Visual Basic Reference

Visual Studio 6.0

Hide Method

See Also Example Applies To

Hides an **MDIForm** or **Form** object but doesn't unload it.

Syntax

object.**Hide**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object*.

Remarks

When a form is hidden, it's removed from the screen and its **Visible** property is set to **False**. A hidden form's controls aren't accessible to the user, but they are available to the running Visual Basic application, to other processes that may be communicating with the application through DDE, and to **Timer** control events.

When a form is hidden, the user can't interact with the application until all code in the event procedure that caused the form to be hidden has finished executing.

If the form isn't loaded when the **Hide** method is invoked, the **Hide** method loads the form but doesn't display it.

Note When closing a modal form that has been opened from another modal form, the following code worked in previous versions of Visual Basic:

```
Me.Hide
Me.Hide ' This now causes an error.
```

In current versions of Visual Basic, this code now fails on the second `Me.Hide`. You can substitute `Me.Hide` with `Me.Visible = False` as shown below:

```
Me.Visible = False
Me.Visible = False ' No error occurs.
```

Visual Basic Reference

Hide Method Example

This example uses the **Hide** method to hide a form. To try this example, paste the code into the Declarations section of a non-MDI form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Msg   ' Declare variable.
   Hide   ' Hide form.
   Msg = "Choose OK to make the form reappear."
   MsgBox Msg   ' Display message.
   Show   ' Show form again.
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

HitTest Method (ListView, TreeView Controls)

See Also Example Applies To

Returns a reference to the **ListItem** object or **Node** object located at the coordinates of x and y. Most often used with drag-and-drop operations to determine if a drop target item is available at the present location.

Syntax

object.**HitTest (***x* **As Single,** *y* **As Single)**

The **HitTest** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x,y	Coordinates of a target object, which is either a **Node** object or a **ListItem** object.

Remarks

If no object exists at the specified coordinates, the **HitTest** method returns **Nothing**.

The **HitTest** method is most frequently used with the **DropHighlight** property to highlight an object as the mouse is dragged over it. The **DropHighlight** property requires a reference to a specific object that is to be highlighted. In order to determine that object, the **HitTest** method is used in combination with an event that returns x and y coordinates, such as the DragOver event, as follows:

```
Private Sub TreeView1_DragOver _
(Source As Control, X As Single, Y As Single, State As Integer)
   Set TreeView1.DropHighlight = TreeView1.HitTest(X,Y)
End Sub
```

Subsequently, you can use the **DropHighlight** property in the DragDrop event to return a reference to the last object the source control was dropped over, as shown in the following code:

```
Private Sub TreeView1_DragDrop _
(Source As Control, x As Single, y As Single)
   ' DropHighlight returns a reference to object drop occurred over.
   Me.Caption = TreeView1.DropHighlight.Text
   ' To release the DropHighlight reference, set it to Nothing.
   Set TreeView1.DropHighlight = Nothing
End Sub
```

Note in the preceding example that the **DropHighlight** property is set to **Nothing** after the procedure is completed. This must be done to release the highlight effect.

Visual Basic: Windows Controls

HitTest Method (ListView, TreeView Controls) Example

This example adds several **Node** objects to a **TreeView** control. After you select a **Node** object, you can drag it to any other **Node**. To try the example, place **TreeView** and **ImageList** controls on a form and paste the code into the form's Declaration section. Run the example and drag **Node** objects around to see the result.

