Visual Basic for Applications Reference

Visual Studio 6.0

File Object

See Also Example Properties Methods Events Specifics

Description

Provides access to all the properties of a file.

Remarks

The following code illustrates how to obtain a **File** object and how to view one of its properties.

```
Sub ShowFileInfo(filespec)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(filespec)
    s = f.DateCreated
    MsgBox s
End Sub
```

Visual Basic Extensibility Reference

Visual Studio 6.0

FileControlEvents Object

See Also Example Properties Methods Events

Represents all events supplied by Visual Basic which support file control.

Syntax

FileControlEvents

Remarks

The **FileControlEvents** object replaces the **FileControl** object in Visual Basic version 4.0. It works the same as before, only its events have been changed to allow multiple project support.

Visual Basic Reference

Visual Studio 6.0

FileListBox Control

See Also Example Properties Methods Events

A **FileListBox** control locates and lists files in the directory specified by the **Path** property at run time. Use this control to display a list of files selected by file type. You can create dialog boxes in your application that, for example, enable the user to select a file or group of files.

Syntax

FileListBox

Remarks

Set the **List**, **ListCount**, and **ListIndex** properties to enable a user to access items in the list. If you also display the **DirListBox** and **DriveListBox** controls, you can write code to synchronize them with the **FileListBox** control and with each other.

Visual Basic for Applications Reference

Visual Studio 6.0

Files Collection

See Also Example Properties Methods Events Specifics

Description

Collection of all **File** objects within a folder.

Remarks

The following code illustrates how to get a **Files** collection and iterate the collection using the **For Each...Next** statement:

```
Sub ShowFolderList(folderspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    Set fc = f.Files
    For Each f1 in fc
        s = s & f1.name
        s = s & vbCrLf
    Next
    MsgBox s
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

FileSystemObject Object

See Also Example Properties Methods Events Specifics

Description

Provides access to a computer's file system.

Syntax

Scripting.FileSystemObject

Remarks

The following code illustrates how the **FileSystemObject** is used to return a **TextStream** object that can be read from or written to:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.CreateTextFile("c:\testfile.txt", True)
a.WriteLine("This is a test.")
a.Close
```

In the code shown above, the **CreateObject** function returns the **FileSystemObject** (`fs`). The **CreateTextFile** method then creates the file as a **TextStream** object (a), and the **WriteLine** method writes a line of text to the created text file. The **Close** method flushes the buffer and closes the file.

Visual Studio 6.0

Visual Basic: MSChart Control

Fill Object

See Also Example Properties Methods Events

Describes the type and appearance of an object's backdrop in a chart.

Syntax

Fill

Visual Basic: MSChart Control

Fill Object Example

The following example sets a gradient backdrop for a chart using the **Fill** object.

```
Private Sub Command1_Click()
   With MSChart1.backdrop.Fill
      ' Set a brush pattern backdrop.
      .Style = VtFillStyleBrush
      .Brush.Style = VtBrushPattern50Percent
   End With
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

FlatScrollBar Control

See Also Example Properties Methods Events

The **FlatScrollBar** control is a mouse-sensitive version of the standard Windows scroll bar that offers two-dimensional formatting options. It can also replace the standard Windows three-dimensional scroll bar. The **FlatScrollBar** provides increased interactivity when using the scroll arrows and the scroll box.

Syntax

FlatScrollBar

Remarks

The **FlatScrollBar** provides you with three formatting options.

- A two-dimensional appearance that duplicates the look of the scroll bar found in Internet Explorer 4.0. The scroll arrows and the scroll bar thumb are mouse sensitive; they change color in response to the mouse pointer passing over them.
- A two-dimensional appearance that becomes three-dimensional in response to the mouse pointer. The scroll arrows and thumb are beveled when the pointer passes over them. This reproduces the look of the scrollbar seen in Microsoft Encarta Encyclopedia.
- An appearance identical to that of a standard three-dimensional Windows scroll bar. Three-dimensional mode does not have mouse-sensitive features.

With the **FlatScrollBar** you can disable either of the scroll arrows, this provides additional feedback to the user as an indication to scroll in a particular direction based on other factors in the program.

The **FlatScrollBar** can serve as either a horizontal or a vertical scrollbar depending on you set its **Orientation** property.

Distribution Note The **FlatScrollBar** control is part of a group of ActiveX controls that are found in the MSCOMCT2.OCX file. To use the **FlatScrollBar** control in your application, you must add the MSCOMCT2.OCX file to the project. When distributing your application, install the MSCOMCT2.OCX file in the user's Microsoft Windows System or System32 directory. For more information on how to add an ActiveX control to a project, see "Adding Controls to a Project" in the *Programmer's Guide*.

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Folder Object

See Also Example Properties Methods Events Specifics

Description

Provides access to all the properties of a folder.

Remarks

The following code illustrates how to obtain a **Folder** object and how to return one of its properties:

```
Sub ShowFolderInfo(folderspec)
    Dim fs, f, s,
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    s = f.DateCreated
    MsgBox s
End Sub
```

Visual Basic for Applications Reference

Visual Studio 6.0

Folders Collection

See Also Example Properties Methods Events Specifics

Description

Collection of all **Folder** objects contained within a **Folder** object.

Remarks

The following code illustrates how to get a **Folders** collection and how to iterate the collection using the **For Each...Next** statement:

```
Sub ShowFolderList(folderspec)
    Dim fs, f, f1, fc, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    Set fc = f.SubFolders
    For Each f1 in fc
        s = s & f1.name
        s = s &  vbCrLf
    Next
    MsgBox s
End Sub
```

Visual Studio 6.0

Visual Basic: MSChart Control

Footnote Object

See Also Example Properties Methods Events

Descriptive text that appears beneath a chart.

Syntax

Footnote

Visual Basic: MSChart Control

Footnote Object Example

The following example sets the footnote location, text and color for a chart.

