Visual Basic Extensibility Reference

Visual Studio 6.0

AddIn Object

See Also Example Properties Methods Events

The **AddIn** object provides information about an add-in to other add-ins.

Syntax

object.**AddIn**

Remarks

An **AddIn** object is created for every add-in that appears in the Vbaddin.ini file.

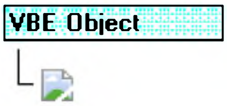
This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

AddIns Collection

See Also Example Properties Methods Events



Returns a collection of add-ins listed in the Vbaddin.ini file.

Syntax

object.**AddIns**

Remarks

The **AddIns** collection is accessed through the **VBE** object. Every add-in listed in the Add-In Manager in an instance of Visual Basic has an object in the **AddIns** collection.

This collection replaces the **ExternalObjects** collection used in Visual Basic version 4.0.

This documentation is archived and is not being maintained.

Visual Basic: ADO Data Control

Visual Studio 6.0

ADO Data Control

See Also Example Properties Methods Events

The **ADO Data Control** is similar to the intrinsic **Data** control and the **Remote Data Control** (RDC). The **ADO Data Control** allows you to quickly create a connection to a database using **Microsoft ActiveX Data Objects** (ADO).

Remarks

At design time, you can create a connection by setting the **ConnectionString** property to a valid connection string, then set the **RecordSource** property to a statement appropriate to the database manager. You can also set the **ConnectionString** property to the name of a file that defines a connection; the file is generated by a **Data Link** dialog box, which appears when you click **ConnectionString** on the Properties window and then select or create a Data Link file.

Connect the **ADO Data Control** to a data-bound control such as the **DataGrid**, **DataCombo**, or **DataList** control by setting the **DataSource** property to the **ADO Data Control**.

At run time, you can dynamically set the **ConnectionString** and **RecordSource** properties to change the database. Alternatively, you can set the **Recordset** property directly to a previously-opened recordset.

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

AmbientProperties Object

See Also Example Properties Methods Events

An **AmbientProperties** object holds ambient information from a container to suggest behavior to controls contained within the container.

Remarks

Containers provide ambient properties in order to suggest behavior to controls. As an example, **BackColor** is one of the standard ambient properties; the container is suggesting what the well-behaved control should set its **BackColor** property to.

The **AmbientProperties** objects properties are the ambient properties of the container. These properties are read-only.

Some ambient properties are standard, while others are specific to certain containers. A control may access non-standard ambient properties, but this will make the control container-specific. The control should handle the case where an ambient property is not present in the current container.

When the control is compiled, Visual Basic has no way of knowing what container-specific ambient properties may be available when the control is run; therefore references to container-specific ambient properties will always be late bound.

The **AmbientProperties** object is not available when the Initialize event is raised; but is available when the InitProperties event or ReadProperties event is raised.

The **AmbientProperties** object has several standard properties:

- The **BackColor** property, a Color that contains the suggested interior color of the contained control. The Visual Basic supplied default if the container does not support this property is 0x80000005: the system color for a window background.
- The **DisplayAsDefault** property, a Boolean that specifies if the control is the default control. The Visual Basic supplied default if the container does not support this property is **False**.
- The **DisplayName** property, a String containing the name that the control should display for itself. The Visual Basic supplied default if the container does not support this property is an empty string: "".
- The **Font** property, a **Font** object that contains the suggested font information of the contained control. The Visual Basic supplied default if the container does not support this property is MS Sans Serif 8.
- The **ForeColor** property, a Color that contains the suggested foreground color of the contained control. The Visual Basic supplied default if the container does not support this property is 0x80000008: the system color for window text.
- The **LocaleID** property, a Long that specifies the language and country of the user. The Visual Basic supplied default if the container does not support this property is the current system locale ID.
- The **MessageReflect** property, a Boolean that specifies if the container supports message reflection. The Visual Basic supplied default if the container does not support this property is **False**.

- The **Palette** property, a **Picture** object whose palette specifies the suggested palette for the contained control.
- The **RightToLeft** property, a Boolean that indicates the text display direction and control visual appearance on a bidirectional system. The Visual Basic supplied default if the container does not support this property is **False**.
- The **ScaleUnits** property, a String containing the name of the coordinate units being used by the container. The Visual Basic supplied default if the container does not support this property is an empty string: "".
- The **ShowGrabHandles** property, a Boolean that specifies if the container handles the showing of grab handles. The Visual Basic supplied default if the container does not support this property is **True**.
- The **ShowHatching** property, a Boolean that specifies if the container handles the showing of hatching. The Visual Basic supplied default if the container does not support this property is **True**.
- The **SupportsMnemonics** property, a Boolean that specifies if the container handles access keys for the control. The Visual Basic supplied default if the container does not support this property is **False**.
- The **TextAlign** property, an enumeration that specifies how text is to be aligned. The Visual Basic supplied default if the container does not support this property is **0 - General Align**.
- The **UserMode** property, a Boolean that specifies if the environment is in design mode or end user mode. The Visual Basic supplied default if the container does not support this property is **True**.
- The **UIDead** property, a Boolean that specifies if the User Interface is unresponsive. The Visual Basic supplied default if the container does not support this property is **False**.

Note When you access the Font property of the AmbientProperties object, you do not get a reference to the container's font. Rather, you get a copy (clone) of the font.

The reason for this is that AmbientProperties.Font is commonly used to initialize a control's font whenever an instance of the control is added to a container. If a reference to the font itself were supplied, then changing the control's font would change the container's font as well.

If for some reason your control requires a reference to the container's font, you can obtain it by accessing the container through the Parent property of the UserControl object.

This documentation is archived and is not being maintained.