Note The graphics files in the code below can be found on Disk 1 of the Visual Basic or Visual Studio CDs, in the Common\Graphics directory. Change the path in the code, or copy the graphics files to your hard disk before running the code.

```
' Declare global variables.
Dim indrag As Boolean ' Flag that signals a Drag Drop operation.
Dim nodX As Object ' Item that is being dragged.

Private Sub Form_Load()
   ' Load a bitmap into an Imagelist control.
   Dim imgX As ListImage
   Dim BitmapPath As String
   BitmapPath = "icons\mail\mail01a.ico" ' Change to a valid path.
   Set imgX = imagelist1.ListImages.Add(, , LoadPicture(BitmapPath))

   ' Initialize TreeView control and create several nodes.
   TreeView1.ImageList = imagelist1
   Dim nodX As Node   ' Create a tree.
   Set nodX = TreeView1.Nodes.Add(, , , "Parent1", 1)
   Set nodX = TreeView1.Nodes.Add(, , , "Parent2", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 1", 1)
   Set nodX = TreeView1.Nodes.Add(1, tvwChild, , "Child 2", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 3", 1)
   Set nodX = TreeView1.Nodes.Add(2, tvwChild, , "Child 4", 1)
   Set nodX = TreeView1.Nodes.Add(3, tvwChild, , "Child 5", 1)
   nodX.EnsureVisible ' Expand tree to show all nodes.
End Sub

Private Sub TreeView1_MouseDown_
(Button As Integer, Shift As Integer, x As Single, y As Single)
   Set nodX = TreeView1.SelectedItem ' Set the item being dragged.
End Sub

Private Sub TreeView1_MouseMove _
(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = vbLeftButton Then ' Signal a Drag operation.
      indrag = True ' Set the flag to true.
      ' Set the drag icon with the CreateDragImage method.
      TreeView1.DragIcon = TreeView1.SelectedItem.CreateDragImage
      TreeView1.Drag vbBeginDrag ' Drag operation.
   End If
End Sub

Private Sub TreeView1_DragDrop_
```

```
(Source As Control, x As Single, y As Single)
   If TreeView1.DropHighlight Is Nothing Then
      Set TreeView1.DropHighlight = Nothing
      indrag = False
      Exit Sub
   Else
      If nodX = TreeView1.DropHighlight Then Exit Sub
      Cls
      Print nodX.Text & " dropped on " & TreeView1.DropHighlight.Text
      Set TreeView1.DropHighlight = Nothing
      indrag = False
   End If
End Sub

Private Sub TreeView1_DragOver(Source As Control, x As Single, y As Single, State As Integer)
   If indrag = True Then
      ' Set DropHighlight to the mouse's coordinates.
      Set TreeView1.DropHighlight = TreeView1.HitTest(x, y)
   End If
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

HitTest Method (MonthView Control)

See Also Example Applies To

Returns the date located at a particular set of coordinates. Most often used with drag-and-drop operations to determine if a drop target item is available at the present location.

Syntax

object.**HitTest(***x* **as Single,** *y* **As Single,** *Date* **As Date)**

The **HitTest** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
x, y	Coordinates of a target **date**.
Date	A date expression returned when a mouse operation occurs over a date on the control.

Return Settings

The **HitTest** method returns the following values which specify the part of the calendar over which the mouse pointer is hovering:

Constant	Value	Description
mvwCalendarBack	0	The calendar background.
mvwCalendarDate	1	Calendar date.
mvwCalendarDateNext	2	When this area is clicked, the calendar displays the following month.
mvwCalendarDatePrev	3	When this area is clicked, the calendar displays the previous month.
mvwCalendarDay	4	The day labels above the dates.
mvwCalendarWeekNum	5	The week number, if **ShowWeekNumbers** is set to **True**.
mvwNoWhere	6	Bottom edge of the calendar.

mvwTitleBack	7	Background of the calendar.
mvwTitleBtnNext	8	The Next button in the title area.
mvwTitleBtnPrev	9	The Previous button in the title area.
mvwTitleMonth	10	The month string in the title.
mvwTitleYear	11	The year string in the title.
mvwTodayLink	12	When this area is clicked, the calendar displays the current month and day. Only available if **ShowToday** is set to **True**.

Remarks

The **HitTest** method can be used to determine what area of the calendar is being affected by a mouse operation.

If no date exists at the specified coordinates, the **HitTest** method returns **Null**. The *Date* will also be Null if no date exists at the coordinates.

Visual Basic: Windows Controls

HitTest Method (MonthView Control) Example

This example uses the **HitTest** method to display the date in a label control as the mouse passes over it. To try the example, place **MonthView** and **Label** controls on a form. Then paste the following code into the Declarations section.

```
Private Sub MonthView1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   Dim iResult As Integer
   Dim dtMyDate As Date
   iResult = MonthView1.HitTest(x, y, dtMyDate)
   Label1.Caption = dtMyDate
End Sub
```

Visual Basic: DataGrid Control

Visual Studio 6.0

HoldFields Method

See Also Example Applies To

Sets the current column/field layout as the customized layout.

Syntax

object.**HoldFields**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The **HoldFields** method sets the current column/field layout as the customized layout so that subsequent ReBind operations will use the current layout for display. You can resume the grid's automatic layout behavior by invoking the **ClearFields** method.

Visual Basic Extensibility Reference

Visual Studio 6.0

Import Method (VBA Add-In Object Model)

See Also Example Applies To Specifics

Adds a component to a project from a file; returns the newly added component.

Syntax

object.**Import(***filename***) As VBComponent**

The **Import** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
filename	Required. A String specifying path and file name of the component that you want to import the component from.

Remarks

You can use the **Import** method to add a component, form, module, class, and so on, to your project.

Visual Basic Extensibility Reference

Import Method Example

The following example uses the **Import** method on the **VBComponents** collection to copy the contents of the test.bas file into a code module.

