Visual Basic Reference

Visual Studio 6.0

Label Control

See Also Example Properties Methods Events

A **Label** control is a graphical control you can use to display text that a user can't change directly.

Syntax

Label

Remarks

You can write code that changes the text displayed by a **Label** control in response to events at run time. For example, if your application takes a few minutes to commit a change, you can display a processing-status message in a **Label**. You can also use a **Label** to identify a control, such as a **TextBox** control, that doesn't have its own **Caption** property.

Set the **AutoSize** and **WordWrap** properties if you want the **Label** to properly display variable-length lines or varying numbers of lines.

A **Label** control can also act as a destination in a DDEconversation. Set the **LinkTopic** property to establish a link, set the **LinkItem** property to specify an item for the conversation, and set the **LinkMode** property to activate the link. When these properties have been set, Visual Basic attempts to initiate the conversation and displays a message if it's unable to do so.

Set the **UseMnemonic** property to **True** if you want to define a character in the **Caption** property of the **Label** as an access key. When you define an access key in a **Label** control, the user can press and hold down ALT+ the character you designate to move the focus to the next control in the tab order.

Visual Basic Reference

Visual Studio 6.0

Label Control (Data Report Designer)

See Also Example Properties Methods Events

A Label control is a graphical control you can use to display text that a user can't change directly.

Syntax

RptLabel

Remarks

The Data Report designer version of the Label control is similar to the standard Visual Basic intrinsic **Label** control in providing a method of displaying text on any application. Beyond this basic capability, however, some of the standard **Label** control's properties are unavailable on the Data Report version.

Visual Studio 6.0

Visual Basic: MSChart Control

Label Object

See Also Example Properties Methods Events

An item within a **Labels** collection that describes a specific chart axis label.

Syntax

axis.**Label**

Visual Studio 6.0

Visual Basic: MSChart Control

Labels Collection

See Also Example Properties Methods Events

A group of chart axis labels.

Syntax

axis.**Labels**(*index*)

The **Labels** collection syntax has these parts:

Part	Description
axis	An **Axis** object.
index	Identifies a specific axis label (**Label** object)within the current collection.

Visual Studio 6.0

Visual Basic: MSChart Control

LCoor Object

See Also Example Properties Methods Events

Describes a long integer *x* and *y* coordinate pair.

Syntax

Lcoor

Visual Studio 6.0

Visual Basic: MSChart Control

Legend Object

See Also Example Properties Methods Events

Represents the graphical key and accompanying text that describes a chart series.

Syntax

Legend

Visual Basic: MSChart Control

Legend Object Example

The following example sets the text and backdrop parameters for a chart legend.

```
Private Sub Command1_Click()
   With MSChart1.Legend
      ' Make Legend Visible.
      .Location.Visible = True
      .Location.LocationType = VtChLocationTypeRight
      ' Set Legend properties.
      .TextLayout.HorzAlignment = _
      VtHorizontalAlignmentRight    ' Right justify.
      ' Use Yellow text.
      .VtFont.VtColor.Set 255, 255, 0
      .Backdrop.Fill.Style = VtFillStyleBrush
      .Backdrop.Fill.Brush.Style = VtBrushStyleSolid
      .Backdrop.Fill.Brush.FillColor.Set 255, 0, 255
   End With
End Sub
```

Visual Basic Reference

Visual Studio 6.0

LicenseInfo Object

See Also Example Properties Methods Events

Represents the license key and programmatic ID (progID) of a control.

Syntax

LicenseInfo

Remarks

When adding a control that requires a license key to the Controls collection, you must first add the control's license to the Licenses collection. Using the Add method, each pair of license key and progID become properties of the LicenseInfo object.

To determine the license key and progId of each control, use the **For Each** statement with the **LicenseInfo** object, as shown below:

```
Dim li As LicenseInfo
For Each li in Licenses
   Debug.Print li.ProgID, li.LicenseKey
Next
```

Visual Basic Reference

Visual Studio 6.0

Licenses Collection

See Also Example Properties Methods Events

A collection of **LicenseInfo** objects that contain license key information required when adding a licensed control to the **Controls** collection.

