This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Err Object

See Also Example Properties Methods Events Specifics

Contains information about run-time errors.

Remarks

The properties of the **Err** object are set by the generator of an error Visual Basic, an object, or the programmer.

The default property of the **Err** object is **Number**. Because the default property can be represented by the object name **Err**, earlier code written using the **Err** function or **Err** statement doesn't have to be modified.

When a run-time error occurs, the properties of the **Err** object are filled with information that uniquely identifies the error and information that can be used to handle it. To generate a run-time error in your code, use the **Raise** method.

The **Err** object's properties are reset to zero or zero-length strings ("") after an **Exit Sub**, **Exit Function**, **Exit Property** or **Resume Next** statement within an error-handling routine. Using any form of the **Resume** statement outside of an error-handling routine will not reset the **Err** object's properties. The **Clear** method can be used to explicitly reset **Err**.

Use the **Raise** method, rather than the **Error** statement, to generate run-time errors for system errors and class modules. Using the **Raise** method in other code depends on the richness of the information you want to return.

The **Err** object is an intrinsic object with global scope. There is no need to create an instance of it in your code.

Visual Basic for Applications Reference

Err Object Example

This example uses the properties of the **Err** object in constructing an error-message dialog box. Note that if you use the **Clear** method first, when you generate a Visual Basic error with the **Raise** method, Visual Basic's default values become the properties of the **Err** object.

```
Dim Msg
' If an error occurs, construct an error message
On Error Resume Next   ' Defer error handling.
Err.Clear
Err.Raise 6   ' Generate an "Overflow" error.
' Check for error, then show message.
If Err.Number <> 0 Then
   Msg = "Error # " & Str(Err.Number) & " was generated by " _
         & Err.Source & Chr(13) & Err.Description
   MsgBox Msg, , "Error", Err.Helpfile, Err.HelpContext
End If
```

Visual Basic Reference

Visual Studio 6.0

Error Object (Data Report Designer)

See Also Example Properties Methods Events

An Error object contains details about run time errors.

Syntax

RptError

Visual Basic Reference

Visual Studio 6.0

EventInfo Object

See Also Example Properties Methods Events

Represents event information raised by a control assigned to a **VBControlExtender** object variable.

Syntax

EventInfo

Remarks

The **EventInfo** object is available in the ObjectEvent event, which is an event of the **VBControlExtender** object. In common practice, a control that is dynamically added to the **Controls** collection using the **Add** method will be assigned to an object variable of the type **VBControlExtender**. The ObjectEvent event can then be used to trap all events raised by the control, and the **EventInfo** object specifically represents any parameters passed by the raised events.

The **EventInfo** object has two properties: the **Name** property returns the name of the raised event; the **EventParameters** property returns a reference to the **EventParameters** collection that allows you to return values of all event parameters.

Visual Basic Reference

Visual Studio 6.0

EventParameter Object

See Also Example Properties Methods Events

Represents a control event parameter.

Syntax

EventParameter

Remarks

The **EventParameter** object is used to evaluate and return the values contained by an event of a control that has been dynamically added to the **Controls** collection.

The **EventParameters** collection is part of the **EventInfo** object.

The **EventParameter** object has two properties: **Name** and **Value**. The **Name** property returns the name of a parameter. You can then evaluate and set the **Value** property as appropriate.

Visual Basic Reference

ObjectEvent Event, EventParameter Object Examples

The first example below uses the ObjectEvent event to print all parameter names and values.

```
Option Explicit
Private WithEvents extObj As VBControlExtender

' Code to set the object variable to a user control not shown.

Private Sub extObj_ObjectEvent(Info As EventInfo)
   Dim p As EventParameter
   Debug.Print Info.Name

   For Each p In Info.EventParameters
      Debug.Print p.Name, p.Value
   Next
End Sub
```

The second example allows you to check for a generic event raised by the control.

```
Private Sub extObj_ObjectEvent(Info As EventInfo)
   Dim p As EventParameter
   Select Case Info.Name
   Case "UserName"
      ' Check User name value.
      MsgBox Info.EventParameters("UserName").Value
   ' Other cases now shown
   Case Else ' Unknown Event
      ' Handle unknown events here.
   End Select
End Sub
```

Visual Basic Extensibility Reference

Visual Studio 6.0

Events Object

See Also Example Properties Methods Events Specifics

Supplies properties that enable add-ins to connect to all events in Visual Basic for Applications.

Remarks

The **Events** object provides properties that return event source objects. Use the properties to return event source objects that notify you of changes in the Visual Basic for Applications environment.

The properties of the **Events** object return objects of the same type as the property name. For example, the **CommandBarEvents** property returns the **CommandBarEvents** object.

Visual Basic Reference

Visual Studio 6.0

ExportFormat Object

See Also Example Properties Methods Events

Using the **ExportFormat** object, you can programmatically determine various attributes of text exported from a Data Report.

Syntax

ExportFormat

Remarks

When exporting a report using the **ExportReport** method, you must specify an **ExportFormat** object.

Visual Basic Reference

Visual Studio 6.0

ExportFormats Collection

See Also Example Properties Methods Events

A collection of **ExportFormat** objects.