```
Application.VBE.ActiveVBProject.VBComponents.Import("test.bas")
```

Visual Studio 6.0

Visual Basic: MSChart Control

InitializeLabels Method

See Also Example Applies To

Assigns each label in the first level of data grid labels a unique identifier.

Syntax

object.**InitializeLabels**

The object placeholder represents an object expression that evaluates to an object in the Applies To list.

Visual Basic Reference

Visual Studio 6.0

Insert Method

See Also Example Applies To

Inserts an existing **ExportFormat** object into the **ExportFormats** collection.

Syntax

object.**Insert** *format*

The **Insert** method syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
format	The **ExportFormat** object to insert.

Visual Basic Reference

Insert Method, FileFilter, FileFormatString Property Example

The example creates a new **ExportFormat** object and inserts it into the **ExportFormats** collection.

```
Dim exfSpecial As New ExportFormat
With exfSpecial
   .FileFormatString = "Special Report (*.txt)"
   .FileFilter = "*.txt"
   .FormatType = rptFmtText
   .Key = "Special"
   .Template = "Special Report" & vbCrLf & rptTagBody
End With

With DataReport1
   .ExportFormats.Insert exfSpecial
   .ExportReport "Special"
End With
```

Visual Studio 6.0

Visual Basic: MSChart Control

InsertColumnLabels Method

See Also Example Applies To

Inserts levels of labels for the data columns in a data grid associated with a chart.

Syntax

object.**InsertColumnLabels** (*labelIndex*, *count*)

The **InsertColumnLabels** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
labelIndex	Integer. Identifies the number of the first level of labels you want to insert. Column label levels are numbered bottom to top, beginning with 1.
count	Integer. Specifies the number of label levels you want to insert. The number of columns being inserted is calculated from the column identified in *labelIndex* up.

Visual Studio 6.0

Visual Basic: MSChart Control

InsertColumns Method

See Also Example Applies To

Adds one or more data columns to the data grid associated with a chart.

Syntax

object.**InsertColumns** (*column*, *count*)

The **InsertColumns** method syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
column	Integer. Identifies a specific data column. Columns are numbered from left to right beginning with 1.
count	Integer. Specifies the number of columns you want to insert.

Visual Basic Extensibility Reference

Visual Studio 6.0

InsertFile Method

See Also Example Applies To Specifics

Inserts code from a file into a code module.

object.**InsertFile(***filename***) As String**

The **InsertFile** syntax has these parts:

Part	**Description**
object	Required. An object expression that evaluates to an object in the Applies To list.
filename	Required. A String specifying the file containing the code to insert into the code module.

Visual Basic Extensibility Reference

Visual Studio 6.0

InsertLines Method

See Also Example Applies To Specifics

Inserts a line or lines of code at a specified location in a block of code.

Syntax

object.**InsertLines(***line*, *code***)**

The **InsertLines** syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
line	Required. A Long specifying the location at which you want to insert the code.
code	Required. A String containing the code you want to insert.

Remarks

If the text you insert using the **InsertLines** method is carriage returnlinefeed delimited, it will be inserted as consecutive lines.

Visual Basic Extensibility Reference

InsertLines Method Example

The following example uses the **InsertLines** method to insert a line, Option Explicit, in the specified code pane.

