This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

CategoryScale Object

See Also Example Properties Methods Events

The scale for a category axis.

Syntax

CategoryScale

© 2017 Microsoft

Visual Basic: MSChart Control

CategoryScale Object Example

The following example sets the scaling attributes for a category axis.

```
Private Sub Command1_Click()
   ' Sets scaling attributes for a category axis.
   MSChart1.ChartType = VtChChartType2dLine
   With MSChart1.Plot.Axis(VtChAxisIdX)
      .ValueScale.MajorDivision = 10
      .ValueScale.MinorDivision = 5
      .CategoryScale.Auto = False          ' Sets manual scaling.
      .CategoryScale.DivisionsPerLabel = 2 ' Label appears every two
                                           ' divisions.
      .CategoryScale.DivisionsPerTick = 2  ' Ticks appear every two
                                           ' divisions.
      .CategoryScale.LabelTick = True    ' Labels displayed on top of
                                       ' Tick marks.
   End With
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

CheckBox Control

See Also Example Properties Methods Events

A **CheckBox** control displays an X when selected; the X disappears when the **CheckBox** is cleared. Use this control to give the user a True/False or Yes/No option. You can use **CheckBox** controls in groups to display multiple choices from which the user can select one or more. You can also set the value of a **CheckBox** programmatically with the Value property.

Syntax

CheckBox

Remarks

CheckBox and **OptionButton** controls function similarly but with an important difference: Any number of **CheckBox** controls on a form can be selected at the same time. In contrast, only one **OptionButton** in a group can be selected at any given time.

To display text next to the **CheckBox**, set the **Caption** property. Use the **Value** property to determine the state of the controlselected, cleared, or unavailable.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Clipboard Object

See Also Example Properties Methods Events

Provides access to the system Clipboard.

Syntax

Clipboard

Remarks

The **Clipboard** object is used to manipulate text and graphics on the Clipboard. You can use this object to enable a user to copy, cut, and paste text or graphics in your application. Before copying any material to the **Clipboard** object, you should clear its contents by as performing a **Clear** method, such as `Clipboard.Clear`.

Note that the **Clipboard** object is shared by all Windows applications, and thus, the contents are subject to change whenever you switch to another application.

The **Clipboard** object can contain several pieces of data as long as each piece is in a different format. For example, you can use the **SetData** method to put a bitmap on the **Clipboard** with the **vbCFDIB** format, and then use the **SetText** method with the **vbCFText** format to put text on the **Clipboard**. You can then use the **GetText** method to retrieve the text or the **GetData** method to retrieve the graphic. Data on the **Clipboard** is lost when another set of data of the same format is placed on the **Clipboard** either through code or a menu command.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CodeModule Object

See Also Example Properties Methods Events Specifics

Represents the code behind a component, such as a form, class, or document.

Remarks

You use the **CodeModule** object to modify (add, delete, or edit) the code associated with a component.

Each component is associated with one **CodeModule** object. However, a **CodeModule** object can be associated with multiple code panes.

The methods associated with the **CodeModule** object enable you to manipulate and return information about the code text on a line-by-line basis. For example, you can use the **AddFromString** method to add text to the module. **AddFromString** places the text just above the first procedure in the module or places the text at the end of the module if there are no procedures.

Use the **Parent** property to return the **VBComponent** object associated with a code module.

© 2017 Microsoft

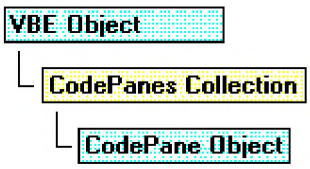
This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CodePane Object

See Also Example Properties Methods Events Specifics



Represents a code pane.

Remarks

Use the **CodePane** object to manipulate the position of visible text or the text selection displayed in the code pane.

You can use the **Show** method to make the code pane you specify visible. Use the **SetSelection** method to set the selection in a code pane and the **GetSelection** method to return the location of the selection in a code pane.

© 2017 Microsoft

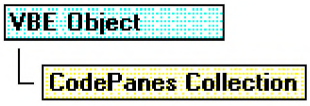
This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CodePanes Collection

See Also Example Properties Methods Events Specifics



Contains the active code panes in the **VBE** object.

Remarks

Use the **CodePanes** collection to access the open code panes in a project.

You can use the **Count** property to return the number of active code panes in a collection.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Collection Object

See Also Example Properties Methods Events Specifics

A **Collection** object is an ordered set of items that can be referred to as a unit.

Remarks

The **Collection** object provides a convenient way to refer to a related group of items as a single object. The items, or members, in a collection need only be related by the fact that they exist in the collection. Members of a collection don't have to share the same data type.

A collection can be created the same way other objects are created. For example:

