Visual Studio 6.0

Visual Basic: MSChart Control

Backdrop Object

See Also Example Properties Methods Events

Represents a shadow or pattern behind a chart element.

Syntax

Backdrop

Remarks

For the **Plot** object's **Backdrop** object, set the **Style** property to **VtFillStyleBrush** to see the effects.

Visual Basic: Windows Controls

Visual Studio 6.0

Band Object

See Also Example Properties Methods Events

A **Band** object represents an individual band in the **Bands** collection of the **CoolBar** control.

Remarks

A band is a region within a CoolBar control which can contain a single child control, caption, and image. Each **Band** may be moved and resized independently by the user at run time.

At design time, use the Insert Band and Remove Band buttons on the Bands tab in the Properties Page of the **CoolBar** control to insert and remove **Band** objects from the **Bands** collection. At run time, you can add and remove **Band** objects by using the **Add** and **Remove** methods of the **Bands** collection.

Visual Basic: Windows Controls

Visual Studio 6.0

Bands Collection

See Also Example Properties Methods Events

A collection whose elements represent the bands on a **CoolBar** control. The **Bands** collection has a **Count** property which specifies the number of **Band** objects in the collection.

Syntax

object.**Bands.Count**

object.**Bands(***index***)**

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

Part	Description
object	An object expression that evaluates to a **CoolBar** control.
index	An integer with a range from 1 to `Bands.Count.`

Remarks

The **Bands** collection enumerates the bands on a CoolBar control. For example, you might use it to change the **BackColor** property of all the bands on a control.

Note The **Bands** collection is not a member of the Visual Basic **Collection** class; you cannot create instances of it.

Visual Basic Reference

Visual Studio 6.0

Binding Object

See Also Example Properties Methods Events

Represents the run time binding of a specific data consumer property to a data field of a data source.

Syntax

Binding

Remarks

Each **Binding** object in the **Binding** collection represents the combination of a data consumer (such as a control) and a single property of the consumer (such as the **Text** property), bound to a specific data source field.

Visual Basic Reference

Visual Studio 6.0

BindingCollection Object

See Also Example Properties Methods Events

A collection of **Binding** objects.

Syntax

BindingCollection

Remarks

The **BindingCollection** object allows you to bind any data provider to any data consumer. To bind a consumer to data provider, use the **Add** method to add a **Binding** object to the collection. Each **Binding** object represents the binding of a specific consumer to the **DataSource** supplied by the **BindingCollection** object.

Data sources that have no design time interface, such as a **Class** configured as a data source (by setting its **DataSourceBehavior** property to **VbDataSource**) or an **ADO Recordset** can be bound at run time using the **BindingObject** collection.

Use standard collection syntax to return or set properties of the members of the collection.

Visual Basic Reference

Binding Object, BindingCollection Object Example

The example uses the **BindingCollection** object to bind a data source to two **TextBox** controls. The example first opens an ADODB recordset object, then sets the **DataSource** property of the **BindingCollection** to the recordset. The code then adds two **Binding** objects to the collection, thereby binding two **TextBox** controls to different fields of the recordset.

To try the example, in the **References** dialog box set a reference to the **Microsoft Data Binding Collection**. In the same dialog box, set a reference to the **Microsoft ActiveX Data Objects Library**. Draw two **TextBox** controls on a Form, and paste the code into the Declarations section. Press F5, and click the form to advance through the recordset.

```
Option Explicit
Private colBndNwind As New BindingCollection
Private rsNwind As New ADODB.Recordset
Private cn As New ADODB.Connection

Private Sub Form_Load()

   ' Set the Connection object parameters.
   With cn
      ' The following connection may or may not work on your computer.
      ' Alter it to find the Nwind.mdb file, which is included with
      ' Visual Basic.
      .Provider = "Microsoft.Jet.OLEDB.3.51"
      .Open "C:\Program Files\DevStudio\VB\Nwind.mdb"
   End With

   ' Open the recordset object.
   rsNwind.Open "Select * From Products", cn

   ' Set the DataSource of the Bindings collection to the recordset.
   Set colBndNwind.DataSource = rsNwind

   ' Add to the Bindings collection.
   With colBndNwind
      .Add Text1, "Text", "ProductName", , "product"
      .Add Text2, "Text", "SupplierID", , "ID"
   End With

   ' Print the properties of the objects in the collection.
   Dim bndObj As Binding
   For Each bndObj In colBndNwind
      Debug.Print "DataField", "PropertyName", "Key"
      Debug.Print bndObj.DataField, bndObj.PropertyName, bndObj.Key
      Debug.Print
   Next
End Sub

Private Sub Form_Click()
   ' Move to the next record by clicking the form.
   rsNwind.MoveNext
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Brush Object

See Also Example Properties Methods Events

The fill type used to display a chart element.

Syntax

Brush

Visual Basic: MSChart Control

Brush Object Example

The following example sets a bold vertical line pattern for the chart backdrop using the **Brush** object.