```
Application.VBE.CodePanes(1).CodeModule.InsertLines 1, "Option Explicit"
```

Visual Basic Reference

Visual Studio 6.0

InsertObjDlg Method

See Also Example Applies To

Displays the Insert Object dialog box.

Syntax

object.**InsertObjDlg**

The *object* is an object expression that evaluates to an object in the Applies To list.

Remarks

At run time, you display this dialog box to enable the user to create a linked or embedded object by choosing the type of object (linked or embedded) and the application provides the object.

Use the **OLETypeAllowed** property to determine the type of object that can be created (linked, embedded, or either) using this dialog box.

When you create a new object, the application associated with the class name (for example, Excel.exe) must be correctly registered with the operating system. (The application setup program should register the application correctly.)

Visual Studio 6.0

Visual Basic: MSChart Control

InsertRowLabels Method

See Also Example Applies To

Inserts levels of labels from the data rows in a data grid associated with a chart.

Syntax

object.**InsertRowLabels** (*labelIndex*, *count*)

The **InsertRowLabels** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
labelIndex	Integer. Identifies the number of the first level of labels you want to insert. Row labels are numbered right to left, beginning with 1.
count	Integer. Specifies the number of label levels you want to insert. Row labels are inserted from the row identified by *labelIndex* to the left.

Visual Studio 6.0

Visual Basic: MSChart Control

InsertRows Method

See Also Example Applies To

Adds one or more data rows to the data grid associated with a chart.

Syntax

object.**InsertRows** (*row, count*)

The **InsertRows** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
row	Integer. Identifies a specific data row. Rows are numbered from top to bottom beginning with 1.
count	Integer. Specifies the number of rows you want to insert. Rows contain null data until you fill them with data.

Visual Basic Extensibility Reference

Visual Studio 6.0

Item Method (Add-Ins)

See Also Example Applies To

Returns an item from the specified collection by either name or index.

- AddIns collection:
- ContainedVBControls collection:
- Members collection:
- SelectedVBControls collection:
- VBControls collection:
- VBNewProjects collection:

Syntax

object.**Item** **(***index***)**

object.**Item** **(***collectionindex*, [*controlindex*]**)** **As VBControl**

object.**Item** **(***var***)** **As Member**

The **Item** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Required. A variant expression specifying the name or index in the collection of the object to be accessed.
collectionindex	Required. A numeric expression specifying the index
controlindex	Optional.
var	Required.

Remarks

There is no guarantee that a given index number for a collection will always point to the same item, because items may be added or deleted from the collection. Using index numbers for the collection is useful only when iterating through the whole collection and no items are added or deleted during the iteration.

Visual Basic Extensibility Reference

Visual Studio 6.0

Item Method (VBA Add-In Object Model)

See Also Example Applies To Specifics

Returns the indexed member of a collection.

Syntax

object.**Item(***index***)**

The **Item** method syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Required. An expression that specifies the position of a member of the collection. If a numeric expression, *index* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *index* must correspond to the *key* argument specified when the member was added to the collection.

The following table lists the collections and their corresponding *key* arguments for use with the **Item** method. The string you pass to the **Item** method must match the collection's *key* argument.

Collection	Key argument
Windows	**Caption** property setting
LinkedWindows	**Caption** property setting
CodePanes	No unique string is associated with this collection.
VBProjects	**Name** property setting
VBComponents	**Name** property setting
References	**Name** property setting
Properties	**Name** property setting

Remarks

The *index* argument can be a numeric value or a string containing the title of the object.

Visual Basic for Applications Reference

Visual Studio 6.0

Item Method

See Also Example Applies To Specifics

Returns a specific member of a **Collection** object either by position or by key.

Syntax

object.**Item(***index***)**

The **Item** method syntax has the following object qualifier and part:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Required. An expression that specifies the position of a member of the collection. If a numeric expression, *index* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *index* must correspond to the ***key*** argument specified when the member referred to was added to the collection.

Remarks

If the value provided as *index* doesnt match any existing member of the collection, an error occurs.

The **Item** method is the default method for a collection. Therefore, the following lines of code are equivalent:

```
Print MyCollection(1)
Print MyCollection.Item(1)
```

Visual Basic for Applications Reference

Item Method Example

This example uses the **Item** method to retrieve a reference to an object in a collection. Assuming `Birthdays` is a **Collection** object, the following code retrieves from the collection references to the objects representing Bill Smith's birthday and Adam Smith's birthday, using the keys "SmithBill" and "SmithAdam" as the *index* arguments. Note that the first call explicitly specifies the **Item** method, but the second does not. Both calls work because the **Item** method is the default for a **Collection** object. The references, assigned to `SmithBillBD` and `SmithAdamBD` using **Set**, can be used to access the properties and methods of the specified objects. To run this code, create the collection and populate it with at least the two referenced members.