Syntax

Licenses

Remarks

If a user control requires a license key, you must add the key to the **Licenses** collection before you add the control.

Visual Studio 6.0

Visual Basic: MSChart Control

Light Object

See Also Example Properties Methods Events

Represents the light source illuminating a three-dimensional chart.

Syntax

Light

Visual Basic: MSChart Control

Light Object Example

The following example sets the ambient light and edge lighting intensity for a chart. To try the example, draw an **MSChart** and two **ComboBox** controls on a form. Paste the code into the **Form** object's code module and run the project. Click the ComboBox controls to see the change of ambient light and edge intensity.

```
Private Sub Form_Load()
    ' Configure the chart.
   With MSChart1
      .Title = "Hold down Control and mousedown on chart"
      .chartType = VtChChartType3dBar
      .Plot.Light.AmbientIntensity = 1   ' 100 % Intensity.
      .Plot.Light.EdgeIntensity = 0.5        ' 50 % Intensity.
      .Plot.Light.EdgeVisible = True
   End With
   'Configure ComboBoxe controls.
   ConfigCombo Combo1
   ConfigCombo Combo2
End Sub

Private Sub ConfigCombo(cmb As ComboBox)
   ' Populate a combobox with values.
   Dim i As Single
   For i = 0 To 1 Step 0.1
      cmb.AddItem i
   Next i
   cmb.ListIndex = 0
End Sub

Private Sub Combo1_Click()
   MSChart1.Plot.Light.AmbientIntensity = Combo1.Text
End Sub

Private Sub Combo2_Click()
   MSChart1.Plot.Light.EdgeIntensity = Combo2.Text
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

LightSource Object

See Also Example Properties Methods Events

Represents the light source used to illuminate elements in a three-dimensional chart.

Syntax

LightSource

LightSources Collection

See Also Example Properties Methods Events

A group of **LightSource** objects in a chart.

Syntax

object.**LightSources**(*index*)

The **LightSources** collection syntax has the following parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Integer. A number that uniquely identifies a member of the collection.

Visual Basic Reference

Visual Studio 6.0

Line Control

See Also Example Properties Methods Events

A **Line** control is a graphical control displayed as a horizontal, vertical, or diagonal line.

Syntax

Line

Remarks

You can use a **Line** control at design time to draw lines on forms. At run time, you can use a **Line** control instead of, or in addition to, the **Line** method. Lines drawn with the **Line** control remain on the form even if the **AutoRedraw** property setting is **False**. **Line** controls can be displayed on forms, in picture boxes, and in frames. You can't use the **Move** method to move a **Line** control at run time, but you can move or resize it by altering its **X1**, **X2**, **Y1**, and **Y2** properties. The effect of setting the **BorderStyle** property depends on the setting of the **BorderWidth** property. If **BorderWidth** isn't 1 and **BorderStyle** isn't 0 or 6, **BorderStyle** is set to 1.

Visual Basic Extensibility Reference

Visual Studio 6.0

LinkedWindows Collection

See Also Example Properties Methods Events Specifics

Contains all linked windows in a linked window frame.

Remarks

Use the **LinkedWindows** collection to modify the docked and linked state of windows in the development environment.

The **LinkedWindowFrame** property of the **Window** object returns a **Window** object that has a valid **LinkedWindows** collection.

Linked window frames contain all windows that can be linked or docked. This includes all windows except code windows, designers, the Object Browser window, and the **Search and Replace** window.

If all the panes from one linked window frame are moved to another window, the linked window frame with no panes is destroyed. However, if all the panes are removed from the main window, it isn't destroyed.

Use the **Visible** property to check or set the visibility of a window.

You can use the **Add** method to add a window to the collection of currently linked windows. A window that is a pane in one linked window frame can be added to another linked window frame. Use the **Remove** method to remove a window from the collection of currently linked windows; this results in the window being unlinked or undocked.

The **LinkedWindows** collection is used to dock and undock windows from the main window frame.

Visual Basic Reference

Visual Studio 6.0

ListBox Control

See Also Example Properties Methods Events

A **ListBox** control displays a list of items from which the user can select one or more. If the number of items exceeds the number that can be displayed, a scroll bar is automatically added to the **ListBox** control.

If no item is selected, the **ListIndex** property value is -1. The first item in the list is **ListIndex** 0, and the value of the **ListCount** property is always one more than the largest **ListIndex** value.

Syntax

ListBox

Remarks

To add or delete items in a **ListBox** control, use the **AddItem** or **RemoveItem** method. Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the **ListBox**. Alternatively, you can add items to the list by using the **List** property at design time.

Visual Basic: Windows Controls

Visual Studio 6.0

ListImage Object, ListImages Collection

See Also Example Properties Methods Events

- A **ListImage** object is a bitmap of any size that can be used in other controls.
- A **ListImages** collection is a collection of **ListImage** objects.

Syntax

imagelist.**ListImages**

imagelist.**ListImages(***index***)**

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to standard collection syntax.

The **ListImage** Object, **ListImages** Collection syntaxes have these parts:

Part	Description
imagelist	Required. An object expression that evaluates to an object in the Applies To list.
index	An integer or string that uniquely identifies the object in the collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

The **ListImages** collection is a 1-based collection.

You can add and remove a **ListImage** at design time using the General tab of the ImageList Control Properties page, or at run time using the **Add** method for **ListImage** objects.

Each item in the collection can be accessed by its index or unique key. For example, to get a reference to the third **ListImage** object in a collection, use the following syntax:

```
Dim imgX As ListImage
   ' Reference by index number.
Set imgX = ImageList.ListImages(3)
   ' Or reference by unique key.
Set imgX = ImageList1.ListImages("third") ' Assuming Key is "third."
   ' Or use Item method.
Set imgX = ImageList1.ListImages.Item(3)
```

Each **ListImage** object has a corresponding mask that is generated automatically using the **MaskColor** property. This mask is not used directly, but is applied to the original bitmap in graphical operations such as the **Overlay** and **Draw** methods.

Visual Basic: Windows Controls

Visual Studio 6.0

ListItem Object, ListItems Collection

See Also Example Properties Methods Events

- A **ListItem** consists of text, the index of an associated icon (**ListImage** object), and, in Report view, an array of strings representing subitems.
- A **ListItems** collection contains one or more **ListItem** objects.

Syntax

listview.**ListItems**

listview.**ListItems**(*index*)

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

The **ListItem** object, **ListItems** collection syntax has these parts:

Part	Description
listview	An object expression that evaluates to a **ListView** control.
index	Either an integer or string that uniquely identifies a member of a **ListItem** collection. The integer is the value of the **Index** property; the string is the value of the **Key** property.

Remarks

ListItem objects can contain both text and pictures. However, to use pictures, you must reference an **ImageList** control using the **Icons** and **SmallIcons** properties.

You can also change the image by using the **Icon** or **SmallIcon** property.

The following example shows how to add **ColumnHeaders** and several **ListItem** objects with subitems to a **ListView** control.

```
Private Sub Form_Load()
   Dim clmX As ColumnHeader
   Dim itmX As ListItem
   Dim i As Integer