```
Private Sub Command1_Click()

   With MSChart1.Footnote
      ' Make Footnote Visible.
      .Location.Visible = True
      .Location.LocationType = _
      VtChLocationTypeBottomLeft

      ' Set Footnote properties.
      .text = "Chart Footnote"
      .VtFont.VtColor.Set 255, 0, 0
   End With

End Sub
```

Visual Basic Reference

Visual Studio 6.0

Form Object, Forms Collection

See Also Example Properties Methods Events

A **Form** object is a window or dialog box that makes up part of an application's user interface.

A **Forms** collection is a collection whose elements represent each loaded form in an application. The collection includes the application's MDI form, MDI child forms, and non-MDI forms. The **Forms** collection has a single property, **Count**, that specifies the number of elements in the collection.

Syntax

Form

Forms(*index***)**

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

Remarks

You can use the **Forms** collection to iterate through all loaded forms in an application. It identifies an intrinsic global variable named **Forms**. You can pass **Forms**(*index*) to a function. whose argument is specified as a **Forms** class.

Forms have properties that determine aspects of their appearance, such as position, size, and color; and aspects of their behavior, such as whether or not they are resizable.

Forms can also respond to events initiated by a user or triggered by the system. For example, you could write code in a form's Click event procedure that would enable the user to change the color of a form by clicking it.

In addition to properties and events, you can use methods to manipulate forms using code. For example, you can use the **Move** method to change a form's location and size.

A special kind of form, the MDI form, can contain other forms called MDI child forms. An MDI form is created with the MDI Form command on the Insert menu; an MDI child form is created by choosing New Form from the File menu and then setting the **MDIChild** property to **True**.

You can create multiple instances of forms in code by using the **New** keyword in **Dim**, **Set**, and **Static** statements.

When designing forms, set the **BorderStyle** property to define a form's border, and set the **Caption** property to put text in the title bar. In code, you can use the **Hide** and **Show** methods to make forms invisible or visible at run time.

Note Setting **BorderStyle** to 0 removes the border. If you want your form to have a border without the title bar, Control-menu box, Maximize button, and Minimize button, delete any text from the form's **Caption** property, and set the form's **ControlBox**, **MaxButton**, and **MinButton** properties to **False**.

Form is an Object data type. You can declare variables as type **Form** before setting them to an instance of a type of form that was declared at design time. Similarly, you can pass an argument to a procedure as type **Form**.

Forms also can act as sources in a DDEconversation, with a **Label**, **PictureBox**, or **TextBox** control furnishing the data.

You can access the collection of controls on a **Form** using the **Controls** collection. For example, to hide all the controls on an **Form** you can use code similar to the following:

```
For Each Control in Form1.Controls
   Control.Visible = False
Next Control
```

Visual Basic Reference

Forms Collection Example

This example fills a list box with the captions of all the currently loaded forms.

```
Private Sub Form_Activate ()
   Dim I   ' Declare variable.
   ' Refill list (in case an instance was added or removed).
   lstForms.Clear   ' Clear list box.
   For I = 0 To Forms.Count - 1
      lstForms.AddItem Forms(I).Caption
   Next I
End Sub
```

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Frame Control

See Also Example Properties Methods Events

A **Frame** control provides an identifiable grouping for controls. You can also use a **Frame** to subdivide a form functionallyfor example, to separate groups of **OptionButton** controls.

Syntax

Frame

Remarks

To group controls, first draw the **Frame** control, and then draw the controls inside the **Frame**. This enables you to move the **Frame** and the controls it contains together. If you draw a control outside the **Frame** and then try to move it inside, the control will be on top of the **Frame** and you'll have to move the **Frame** and controls separately.

To select multiple controls in a **Frame**, hold down the CTRL key while using the mouse to draw a box around the controls.

Visual Basic: MSChart Control

Frame Object Example

The following example sets a blue, double-line frame on a chart backdrop.

```
Private Sub Command1_Click()
   With MSChart1.backdrop.Frame
      .Style = VtFrameStyleDoubleLine
      .Width = 2
      .FrameColor.Set 0, 0, 255   ' Blue frame.
      .SpaceColor.Set 255, 0, 0   ' Red spacing.
   End With
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Function Control (Data Report Designer)

See Also Example Properties Methods Events

A Function control displays figures calculated at run time using various built-in functions.

Syntax

rptFunction

Remarks

The functions contained by the Function control include:

Function	**Description**
Sum	Sums the values of a field.
Min	Displays the minimum value of a field.
Max	Displays the maximum value of a field.
Average	Displays the average of values.
Standard Deviation	Displays the standard deviation for a column of figures.
Standard Error	Displays the standard error for a column of figures.
Value Count	Displays the number of fields containing non-null values.
Row Count	Displays the number of rows in a section of the report.

The Function control can only be placed in a group footer section of a Data Report designer.

The **Data Environment** also features an aggregate field that has some similar functions as the Function control.