```
Dim SmithBillBD As Object
Dim SmithAdamBD As Object
Dim Birthdays
Set SmithBillBD = Birthdays.Item("SmithBill")
Set SmithAdamBD = Birthdays("SmithAdam")
```

Visual Basic Reference

Visual Studio 6.0

Item Property (ActiveX Controls)

See Also Example Applies To

Returns a specific member of a **Collection** object either by position or by key.

Syntax

object.**Item(** *index* **)**

The **Item** property syntax has the following object qualifier and part:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
index	Required. An expression that specifies the position of a member of the collection. If a numeric expression, index must be a number from 1 to the value of the collection's Count property. If a string expression, index must correspond to the key argument specified when the member referred to was added to the collection.

Remarks

If the value provided as index doesnt match any existing member of the collection, an error occurs.

Item is the default property for a collection. Therefore, the following lines of code are equivalent:

```
Print MyCollection(1)
Print MyCollection.Item(1)
```

Visual Basic for Applications Reference

Visual Studio 6.0

Items Method

See Also Example Applies To Specifics

Description

Returns an array containing all the items in a **Dictionary** object.

Syntax

object.**Items**

The *object* is always the name of a **Dictionary** object.

Remarks

The following code illustrates use of the **Items** method:

```
Dim a, d, i              'Create some variables
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athens"      'Add some keys and items
d.Add "b", "Belgrade"
d.Add "c", "Cairo"
a = d.Items              'Get the items
For i = 0 To d.Count -1 'Iterate the array
    Print a(i)           'Print item
Next
...
```

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Keys Method

See Also Example Applies To Specifics

Description

Returns an array containing all existing keys in a **Dictionary** object.

Syntax

object.**Keys**

The *object* is always the name of a **Dictionary** object.

Remarks

The following code illustrates use of the **Keys** method:

```
Dim a, d, i              'Create some variables
Set d = CreateObject("Scripting.Dictionary")
d.Add "a", "Athens"      'Add some keys and items.
d.Add "b", "Belgrade"
d.Add "c", "Cairo"
a = d.keys               'Get the keys
For i = 0 To d.Count -1 'Iterate the array
    Print a(i)           'Print key
Next
...
```

Visual Basic Reference

Visual Studio 6.0

KillDoc Method

See Also Example Applies To

Immediately terminates the current print job.

Syntax

object.**KillDoc**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

If the operating system's Print Manager is handling the print job (the Print Manager is running and has background printing enabled), **KillDoc** deletes the current print job and the printer receives nothing.

If Print Manager isn't handling the print job (background printing isn't enabled), some or all of the data may be sent to the printer before **KillDoc** can take effect. In this case, the printer driver resets the printer when possible and terminates the print job.

Visual Basic Reference

KillDoc Method Example

This example uses the **KillDoc** method to terminate the current print job. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click()
   For i = 1 To 40
      Printer.CurrentX = 1440   ' Set left margin.
      Printer.CurrentY = (i * 300)   ' Advance page to next line.
      Printer.Print "This is line" & Str$(i) & " of text."
      On Error Resume Next   ' Catch any printer error.
      If i = 26 Then
         Printer.KillDoc   ' Terminate print job abruptly.
         Printer.EndDoc
         End
      End If
   Next i
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Layout Method

See Also Example Applies To

Lays out a chart, forcing recalculation of automatic values.

Syntax

object.**Layout**

The object placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

A chart is laid out the first time it is drawn. When any chart settings change, the chart is again laid out at the next draw. There are a number of settings the chart calculates, such as the axis minimum and maximum values, based on the chart type or some other setting. These values are not determined until the chart is laid out. If you attempt to "get" these automatic values before the chart is properly laid out, they will not reflect the new values.

Visual Basic Reference

Visual Studio 6.0

Line Method

See Also Example Applies To

Draws lines and rectangles on an object.

Syntax

object.**Line** [**Step**] **(***x1*, *y1***)** [**Step**] - **(***x2*, *y2***)**, [*color*], [**B**][**F**]

The **Line** method syntax has the following object qualifier and parts:

Part	**Description**
object	Optional. Object expression that evaluates to an object in the Applies To list. If object is omitted, the **Form** with the focus is assumed to be *object*.
Step	Optional. Keyword specifying that the starting point coordinates are relative to the current graphics position given by the **CurrentX** and **CurrentY** properties.
(*x1*, *y1***)**	Optional. **Single** values indicating the coordinates of the starting point for the line or rectangle. The **ScaleMode** property determines the unit of measure used. If omitted, the line begins at the position indicated by **CurrentX** and **CurrentY**.
Step	Optional. Keyword specifying that the end point coordinates are relative to the line starting point.
(*x2*, *y2***)**	Required. **Single** values indicating the coordinates of the end point for the line being drawn.
color	Optional. **Long** integer value indicating the RGB color used to draw the line. If omitted, the **ForeColor** property setting is used. You can use the **RGB** function or **QBColor** function to specify the color.
B	Optional. If included, causes a box to be drawn using the coordinates to specify opposite corners of the box.
F	Optional. If the **B** option is used, the **F** option specifies that the box is filled with the same color used to draw the box. You cannot use **F** without **B**. If **B** is used without **F**, the box is filled with the current **FillColor** and **FillStyle**. The default value for **FillStyle** is transparent.

Remarks

To draw connected lines, begin a subsequent line at the end point of the previous line.

The width of the line drawn depends on the setting of the **DrawWidth** property. The way a line or box is drawn on the background depends on the setting of the **DrawMode** and **DrawStyle** properties.

When **Line** executes, the **CurrentX** and **CurrentY** properties are set to the end point specified by the arguments.

This method cannot be used in an **WithEnd With** block.

Visual Basic Reference

Line Method Example

This example uses the **Line** method to draw concentric boxes on a form. To try this example, paste the code into the General section of a form. Then press F5 and click the form.

