Visual Basic Reference

Visual Studio 6.0

EditProperty Event

See Also Example Applies To

Occurs when a property page is opened because of the developer pressing the ellipsis button to display a particular property for editing.

Syntax

Sub *object*_**EditProperty(***PropertyName* **As String)**

The EditProperty event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
PropertyName	A string that identifies the property that is to be displayed and edited by the property page.

Remarks

This event happens when a property is assigned a property page via the **Attributes** dialog box. Assigning a property page through the **Attributes** dialog box means that the property is displayed in the Properties window with an ellipsis () next to it, and the developer can press the ellipsis button and the property page is automatically opened; the EditProperty event is then raised, so that the property page author can put the cursor on the correct field.

Visual Basic Reference

Visual Studio 6.0

EditQuery Event

See Also Example Applies To

Occurs when you modify the SQL text of a DECommand object by either clicking **Build** in the **Command Properties** dialog box or selecting **Design** from the shortcut menu or Data Environment designer toolbar.

Syntax

Sub *object*_**EditQuery(***value*, *string1*, *string2***)**

Parameters

The **EditQuery** event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the DECommand object whose SQL text is being modified.
string1	A string expression that specifies the existing SQL text of the DECommand object.
string2	A string expression that specifies the connection string of the associated DEConnection object.

Remarks

This property enables communication between Data View and the Data Environment designer.

Visual Basic Reference

Visual Studio 6.0

EndRequest Event

See Also Example Applies To

Occurs when the **WebClass** object finishes processing an HTTP request and returns a response to the client.

Syntax

Private Sub *object*_**EndRequest()**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list.

Remark

Unless a fatal error occurs, the EndRequest is the last event received by a **WebClass** for a given HTTP request. If a fatal error occurs before EndRequest is fired, the EndRequest event will not be fired. If a fatal error occurs within the EndRequest event, the FatalErrorResponse event will be fired after the EndRequest event.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

EnterCell Event

SeeAlso Example Applies To

Occurs when the currently active cell changes to a different cell.

Syntax

Private Sub *object*_**EnterCell()**

The EnterCell event syntax has one part:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks

Clicking on a fixed row causes this event to occur on the first non-fixed column in that row. Dragging the mouse over a cell does not cause this event to occur.

Visual Basic Reference

Visual Studio 6.0

EnterFocus Event

See Also Example Applies To

Occurs when focus enters the object. The object itself could be receiving focus, or a constituent control could be receiving focus.

Syntax

Sub *object*_**EnterFocus()**

The EnterFocus event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.

Remarks

This event is useful if *object* needs to know that the focus is now inside of it.

The EnterFocus event is raised before any GotFocus event; the GotFocus event will only be raised in *object* or constituent control of *object* that actually got the focus.

Visual Basic: ADO Data Control

Visual Studio 6.0

Error Event (ADO Data Control)

See Also Example Applies To

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

object_**Error**([*Index* **As Integer**,] **ByVal** *ErrorNumber* **As Long**, *Description* **As String**, **ByVal** *Scode* **As Long**, **ByVal** *Source* **As String**, **ByVal** *HelpFile* **As String**, **ByVal** *HelpContext* **As Long**, *fCancelDisplay* **As Boolean**)

The Error event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
Index	Identifies the control if it's in a control array.
ErrorNumber	The native error number.
Description	Describes the error.
Scode	Error code returned by the server.
Source	Source of the error.
HelpFile	The path to a Help file containing more information on the error.
HelpContext	The Help topic context number.
fCancelDisplay	A boolean value that can be set to cancel the error display, as shown in Settings.

Settings

The settings for *fCancelDisplay* are:

Constant	**Value**	**Description**
False	0	Continue.

True	-1	(Default) Display the error message.

Remarks

The Error event will occur whenever an error not caused by Visual Basic halts an operation. Errors can fall into two categories: errors generated by the **ADO**, and general errors (for example, an out of memory error). In cases where the source is **ADO**, the **ADO Data Control** may add contextual text to the description string.