```
Dim X As New Collection
```

Once a collection is created, members can be added using the **Add** method and removed using the **Remove** method. Specific members can be returned from the collection using the **Item** method, while the entire collection can be iterated using the **For Each...Next** statement.

© 2017 Microsoft

Visual Basic for Applications Reference

Collection Object Example

This example creates a **Collection** object (`MyClasses`), and then creates a dialog box in which users can add objects to the collection. To see how this works, choose the **Class Module** command from the **Insert** menu and declare a public variable called `InstanceName` at module level of Class1 (type **Public** `InstanceName`) to hold the names of each instance. Leave the default name as Class1. Copy and paste the following code into the General section of another module, and then start it with the statement `ClassNamer` in another procedure. (This example only works with host applications that support classes.)

```
Sub ClassNamer()
   Dim MyClasses As New Collection   ' Create a Collection object.
   Dim Num   ' Counter for individualizing keys.
   Dim Msg As String   ' Variable to hold prompt string.
   Dim TheName, MyObject, NameList   ' Variants to hold information.
   Do
      Dim Inst As New Class1   ' Create a new instance of Class1.
      Num = Num + 1   ' Increment Num, then get a name.
      Msg = "Please enter a name for this object." & Chr(13) _
       & "Press Cancel to see names in collection."
      TheName = InputBox(Msg, "Name the Collection Items")
      Inst.InstanceName = TheName   ' Put name in object instance.
      ' If user entered name, add it to the collection.
      If Inst.InstanceName <> "" Then
         ' Add the named object to the collection.
         MyClasses.Add item := Inst, key := CStr(Num)
      End If
      ' Clear the current reference in preparation for next one.
      Set Inst = Nothing
   Loop Until TheName = ""
   For Each MyObject In MyClasses   ' Create list of names.
      NameList = NameList & MyObject.InstanceName & Chr(13)
   Next MyObject
   ' Display the list of names in a message box.
   MsgBox NameList, , "Instance Names In MyClasses Collection"

   For Num = 1 To MyClasses.Count   ' Remove name from the collection.
      MyClasses.Remove 1   ' Since collections are reindexed
            ' automatically, remove the first
   Next      ' member on each iteration.
End Sub
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

Column Object

See Also Example Properties Methods Events

A **Column** object represents a column within a **DataGrid** control.

Remarks

You manipulate a column in a **DataGrid** control by using a **Column** object's methods and properties. With a **Column** object, you can modify attributes of the column header as well as the column itself.

Note A DataGrid object can contain only 32767 columns, as column indices are stored in integers.

To use a **Column** object, you can either use the **Columns** property of the **DataGrid** control directly or assign each column to a separate variable dimensioned as a **Column** object. The following demonstrates the latter:

```
Dim Col1, Col2 as Column
Set Col1 = DataGrid1.Columns(0)
Set Col2 = DataGrid1.Columns(1)
Col1.Caption = "Column 1"
Col2.Caption = "Column 2"
```