```
Sub Form_Click ()
   Dim CX, CY, F, F1, F2, I   ' Declare variables
   ScaleMode = 3   ' Set ScaleMode to pixels.
   CX = ScaleWidth / 2   ' Get horizontal center.
   CY = ScaleHeight / 2   ' Get vertical center.
   DrawWidth = 8   ' Set DrawWidth.
   For I = 50 To 0 Step -2
      F = I / 50   ' Perform interim
      F1 = 1 - F: F2 = 1 + F   ' calculations.
      Forecolor = QBColor(I Mod 15)   ' Set foreground color.
      Line (CX * F1, CY * F1)-(CX * F2, CY * F2), , BF
   Next I
   DoEvents   ' Yield for other processing.
   If CY > CX Then   ' Set DrawWidth.
      DrawWidth = ScaleWidth / 25
   Else
      DrawWidth = ScaleHeight / 25
   End If
   For I = 0 To 50 Step 2   ' Set up loop.
      F = I / 50   ' Perform interim
      F1 = 1 - F: F2 = 1 + F   ' calculations.
      Line (CX * F1, CY)-(CX, CY * F1)   ' Draw upper-left.
      Line -(CX * F2, CY)   ' Draw upper-right.
      Line -(CX, CY * F2)   ' Draw lower-right.
      Line -(CX * F1, CY)   ' Draw lower-left.
      Forecolor = QBColor(I Mod 15)   ' Change color each time.
   Next I
   DoEvents   ' Yield for other processing.
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LinkExecute Method

See Also Example Applies To

Sends a command string to the source application in a DDE conversation. Doesn't support named arguments.

Syntax

object.**LinkExecute** *string*

The **LinkExecute** method syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
string	Required. String expression containing a command recognized by the source application.

Remarks

The actual value of *string* varies depending on the source application. For example, Microsoft Excel and Microsoft Word for Windows accept command strings that consist of their macro commands enclosed in square brackets ([ ]). To see command strings that a source application accepts, consult documentation for that application.

Visual Basic Reference

LinkExecute Method Example

This example establishes a DDE link with Microsoft Excel, places some values into cells in the first row of a new worksheet, and charts the values. **LinkExecute** sends Microsoft Excel the command to activate a worksheet, select some values, and chart them. To try this example, Microsoft Excel must be installed on your computer and in the path statement of your Autoexec.bat file. Paste the code into the Declarations section of a form that has a **TextBox** control with the default name Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declare variables.
   Q = Chr(34)   ' Define quotation marks.
   ' Create a string containing Microsoft Excel macro commands.
   Cmd = "[ACTIVATE(" & Q &"SHEET1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)   ' Start Microsoft Excel.
      Text1.LinkTopic = "Excel|Sheet1"   ' Set link topic.
      Text1.LinkItem = "R1C1"   ' Set link item.
      Text1.LinkMode = vbLinkManual   ' Set link mode.
   End If
   For I = 1 To 5
      Row = I   ' Define row number.
      Text1.LinkItem = "R" & Row & "C1"   ' Set link item.
      Text1.Text = Chr(64 + I)   ' Put value in Text.
      Text1.LinkPoke   ' Poke value to cell.
      Text1.LinkItem = "R" & Row & "C2"   ' Set link item.
      Text1.Text = Row   ' Put value in Text.
      Text1.LinkPoke   ' Poke value to cell.
   Next I
   On Error Resume Next
   Text1.LinkExecute Cmd   ' Carry out Microsoft Excel commands.
   MsgBox "LinkExecute DDE demo with Microsoft Excel finished.", 64
 End
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LinkPoke Method

See Also Example Applies To

Transfers the contents of a **Label**, **PictureBox**, or **TextBox** control to the source application in a DDE conversation.