   For i = 1 To 3
      Set clmX = ListView1.ColumnHeaders.Add()
```

```
      clmX.Text = "Col" & i
   Next i

   ' Add 10 items to list, all with the same icon

   For i = 1 To 10
      Set itmX = ListView1.ListItems.Add()
      itmX.SmallIcon = 1
      itmX.Text = "ListItem " & i
      itmX.SubItems(1) = "Subitem 1"
      itmX.SubItems(2) = "Subitem 2"
   Next i
End Sub
```

Visual Basic: Windows Controls

Add Method (ListItems, ColumnHeaders), ListItems Property, SubItems Property Example

The following example uses the Biblio.mdb database as a source to populate a **ListView** control with **ListItem** objects. To try this example, place a **ListView** control on a form and paste the code into the Declarations section. You must also be sure that the Biblio.mdb has been installed on your machine. In the code below, check the path in the **OpenDatabase** function and change it to reflect the actual path to Biblio.mdb on your machine.

Note The example will not run unless you add a reference to the Microsoft DAO 3.51 Object Library. To do this, on the **Project** menu click **References**. Search for Microsoft DAO 3.51 Object Library and click the checkbox to select it.

```
Private Sub Form_Load()
   ' Add ColumnHeaders. The width of the columns is
   ' the width of the control divided by the number of
   ' ColumnHeader objects.
   ListView1.ColumnHeaders. _
   Add , , "Author", ListView1.Width / 3
   ListView1.ColumnHeaders. _
   Add , , "Author ID", ListView1.Width / 3, _
   lvwColumnCenter
   ListView1.ColumnHeaders. _
   Add , , "Birthdate", ListView1.Width / 3
   ' Set View property to Report.
   ListView1.View = lvwReport

   ' Declare object variables for the
   ' Data Access objects.
   Dim myDb As Database, myRs As Recordset
   ' Set the Database to the BIBLIO.MDB database.
   ' IMPORTANT: the Biblio.mdb must be on your
   ' machine, and you must set the correct path to
   ' the file in the OpenDatabase function below.
   Set myDb =    DBEngine.Workspaces(0) _
      .OpenDatabase("c:\Program Files\VB\BIBLIO.MDB")
   ' Set the recordset to the "Authors" table.
   Set myRs = _
   myDb.OpenRecordset("Authors", dbOpenDynaset)

   ' Declare a variable to add ListItem objects.
   Dim itmX As ListItem

   ' While the record is not the last record,
   ' add a ListItem object. Use the author field for
   ' the ListItem object's text. Use the AuthorID
   ' field for the ListItem object's SubItem(1).
   ' Use the "Year of Birth" field for the ListItem
   ' object's SubItem(2).

   While Not myRs.EOF
```

```
      Set itmX = ListView1.ListItems. _
      Add(, , CStr(myRs!Author))   ' Author.

      ' If the AuthorID field is not null, then set
      ' SubItem 1 to it.
      If Not IsNull(myRs!Au_id) Then
         itmX.SubItems(1) = CStr(myRs!Au_id)
      End If

      ' If the birth field is not Null, set
      ' SubItem 2 to it.
      If Not IsNull(myRs![Year Born]) Then
         itmX.SubItems(2) = myRs![Year Born]
      End If
      myRs.MoveNext   ' Move to next record.
   Wend
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

ListSubItem Object

See Also Example Properties Methods Events

The **ListSubItem** object represents a subitem in a **ListView** control.

Syntax

ListSubItem

Remarks

The **ListSubItem** object replaces the **SubItems** string array found in previous versions of the **ListView** control.