Visual Basic Reference

Visual Studio 6.0

Error Event (Data Control)

See Also Example Applies To

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

Private Sub *object*_**Error (**[*index* **As Integer**,] *dataerr* **As Integer**, *response* **As Integer)**

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
dataerr	The error number.
response	A number corresponding to the response you want to take, as described in Settings.

Settings

The settings for *response* are:

Constant	Value	Description
vbDataErrContinue	0	Continue
vbDataErrDisplay	1	(Default) Display the error message

Remarks

These constants are listed in the Visual Basic (VB) object library in the Object Browser.

You usually provide error-handling functionality for run-time errors in your code. However, run-time errors can occur when none of your code is running, as when:

- A user clicks a **Data** control button.
- The **Data** control automatically opens a database and loads a **Recordset** object after the Form_Load event.
- A custom control performs an operation such as the **MoveNext** method, the **AddNew** method, or the **Delete** method.

If an error results from one of these actions, the Error event occurs.

If you don't code an event procedure for the Error event, Visual Basic displays the message associated with the error.

Errors that occur *before* the Form_Load event, are not trappable and do not trigger the Error event. For example, if at design time you set the properties of the Data control to point to an unknown database table an untrappable error results.

Visual Basic Reference

Error Event Example

This example displays an Open dialog box if the database specified in the **Data** control's **DatabaseName** property isn't found *after* the Form_Load event is complete.

```
Private Sub Data1_Error (DataError As Integer, Response As Integer)
   Select Case DataError
      ' If database file not found.
      Case 3024
         ' Display an Open dialog box.
         CommonDialog1.ShowOpen
      ...
   End Select
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Error Event (Data Report Designer)

See Also Example Applies To

Occurs when an error halts an operation.

Syntax

Private Sub *object*_**Error(***JobType* **As AsyncTypeConstants**, *Cookie* **As Long**, *ErrObj* **As RptError**, *ShowError* **As Boolean)**

The Error event syntax has these parts:

Part	**Description**
object	Required. An object expression that evaluates to an object in the Applies To list.
JobType	Returns the type of operation, as shown in Settings.
Cookie	Returns the ID of the operation. The ID is set when an asynchronous method such as **ExportReport** or **PrintReport** is invoked.
ErrObj	Returns the number of the error.
ShowError	Sets a value that specifies if the error dialog displays.

Settings

The *JobType* settings are:

Constant	**Value**	**Description**
rptAsyncPreview	0	The report is processing a Preview operation.
rptAsyncPrint	1	The report is processing a Print operation.
rptAsyncReport	2	The report is processing an ExportReport operation.

Visual Basic: DataGrid Control

Visual Studio 6.0

Error Event (DataGrid Control)

See Also Example Applies To

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

Private Sub *object*_**Error(**[*index* **As Integer**,] **ByVal** *dataerror* **As Integer**, *response* **As Integer)**

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Index	An integer that identifies a control if it is in a control array.
dataerror	An integer that identifies the error that occurred.
response	An integer that may be set to 0 to suppress error message display, as described in Settings.

Settings

The settings for *response* are:

Setting	Description
0	No error message will be displayed.
1	(Default) The message associated with the error will be displayed.

Remarks

Even if your application handles run time errors in code, errors can still occur when none of your code is executing, as when the user clicks a **Data** control button or changes the current record by interacting with a bound control. If a data access error results from such an action, the Error event is fired.

Not adding code to this event is equivalent to setting the *response* argument to 0.

Note Use the **ErrorText** property to retrieve the error string that will be displayed.

Visual Basic: RDO Data Control

Visual Studio 6.0

Error Event (Remote Data)

See Also Example Applies To

Occurs only as the result of a data access error that takes place when no Visual Basic code is being executed.