Syntax

object.**LinkPoke**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The *object* is the name of a **Label**, **PictureBox**, or **TextBox** involved in a DDE conversation as a destination. If *object* is a **Label**, **LinkPoke** transfers the contents of the **Caption** property to the source. If *object* is a **PictureBox**, **LinkPoke** transfers the contents of the **Picture** property to the source. If *object* is a **TextBox**, **LinkPoke** transfers the contents of the **Text** property to the source.

Typically, information in a DDE conversation flows from source to destination. However, **LinkPoke** allows a destination object to supply data to the source. Not all source applications accept information supplied this way; if the source application doesn't accept the data, an error occurs.

Visual Basic Reference

LinkPoke Method Example

This example establishes a DDE link with Microsoft Excel, places some values into cells in the first row of a new worksheet, and charts the values. **LinkPoke** sends the values to be charted to the Microsoft Excel worksheet. To try this example, Microsoft Excel must be installed and in the path statement of your Autoexec.bat file. Paste the code into the Declarations section of a form that has a **TextBox** control with the default name Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim Cmd, I, Q, Row, Z   ' Declare variables.
   Q = Chr(34)   ' Define quotation marks.
   ' Create a string containing Microsoft Excel macro commands.
   Cmd = "[ACTIVATE(" & Q &"SHEET1" & Q & ")]"
   Cmd = Cmd & "[SELECT(" & Q & "R1C1:R5C2" & Q & ")]"
   Cmd = Cmd & "[NEW(2,1)][ARRANGE.ALL()]"
   If Text1.LinkMode = vbNone Then
      Z = Shell("Excel", 4)   ' Start Microsoft Excel.
      Text1.LinkTopic = "Excel|Sheet1"   ' Set link topic.
      Text1.LinkItem = "R1C1"   ' Set link item.
      Text1.LinkMode = vbLinkManual   ' Set link mode.
   End If
   For I = 1 To 5
      Row = I   ' Define row number.
      Text1.LinkItem = "R" & Row & "C1"   ' Set link item.
      Text1.Text = Chr(64 + I)   ' Put value in Text.
      Text1.LinkPoke   ' Poke value to cell.
      Text1.LinkItem = "R" & Row & "C2"   ' Set link item.
      Text1.Text = Row   ' Put value in Text.
      Text1.LinkPoke   ' Poke value to cell.
   Next I
   Text1.LinkExecute Cmd   ' Carry out Microsoft Excel commands.
   On Error Resume Next
   MsgBox "LinkPoke DDE demo with Microsoft Excel finished.", 64
 End
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LinkRequest Method

See Also Example Applies To

Asks the source application in a DDE conversation to update the contents of a **Label**, **PictureBox**, or **TextBox** control.

Syntax

object.**LinkRequest**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The *object* is the name of a **Label**, **PictureBox**, or **TextBox** involved in a DDE conversation as a destination. **LinkRequest** causes the source application to send the most current data to *object*, updating the **Caption** property setting if *object* is a **Label**, the **Picture** property setting if *object* is a **PictureBox**, or the **Text** property setting if *object* is a **TextBox**.

If the **LinkMode** property of *object* is set to Automatic (1 or **vbLinkAutomatic**), the source application automatically updates *object* and **LinkRequest** isn't needed. If the **LinkMode** property of *object* is set to Manual (2 or **vbLinkManual**), the source application updates *object* only when **LinkRequest** is used. If the **LinkMode** property of *object* is set to Notify (3 or **vbLinkNotify**), the source notifies the destination that data has changed by invoking the LinkNotify event. The destination must then use **LinkRequest** to update the data.