```
Private Sub Command1_Click()
   ' Sets Backdrop to Fill - Brush Style.
   MSChart1.Backdrop.Fill.Style = VtFillStyleBrush
   ' Sets a pattern for the chart backdrop using the
   ' Brush object.
   With MSChart1.Backdrop.Fill.Brush
      .Style = VtBrushStylePattern
      .Index = VtBrushPatternBoldVertical
   ' Sets Pattern to Bold Vertical lines.
      .FillColor.Set 255, 0, 0   ' Fill Color = Red.
      .PatternColor.Set 0, 0, 255   ' Pattern Color =
                                 ' Blue.
   End With
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

Button Object

See Also Example Properties Methods Events

A **Button** object represents an individual button in the **Buttons** collection of a **Toolbar** control.

Remarks

For each **Button** object, you can add text or a bitmap image, or both, from an **ImageList** control, and set properties to change its state and style.

At design time, use the Insert Button and Remove Button buttons on the Buttons tab in the Properties Page of the **Toolbar** control to insert and remove **Button** objects from the **Buttons** collection. At run time, you can also add **Button** objects by using the **Add** method of the **Buttons** collection.

At design time and run time, you can set the **Caption**, **Image**, **Value**, **MixedState**, and **ToolTipText** properties to change the appearance of each **Button** object.

Whenever a button is clicked on the **Toolbar** control, the ButtonClick event is called with the selected **Button** object passed in as a parameter. To cause some action to occur when a button is clicked, use the **Index** or **Key** properties in a **Select Case** statement as in the following code:

```
Select Case Button.Key
  Case Is = "open" ' Open file.
  ' Add code to Open a file here
  Case Is = "save" ' Save file.
  ' Add code to Save a file here
  Case Else
  ' If any other button is pressed
End Select
```

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonMenu Object

See Also Example Properties Methods Events

The **ButtonMenu** object represents a menu that drops down from the **Toolbar** control's **Button** object.

Syntax

ButtonMenu

Remarks

Button menus only appear when the **Button** object's **Style** property is set to **tbrDropdown**.

Visual Basic: Windows Controls

ButtonMenu Object, ButtonMenuClick Event Example

The example adds five **Button** objects to a **Toolbar** control and also adds two **ButtonMenu** objects to each **Button** object. When a ButtonMenu object is clicked, the ButtonMenuClick event is used to determine its behavior. To try the example, place a Toolbar control on a form and paste the code into the Declarations section of the code module.

```
Option Explicit

Private Sub Form_Load()
   Dim i As Integer
   Dim btn As Button

   ' Add five Button objects to the Toolbar control.
   For i = 1 To 5
      Set btn = Toolbar1.Buttons.Add(Caption:= i, Style:= tbrDropDown)
      ' Add two ButtonMenu objects to each Button.
         btn.ButtonMenus.Add Text:="Help"
         btn.ButtonMenus.Add Text:="Options"
   Next i
End Sub

Private Sub Toolbar1_ButtonMenuClick(ByVal ButtonMenu As ComctlLib.ButtonMenu)
   Select Case ButtonMenu.Index
   Case 1
      MsgBox "Press the button."
   Case 2
      MsgBox "Offer some option"
   End Select
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

ButtonMenus Collection

See Also Example Properties Methods Events

A collection of **ButtonMenu** objects.

Syntax

ButtonMenus

Remarks

Use the **ButtonMenus** property to return a reference to the collection.

Visual Basic: Windows Controls

Visual Studio 6.0

Buttons Collection

See Also Example Properties Methods Events

A **Buttons** collection is a collection of **Button** objects for a **Toolbar** control.

Syntax

toolbar.**Buttons**(*index*)

toolbar.**Buttons.Item**(*index*)

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

Part	Description
toolbar	An object expression that evaluates to a **Toolbar** control.
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 **Buttons** collection is a 1-based collection, which means the collection's **Index** property begins with the number 1 (versus 0 in a 0-based collection).

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

```
Dim btnX As Button
   ' Reference by index number.
Set btnX = Toolbar1.Buttons(3)
   ' Or reference by unique key.
Set btnX = Toolbar1.Buttons("third") ' Assuming Key is "third."
   ' Or use Item method.
Set btnX = Toolbar1.Buttons.Item(3)
```