Visual Basic: Windows Controls

Visual Studio 6.0

Animation Control

See Also Example Properties Methods Events

The **Animation** control allows you to create buttons which display animations, such as .avi files, when clicked. The control can play only AVI files that have no sound. In addition, the Animation control can display only uncompressed .avi files or .avi files that have been compressed using Run-Length Encoding (RLE).

Remarks

If you attempt to load an .avi file that includes sound data or that is in a format not supported by the control, an error (error 35752) is returned.

An example of this control is the file copy progress bar in Windows 95/98, which uses an **Animation** control. Pieces of paper "fly" from one folder to another while the copy operation executes.

Visual Basic: Windows Controls

Open Method (Animation Control) Example

The following example opens an .avi file by using the Open Dialog, and begins playing it automatically. To try the example, place an **Animation** control and a **CommonDialog** control on a form, and paste the code into the form's Declarations section. Run the example, and choose an .avi file to open.

```
Private Sub Animation1_Click ()
   With CommonDialog1
      .Filter = "avi (*.avi)|*.avi"
      .ShowOpen
   End With
   With Animation1
      .Autoplay = True
      .Open CommonDialog1.Filename
   End With
End Sub
```

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

App Object

See Also Example Properties Methods Events

The **App** object is a global object accessed with the **App** keyword. It determines or specifies information about the application's title, version information, the path and name of its executable file and Help files, and whether or not a previous instance of the application is running.

Syntax

App

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

AsyncProperty Object

See Also Example Properties Methods Events

The **AsyncProperty** object is passed in to the AsyncReadComplete event and contains the results of the **AsyncRead** method.

Visual Studio 6.0

Visual Basic: MSChart Control

Axis Object

See Also Example Properties Methods Events

Represents an axis on a chart.

Syntax

Axis

Remarks

Three axes are available: *x*, *y*, and *z*. The *z* axis is visible only when the chart is a 3D chart.

Visual Basic: MSChart Control

Axis Object Example

The following example reads the number of label levels present on the *x* axis using the *x* **Axis** object.

```
Private Sub Command1_Click()
   Dim XAxis As Object
   Dim NumberOfLevels As Integer
   ' Read the number of label level present on the X
   ' Axis.
   Set XAxis = MSChart1.Plot.Axis(VtChAxisIdX, 1)
   NumberOfLevels = XAxis.LabelLevelCount
   MsgBox "Number of Label Levels = " _
   & Str(NumberOfLevels)
End Sub
```

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

AxisGrid Object

See Also Example Properties Methods Events

Represents the planar area surrounding a chart axis.

Syntax

AxisGrid

Visual Basic: MSChart Control

AxisGrid Object Example

The following example changes the appearance of the y **AxisGrid** line style.

```
Private Sub Command1_Click()
   With MSChart1.Plot.Axis(VtChAxisIdY)
      .CategoryScale.Auto = False
      .ValueScale.MajorDivision = 10
      .ValueScale.MinorDivision = 5
      .AxisGrid.MajorPen.Style = VtPenStyleSolid
      .AxisGrid.MajorPen.VtColor.Green = 255
      .AxisGrid.MajorPen.Width = 2
      .AxisGrid.MinorPen.Style = VtPenStyleDashed
      .AxisGrid.MinorPen.VtColor.Red = 255
      .AxisGrid.MinorPen.Width = 1
      .AxisGrid.MinorPen.Cap = VtPenCapRound
   End With
End Sub
```

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

AxisScale Object

See Also Example Properties Methods Events

Controls how chart values are plotted on an axis.

Syntax

AxisScale

Visual Basic: MSChart Control

AxisScale Example

The following examples sets the *x* and *y* axes to either a percent scale or a logarithmic for a two-dimensional line chart. To try the example, draw an **MSChart** and two **CommandButton** controls on a form. Paste the code into the **Form** object's code module and run the project.

```
Private Sub Command1_Click()
   ' Change both x and y axes to Percent scale for 2D
   ' Line chart.
   Dim axisID As VtChAxisId
   MSChart1.ChartType = VtChChartType2dLine
   For AxisId = VtChAxisIdX To VtChAxisIdY
      With MSChart1.Plot.Axis(AxisId).AxisScale
         .Type = VtChScaleTypePercent
         .PercentBasis = VtChPercentAxisBasisSumChart
      End With
   Next
End Sub

Private Sub Command2_Click()
   Dim axisID As VtChAxisId
   ' Change both x and y axes to Logarithmic scale for 2D
   ' Line chart.
   MSChart1.ChartType = VtChChartType2dLine
   For axisID = VtChAxisIdX To VtChAxisIdY
      With MSChart1.Plot.Axis(axisID).AxisScale
         .Type = VtChScaleTypeLogarithmic
         .LogBase = 12
      End With
   Next
End Sub
```

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

AxisTitle Object

See Also Example Properties Methods Events

Represents an axis title on a chart.

Syntax

AxisTitle

Visual Basic: MSChart Control

AxisTitle Object Example

The following example makes the axis title visible for all axes of a three-dimensional chart.

```
Private Sub Command1_Click()
   ' Makes Axis title visible for all axes of a
   ' 3D chart.
   Dim axisID As VtChAxisId
   MSChart1.chartType = VtChChartType3dBar
   For axisId = VtChAxisIdX To VtChAxisIdZ
      With MSChart1.Plot.axis(axisId, 1)
         .AxisTitle.VtFont.Size = 14
         .AxisTitle.Visible = True
         Select Case axisId
            Case 0
               .AxisTitle.text = "X Axis Title"
            Case 1
                .AxisTitle.text = "Y Axis Title"
            Case 2
               .AxisTitle.text = "2nd Y Axis Title"
            Case 3
               .AxisTitle.text = "Z Axis Title"
         End Select
      End With
   Next
End Sub
```