If often referring to the columns in a **DataGrid** control, you will increase performance by using the above method to assign values to columns rather than using the **Columns** property as in:

```
DataGrid1.Columns(0).Caption = "Column 1"
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ColumnHeader Object, ColumnHeaders Collection

See Also Example Properties Methods Events

- A **ColumnHeader** object is an item in a **ListView** control that contains heading text.
- A **ColumnHeaders** collection contains one or more **ColumnHeader** objects.

Syntax

listview.**ColumnHeaders**

listview.**ColumnHeaders**(*index*)

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

The **ColumnHeader** object, **ColumnHeaders** 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 an object collection. An integer would be the value of the **Index** property; a string would be the value of the **Key** property.

Remarks

You can view **ColumnHeader** objects in Report view only.

You can add **ColumnHeader** objects to a **ListView** control at both design time and run time.

With a **ColumnHeader** object, a user can:

- Click it to trigger the **ColumnClick** event and sort the items based on that data item.
- Grab the object's right border and drag it to adjust the width of the column.
- Hide **ColumnHeader** objects in Report view.

There is always one column in the **ListView** control, which is Column 1. This column contains the actual **ListItem** objects; not their subitems. The second column (Column 2) contains subitems. Therefore, you always have one more **ColumnHeader** object than subitems and the **ListItem** object's **SubItems** property is a 1-based array of size `ColumnHeaders.Count - 1`.

The number of **ColumnHeader** objects determines the number of subitems each **ListItem** object in the control can have. When you delete a **ColumnHeader** object, all of the subitems associated with the column are also deleted, and each **ListItem** object's subitem array shifts to update the indices of the **ColumnHeader**, causing the remaining column headers' **SubItemIndex** properties to change.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: DataGrid Control

Visual Studio 6.0

Columns Collection

See Also Example Properties Methods Events

The **Columns** collection contains all stored **Column** objects of a **DataGrid** control.

Syntax

Columns(*index***)**

Columns.Item(*index***)**

Remarks

You can use the properties and methods of the **Columns** collection to add and remove **Column** objects, count the number of columns in the **Columns** collection, and address individual columns of the **Columns** collection.

The **Columns** collection can be accessed through the **Columns** property of the **DataGrid** control.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

ComboBox Control

See Also Example Properties Methods Events

A **ComboBox** control combines the features of a **TextBox** control and a **ListBox** controlusers can enter information in the text box portion or select an item from the list box portion of the control.

Syntax

ComboBox

Remarks

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

Note A Scroll event will occur in a **ComboBox** control only when the contents of the dropdown portion of the **ComboBox** are scrolled, not each time the contents of the **ComboBox** change. For example, if the dropdown portion of a **ComboBox** contains five items and the top item is highlighted, a Scroll event will not occur until you press the down arrow six times (or the PGDN key once). After that, a Scroll event occurs for each press of the down arrow key. However, if you then press the up arrow key, a Scroll event will not occur until you press the up arrow key six times (or the PGUP key once). After that, each up arrow key press will result in a Scroll event.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ComboItem Object

See Also Example Properties Methods Events

A **ComboItem** object is an item in the list portion of an **ImageCombo** control. **ComboItem** objects can display text and/or pictures, and they can appear indented from other items in the list.

Syntax

object.ComboItem

The **ComboItem** object syntax has these parts:

Part	Description
object	An object expression that evaluates to an **ImageCombo** control.

Remarks

The **ComboItem** object and its corresponding **ComboItems** collection contain all the items that appear in the list portion of the **ImageCombo** control. Because list items are stored in a collection, each item can have multiple properties associated with it. This makes the task of assigning and managing the images associated with list items much easier.

Because the items in the list are objects in a collection, certain properties found in the standard combo box (such as **List**, **ListIndex**, and **ItemData**) are no longer required.

With a **ComboItem** object you can:

- Specify the text of an item.
- Assign a picture to be displayed next to the item by specifying an index into an **ImageList** control.
- Specify a different image for the item when it is selected from the list.
- Specify an indentation for an item.
- Assign the item a unique **Key** value that can be used to refer to it instead of its index in the collection.

The **ImageCombo** control does not initially contain any **ComboItem** objects. You must add using the **Add** method.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

ComboItems Collection

See Also Example Properties Methods Events

The **ComboItems** collection contains all the **ComboItem** objects in an **ImageCombo** control.

Syntax

object.**ComboItems**(*index*)

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

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

Remarks

The **ComboItems** collection is a 1-based collection of **ComboItem** objects.

The order in which **ComboItem** objects occur within the **ComboItems** collection is the same as their visible position within the list portion of the control. A **ComboItem** object's position within the collection is indicated by its **Index** property.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CommandBar Object

See Also Example Properties Methods Events

The **CommandBar** object contains other **CommandBar** objects, which can act as either buttons or menu commands.

Syntax

CommandBar

© 2017 Microsoft

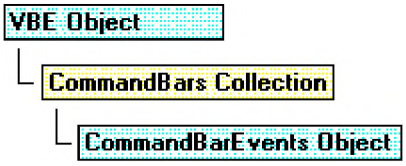
This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CommandBarEvents Object

See Also Example Properties Methods Events Specifics



Returned by the **CommandBarEvents** property. The **CommandBarEvents** object triggers an event when a control on the command bar is clicked.

Remarks

The **CommandBarEvents** object is returned by the **CommandBarEvents** property of the **Events** object. The object that is returned has one event in its interface, the Click event. You can handle this event using the **WithEvents** object declaration.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

CommandBars Collection

See Also Example Properties Methods Events Specifics

Contains all of the command bars in a project, including command bars that support shortcut menus.

Remarks

Use the **CommandBars** collection to enable add-ins to add command bars and controls, or to add controls to existing, built-in, command bars.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

CommandButton Control

See Also Example Properties Methods Events

Use a **CommandButton** control to begin, interrupt, or end a process. When chosen, a **CommandButton** appears pushed in and so is sometimes called a push button.

Syntax

CommandButton

Remarks

To display text on a **CommandButton** control, set its **Caption** property. A user can always choose a **CommandButton** by clicking it. To allow the user to choose it by pressing ENTER, set the **Default** property to **True**. To allow the user to choose the button by pressing ESC, set the **Cancel** property of the **CommandButton** to **True**.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: CommonDialog Control

Visual Studio 6.0

CommonDialog Control

See Also Example Properties Methods Events

The **CommonDialog** control provides a standard set of dialog boxes for operations such as opening and saving files, setting print options, and selecting colors and fonts. The control also has the ability to display help by running the Windows Help engine.

Syntax

CommonDialog

Remarks

The **CommonDialog** control provides an interface between Visual Basic and the routines in the Microsoft Windows dynamic-link library Commdlg.dll. To create a dialog box using this control, Commdlg.dll must be in your Microsoft Windows SYSTEM directory.

You use the **CommonDialog** control in your application by adding it to a form and setting its properties. The dialog displayed by the control is determined by the methods of the control. At run time, a dialog box is displayed or the help engine is executed, when the appropriate method is invoked; at design time, the **CommonDialog** control is displayed as an icon on a form. This icon can't be sized.

The **CommonDialog** control can display the following dialogs using the specified method.

Method	Dialog Displayed
ShowOpen	Show Open Dialog Box
ShowSave	Show Save As Dialog Box
ShowColor	Show Color Dialog Box
ShowFont	Show Font Dialog Box
ShowPrinter	Show Print or Print Options Dialog Box
ShowHelp	Invokes the Windows Help Engine

The **CommonDialog** control automatically provides context sensitive help on the interface of the dialog boxes by clicking:

- The What's This help button in the title bar then clicking the item for which you want more information.

- The right mouse button over the item for which you want more information then selecting the What's This command in the displayed context menu.

The operating system provides the text shown in the Windows 95 (or later) Help popup. You can also display a Help button on the dialog boxes with the **CommonDialog** control by setting the **Flags** property, however, you must provide the help topics in this situation.

Note There is no way to specify where a dialog box is displayed.

For More Information To see help topics for each dialog, click on See Also.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

ContainedControls Collection

See Also Example Properties Methods Events

A collection that allows access to the controls contained within a control that were added by the developer who uses the control.

Syntax

ContainedControls(*index***)**

The placeholder *index* represents an integer with a range from 0 to `ContainedControls.Count - 1`.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

ContainedVBControls Collection

See Also Example Properties Methods Events

The ContainedVBControls collection represents a collection of **VBControl** objects.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Control Object

See Also Example Properties Methods Events

The class name of all Visual Basic internal controls.

Syntax

Control

Remarks

You can dimension a variable as a **Control** object and reference it as you would a control on a form. The following demonstrates this:

```
Dim C as Control
Set C = Command1
```

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Controls Collection

See Also Example Properties Methods Events

A collection whose elements represent the controls on a component. The **Controls** collection has a **Count** property which specifies the number of controls in the collection, and an **Item** method which returns a member of the collection.

Syntax

object.**Controls.Count**

object.**Controls(***index***)**

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

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Index	An integer with a range from 0 to `Controls.Count - 1`.

Note If the component is a Visual Basic module, such as a **Form** or **UserControl**, you don't have to supply the object expression when writing code within the module. If the container is a compiled ActiveX control, such as a **ToolBar** control, however, you must always supply the object expression.

Remarks

The **Controls** collection enumerates loaded controls on a component and can be useful for iterating through them. For example, you might use it to change the **BackColor** property of all the **Label** controls on a **Form**.

The **Controls** collection identifies an intrinsic form-level variable named **Controls**. If you omit the optional *object* placeholder, you must include the **Controls** keyword. However, if you include *object*, you can omit the **Controls** keyword. For example, the following two lines of code have the same effect:

```
MyForm.Controls(6).Top = MyForm.Controls(5).Top + increment
MyForm(6).Top = MyForm(5).Top + increment
```

You can pass **Controls(***index***)** to a function whose argument is specified as a **Controls** class. You can also access members using their name. For example:

```
Controls("Command1").Top
```

You can use the **TypeOf** keyword with the **If** statement, or the **TypeName** function, to determine the type of a control in the **Controls** collection.

Note The **Controls** collection is not a member of the Visual Basic **Collection** class. It has a smaller set of properties and methods than a **Collection** object, and you cannot create instances of it.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

CoolBar Control

See Also Example Properties Methods Events

A **CoolBar** control contains a collection of **Band** objects used to create a configurable toolbar that is associated with a form.

Remarks

A **CoolBar** control is a container control that typically contains two or more **Bands** which may be resized and rearranged by the user. Each **Band** contains a single **Child** control.

© 2017 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

Coor Object

See Also Example Properties Methods Events

Describes a floating *x* and *y* coordinate pair for a chart.

Syntax

Coor

© 2017 Microsoft