Visual Basic Reference

LinkRequest Method Example

This example uses **LinkRequest** to update the contents of a text box with the values in a Microsoft Excel worksheet. To try this example, you must have Microsoft Excel running on your computer. Place some data in the first cells in the first column in the default worksheet (Sheet1.xls). Paste the code into the Declarations section of a form that has a **TextBox** control called Text1, and then press F5 and click the form.

```
Private Sub Form_Click ()
   If Text1.LinkMode = vbNone Then    ' Test link mode.
      Text1.LinkTopic = "Excel|Sheet1"    ' Set link topic.
      Text1.LinkItem = "R1C1"    ' Set link item.
      Text1.LinkMode = vbLinkManual    ' Set link mode.
      Text1.LinkRequest    ' Update text box.
   Else
      If Text1.LinkItem  = "R1C1" Then
         Text1.LinkItem = "R2C1"
         Text1.LinkRequest    ' Update text box.
      Else
         Text1.LinkItem = "R1C1"
         Text1.LinkRequest    ' Update text box.
      End If
   End If
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LinkSend Method

See Also Example Applies To

Transfers the contents of a **PictureBox** control to the destination application in a DDE conversation.

Syntax

object.**LinkSend**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remarks

The *object* must be a **PictureBox** on a **Form** object that is a source in a DDE conversation.

When other applications establish automatic links with a **Form** in your application, Visual Basic notifies them when the contents of a **TextBox** or a **Label** on the **Form** change. However, Visual Basic doesn't automatically notify a DDE destination application when the **Picture** property setting of a **PictureBox** on a source **Form** changes. Because the amount of data in a graphic can be very large and because it seldom makes sense to update a destination application as each pixel in the picture changes, Visual Basic requires that you use the **LinkSend** method to explicitly notify DDE destination applications when the contents of a **PictureBox** changes.

Visual Basic: Winsock Control

Visual Studio 6.0

Listen Method

See Also Example Applies To

Creates a socket and sets it in listen mode. This method works only for TCP connections.

Syntax

object.**Listen**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Arguments

None

Return Value

Void

Remarks

The ConnectionRequest event occurs when there is an incoming connection. When handling ConnectionRequest, the application should use the **Accept** method (on a new control instance) to accept the connection.

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

LoadFile Method

See Also Example Applies To

Loads an .rtf file or text file into a **RichTextBox** control.

Syntax

object.**LoadFile** *pathname*, *filetype*

The **LoadFile** method syntax has these parts:

Part	Description
object	Required. An object expression that evaluates to an object in the Applies To list.
pathname	Required. A string expression defining the path and filename of the file to load into the control.
filetype	Optional. An integer or constant that specifies the type of file loaded, as described in Settings.

Settings

The settings for *filetype* are:

Constant	Value	Description
rtfRTF	0	(Default) RTF. The file loaded must be a valid .rtf file.
rtfText	1	Text. The **RichTextBox** control loads any text file.

Remarks

When loading a file with the **LoadFile** method, the contents of the loaded file replaces the entire contents of the **RichTextBox** control. This will cause the values of the **Text** and **RTFText** properties to change.

You can also use the **Input** function in Visual Basic and the **TextRTF** and **SelRTF** properties of the **RichTextBox** control to read .rtf files. For example, you can load the contents of an .rtf file to the **RichTextBox** control as follows:

```
Open "mytext.rtf" For Input As 1
```

```
RichTextBox1.TextRTF = Strconv(InputB$(LOF(1), 1), vbUnicode)
```

Visual Basic: RichTextBox Control

LoadFile Method Example

This example displays a dialog box to choose an .rtf file, then loads that file into a **RichTextBox** control. To try this example, put a **RichTextBox** control, a **CommandButton** control, and a **CommonDialog** control on a form. Paste this code into the General Declarations section of the form. Then run the example.

```
Private Sub Command1_Click()
   CommonDialog1.Filter = "Rich Text Format files|*.rtf"
   CommonDialog1.ShowOpen
   RichTextBox1.LoadFile CommonDialog1.Filename, rtfRTF
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LogEvent Method

See Also Example Applies To

Logs an event in the application's log target. On Windows NT platforms, the method writes to the NT Event log. On Windows 95/98 platforms, the method writes to the file specified in the **LogPath** property; by default, if no file is specified, events will be written to a file named `vbevents.log`.

Syntax

object.**LogEvent (***logBuffer, eventType***)**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
logBuffer	Required. String to be written to the log.
eventType	Optional. A Long integer that specifies the type of event, as shown in Settings.

Settings

The settings for *eventType* are:

Constant	Value	Description
vbLogEventTypeError	1	Error.
vbLogEventTypeWarning	2	Warning.
vbLogEventTypeInformation	4	Information.

Remarks

Guidelines for logging are available in the Win32 SDK, and those guidelines should be followed when logging either to the NT Event log or the file specified in the **LogPath** property (on Windows 95/98 platforms).