentes herramientas entre ellas nos encontramos con los iconos anteriores.

- Lista de Campos
- Inserción de Cuadros de Texto
- Formulas
- Funciones de Agrupación

Inserción de Infromes Crystal Report desde V.Basic 5.0.

Deberemos de añadir a nuestro proyecto el control Crystal Report e insertar posteriormente dicho control sobre el formulario desde el cual desemos realizar la operación de Impresión por jemeplo insertamos un botón de comando llamado cmdPrint que contiene el siguiete código:

```
Private Sub cmdPrint_Click()
CrystalReport1.Action = 1
End Sub
```

Esto hará que el control Cristal Report1 que antes hemos insertado en el formulario y que ya contiene la información sobre el informe a realizar muestre dicho informe **ReportFileName**

.

Aplicando Condiciones en el Informe

CrystalReport1.**SelectionFormula** = "{pedidos.idempleado}=1"
CrystalReport1.Action = 1
Para más información visualizar el editor

Cystal Reports el simplemente uno de los mejores generadores de reportes que existen en el mercado sirve para CUALQUIER base de datos, desde DBF, hasta archivos de Informix, Oracle, etc.

Crystal te permite generar de manera visual cualuqier tipo de reporte, o sub reporte, puedes manejar dentro de el variables, campos calculados incrustar fomulas, y controlarlo todo desde un programa FiveWin.

Los reportes de Crystal pueden exportarse de manera automatica a Excel, Word o bien a HTML para publicarlos en Internet.

Un demo ?, es un poco dificil porque veras, el producto viene en 32 diskettes ya que incluye las versiones de 16 y 32 bits, la mayoria de las herramientas de desarrollo de aplicaciones para Windows (como visual Basic, Delphi, dBase for Windows, etc.) incluyen versiones no profesionales de CRystal.

BIBLIOGRAFÍA CIBERNETICA

Pagina Principal de Microsoft España, donde puedes hacer búsquedas de cualquier información informática.
http://www.microsoft.com/Spain/Default.asp

Pagina de Microsoft donde te presentan una Guía de información del Visual Basic 98
http://www.microsoft.com/Spain/2000/Guia/32322es.htm

Pagina donde Puedes Accesar la Librería de Cualquier Producto de Microsoft
http://www.microsoft.com/Spain/2000/Guia/Product.asp

BIBLIOTECA DE MICROSOFT ESPAÑA

MSDN Library Visual Studio 6.0
MSDN Library es la referencia esencial para los desarrolladores, con más de un gigabyte de información de programación técnica, incluyendo código de ejemplo, documentación, artículos técnicos, Microsoft Developer Knowledge Base y cualquier otra cosa que pueda necesitar para desarrollar soluciones que mejoren la tecnología de Microsoft.
MSDN Library forma parte de la familia Visual Studio 6.0 de productos de programación, que incluye lo siguiente:

- **Visual Basic**
- **Visual C++**
- **Visual FoxPro**
- **Visual InterDev**
- **Visual J++**
- **Visual SourceSafe**
- **MSDN Library**
- **Visual Studio 6.0.**

LA DIRECCION ES:
http://www.microsoft.com/spain/msdn/library/

En esta Dirección encontrarás como poder usar el programa de Crystal Reports dando instrucciones Paso a Paso.
http://www.pick.co.uk/html/prod/espana/produit_espagne/crystal/crysta_espagne.htm

En esta Dicción encontrarás como generar informes con Crystal Reports
http://www.danysoft.com/cu_cr01.htm

Eljonat@Yahoo.com Elmer J. Lopez D.
Sac_Dquino@Yahoo.com Carlos A. Aquino L.