Syntax

ExportFormats

Remarks

Use the **ExportFormat** object in conjunction with the **ExportReport** method to create customized exported reports. When invoking the **ExportReport** method, specify which **ExportFormat** object to use, as shown below:

```
' Use the ExportFormat object named MyReport.
DataReport1.ExportReport "MyReport"
```

Each object in the collection can represent a different template. To create a template use the **Template** property.

When the end user invokes the **ExportReport** method, the **FileFormatString** property determines what text is displayed in the Save As Type box.

Visual Basic Reference

Add Method, ExportFormats Collection, Template Property Example

This example creates a template, adds an **ExportFormat** object to the **ExportFormats** collection using the new template, and exports the report using the **ExportFormat** object.

```
Private Sub ExportDailyReport()
   DataReport1.Title = "Daily Report" ' This title appears in the report.
   Dim strTemplate As String
   ' Create the template.
   strTemplate = _
   "<HTML>" & vbCrLf & _
   "<HEAD>" & vbCrLf & _
   "<TITLE>" & "MyCompany: " & rptTagTitle & _
   "</TITLE>" & vbCrLf & _
   "<BODY>" & vbCrLf & _
   rptTagBody & vbCrLf & _
   "<BODY>" & vbCrLf & _
   "</HTML>"

   ' Add a new ExportFormat object using the template.
   DataReport1.ExportFormats.Add _
   Key:="DailyReport", _
   FormatType:=rptFmtHTML, _
   FileFormatString:="Daily Report (*.htm)", _
   FileFilter:="*.HTM", _
   Template:=strTemplate

   ' Export the report using the new ExportFormat object.
   DataReport1.ExportReport _
   FormatIndexOrKey:="DailyReport", _
   FileName:="C:\Temp\DailyRpt", _
   Overwrite:=True, _
   ShowDialog:=False, _
   Range:=rptRangeFromTo, _
   Pagefrom:=1, _
   Pageto:=10
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Extender Object

See Also Example Properties Methods Events

An **Extender** object holds properties of the control that are actually controlled by the container of the control rather than by the control itself.

Syntax

Extender

Remarks

Some properties of a control are provided by the container rather than the control; these are extender properties. Examples of extender properties are: **Name**, **Tag** and **Left**. The control still needs to know what the value of these extender properties are, and sometimes needs to be able to change an extender property; the **Extender** object gives the control access to these properties.

Some extender properties are standard, while others are specific to certain containers. A control may access non-standard extender properties, but this will make the control container-specific. If the control makes use of an extender property, the control should handle the case where the extender property is not supported by the current container.

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

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

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

- The **Name** property, a read only String that contains the user-defined name of the control.
- The **Visible** property, a read/write Boolean that specifies if the control is visible or not.
- The **Parent** property, a read only object that represents the container of the control, such as a form in Visual Basic.
- The **Cancel** property, a read only Boolean that indicates that the control is the default **Cancel** button for the container.
- The **Default** property, a read only Boolean that indicates that the control is the default button for the container.

Visual Basic provides several more extender methods, properties and events; other containers are not guaranteed to provide these extender methods, properties and events. These Visual Basic specific extender methods, properties and events are:

- The **Container** property, a read only object that represents the visual container of the control.
- The **DragIcon** property, a read/write Picture that specifies the icon to use when the control is dragged.

- The **DragMode** property, a read/write Integer that specifies if the control will automatically drag, or if the user of the control must call the **Drag** method.
- The **Enabled** property, a read only Boolean that specifies if the control is enabled. This extender property is not present unless the control also has an **Enabled** property with the correct procedure ID. For additional information, refer to the topic "Allowing Your Controls to be Enabled and Disabled" in Chapter 9: Building ActiveX Controls.
- The **Height** property, a read/write Integer that specifies the height of the control in the containers scale units.
- The **HelpContextID** property, a read/write Integer that specifies the context ID to use when the F1 key is pressed when the control has the focus.
- The **Index** property, a read only Integer that specifies the position in a control array this instance of the control occupies.
- The **Left** property, a read/write Integer that specifies the position of the left edge of the control to the left edge of the container, specified in the containers scale units.
- The **TabIndex** property, a read/write Integer that specifies the position of the control in the tab order of the controls in the container.
- The **TabStop** property, a read/write Boolean that specifies if Tab will stop on the control.
- The **Tag** property, a read/write String that contains a user-defined value.
- The **ToolTipText** property, a read/write String that contains the text to be displayed when the cursor hovers over the control for more than a second.
- The **Top** property, a read/write Integer that specifies the position of the top edge of the control to the top edge of the container, specified in the containers scale units.
- The **WhatThisHelpID** property, a read/write Integer that specifies the context ID to use when the **Whats This** pop-up is used on the control.
- The **Width** property, a read/write Integer that specifies the width of the control in the containers scale units.
- The **Drag** method, a method to begin, end, or cancel a drag operation of the control.
- The **Move** method, a method to move the position of the control.
- The **SetFocus** method, a method to set the focus to the control.
- The **ShowWhatsThis** method, a method to display a selected topic in a Help file using the **What's This** popup provided by Help.
- The **ZOrder** method, a method to place the control at the front or back of the z-order within its graphical level.
- The DragDrop event, an event that is raised when another control on the form is dropped on this control.
- The DragOver event, an event that is raised when another control on the form is dragged over this control.
- The GotFocus event, an event that is raised when this control gets the focus.
- The LostFocus event, an event that is raised when this control loses the focus.