Visual Basic: Windows Controls

ListSubItem Object, ListSubItems Collection Example

The example creates twenty **ListItem** objects; for each object, four **ListSubItem** objects are also created. Before creating **ListSubItem** objects, however, the code creates five **ColumnHeader** objectsone for the **ListItem** object, and four for the **ListSubItem** objects. To try the example, place a **ListView** control on a form and paste the code into the Declarations section of the code.

```
Option Explicit

Private Sub Form_Load()
   Dim i As Integer ' Counter
   Dim j As Integer ' Counter for ListSubItems
   Dim sngWidth As Single
   Dim si As ListSubItem
   Dim li As ListItem

   ' You can't see ColumnHeaders or ListSubitems
   ' unless the View is set to lvwReport.
   ListView1.View = lvwReport

   ' Calculate the width of a ColumnHeader object.
   sngWidth = ListView1.Width / 5

   ' Create five ColumnHeader objects.
   For i = 1 To 5
      ListView1.ColumnHeaders.Add Text:="Col " & i, Width:=sngWidth
   Next i

   ' Create twenty ListItem objects. For each ListItem, create four
   ' ListSubItem objects. Set the ForeColor for each object to red.
   For i = 1 To 20
      Set li = ListView1.ListItems.Add(Text:="Item " & i)
      For j = 1 To 4
         Set si = li.ListSubItems.Add(Text:="Subitem " & j)
         si.ForeColor = vbRed
      Next j
   Next i
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

ListSubItems Collection

See Also Example Properties Methods Events

A collection of **ListSubItem** objects.

Syntax

ListSubItems

Remarks

Like **ListItem** objects, **ListSubItem** objects can only be created at run time using the **Add** method. Use the **ListSubItems** property to return a reference to the collection.

Visual Basic: Windows Controls

Visual Studio 6.0

ListView Control

See Also Example Properties Methods Events

The **ListView** control displays items using one of four different views. You can arrange items into columns with or without column headings as well as display accompanying icons and text.

Syntax

`ListView`

Remarks

With a **ListView** control, you can organize list entries, called **ListItem** objects, into one of four different views:

- Large (standard) Icons
- Small Icons
- List
- Report

The **View** property determines which view the control uses to display the items in the list. You can also control whether the labels associated with items in the list wrap to more than one line using the **LabelWrap** property. In addition, you can manage how items in the list are sorted and how selected items appear.

The **ListView** control contains **ListItem** and **ColumnHeader** objects. A **ListItem** object defines the various characteristics of items in the **ListView** control, such as:

- A brief description of the item.
- Icons that may appear with the item, supplied by an **ImageList** control.
- Additional pieces of text, called subitems, associated with a **ListItem** object that you can display in Report view.

You can choose to display column headings in the **ListView** control using the **HideColumnHeaders** property. They can be added at both design and run time. At design time, you can use the Column Headers tab of the **ListView** Control Properties dialog box. At run time, use the **Add** method to add a **ColumnHeader** object to the **ColumnHeaders** collection.

Distribution Note The **ListView** control is part of a group of ActiveX controls that are found in the MSCOMCTL.OCX file. To use the **ListView** control in your application, you must add the MSCOMCTL.OCX file to the project. When distributing your application, install the MSCOMCTL.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a Visual Basic project, see the Visual Basic *Programmer's Guide*.

Visual Studio 6.0

Visual Basic: MSChart Control

Location Object

See Also Example Properties Methods Events

Represents the current position of a textual chart element such as the title, legend, or footnote.

Syntax

Location

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

Visual Basic: MSChart Control

Location Object Example

The following example sets the title location for a chart using the **Location** object. To try the example, draw an **MSChart** and a **ComboBox** control on a form. Paste the code into the **Form** object's code module, and run the project. Click the **ComboBox** to change the title location.

```
Option Explicit

Private Sub Combo1_Click()
   MSChart1.Title.Location.LocationType = Combo1.ListIndex
End Sub

Private Sub Form_Load()
   ' Set Title Text.
   With MSChart1
      .TitleText = "Test Title Location"
      .Title.Location.Visible = True
   End With

   ' Add LocationType constants to ComboBox.
   With Combo1
      .AddItem "VtChLocationTypeTopLeft"     ' 0
      .AddItem "VtChLocationTypeTop"         ' 1
      .AddItem "VtChLocationTypeTopRight"    ' 2
      .AddItem "VtChLocationTypeLeft"        ' 3
      .AddItem "VtChLocationTypeRight"       ' 4
      .AddItem "VtChLocationTypeBottomLeft"  ' 5
      .AddItem "VtChLocationTypeBottom"      ' 6
      .AddItem "VtChLocationTypeBottomRight" ' 7
      .AddItem "VtChLocationTypeCustom"      ' 8
      .ListIndex = 0
   End With
End Sub
```