Syntax

Private Sub *object* **_Error(**[*index* **As Integer**,]*Number* **As Long,** *Description* **As String,** *Scode* **As Long,** *Source* **As String,** *HelpFile* **As String,** *HelpContext* **As Long,** *CancelDisplay* **As Boolean)**

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	Identifies the control if it's in a control array.
Number	The native error number.
Description	Describes the error.
Scode	ODBC error return code.
Source	Source of the error.
HelpFile	The path to a Help file containing more information on the error.
HelpContext	The Help file context number.
CancelDisplay	A number corresponding to the action you want to take, as described in Settings.

Settings

The settings for *CancelDisplay* are:

Constant	Value	Description
rdDataErrContinue	0	Continue.

rdDataErrDisplay	1	(Default) Display the error message.

Remarks

Generally, the Error event arguments correspond to the properties of the **rdoError** object.

You usually provide error-handling functionality for run-time errors in your code. However, run-time errors can occur when none of your code is running, as when:

- A user clicks a RemoteData control button.
- The **RemoteData control** attempts to open an **rdoConnection** and creates **rdoResultset** objects after the Form_Load event.
- A custom control performs an operation, such as the **MoveNext** method, the **AddNew** method, or the **Delete** method.

If an error results from one of these actions, the Error event occurs.

If you don't code an event procedure for the Error event, Visual Basic displays the message associated with the error.

Visual Basic: Winsock Control

Visual Studio 6.0

Error Event (Winsock Control)

See Also Example Applies To

Occurs whenever an error occurs in background processing (for example, failed to connect, or failed to send or receive in the background).

Syntax

*object*_**Error**(*number* **As Integer**, *Description* **As String**, *Scode* **As Long**, *Source* **As String**, *HelpFile* **as String**, *HelpContext* **As Long**, *CancelDisplay* **As Boolean**)

The Error event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
number	An integer that defines the error code. See Settings below for constants.
description	String containing error information.
Scode	The long SCODE
Source	String describing the error source.
HelpFile	String containing the help file name.
HelpContext	Help file context.
CancelDisplay	Indicates whether to cancel the display. The default is **False**, which is to display the default error message box. If you do not want to use the default message box, set CancelDisplay to **True**.

Settings

The settings for *number* are:

Constant	Value	Description
sckOutOfMemory	7	Out of memory

sckInvalidPropertyValue	380	The property value is invalid.
sckGetNotSupported	394	The property can't be read.
sckSetNotSupported	383	The property is read-only.
sckBadState	40006	Wrong protocol or connection state for the requested transaction or request.
sckInvalidArg	40014	The argument passed to a function was not in the correct format or in the specified range.
sckSuccess	40017	Successful.
sckUnsupported	40018	Unsupported variant type.
sckInvalidOp	40020	Invalid operation at current state
sckOutOfRange	40021	Argument is out of range.
sckWrongProtocol	40026	Wrong protocol for the requested transaction or request
sckOpCanceled	1004	The operation was canceled.
sckInvalidArgument	10014	The requested address is a broadcast address, but flag is not set.
sckWouldBlock	10035	Socket is non-blocking and the specified operation will block.
sckInProgress	10036	A blocking Winsock operation in progress.
sckAlreadyComplete	10037	The operation is completed. No blocking operation in progress
sckNotSocket	10038	The descriptor is not a socket.
sckMsgTooBig	10040	The datagram is too large to fit into the buffer and is truncated.
sckPortNotSupported	10043	The specified port is not supported.
sckAddressInUse	10048	Address in use.
sckAddressNotAvailable	10049	Address not available from the local machine.
sckNetworkSubsystemFailed	10050	Network subsystem failed.
sckNetworkUnreachable	10051	The network cannot be reached from this host at this time.
sckNetReset	10052	Connection has timed out when SO_KEEPALIVE is set.
sckConnectAborted	11053	Connection is aborted due to timeout or other failure.
sckConnectionReset	10054	The connection is reset by remote side.
sckNoBufferSpace	10055	No buffer space is available.
sckAlreadyConnected	10056	Socket is already connected.

sckNotConnected	10057	Socket is not connected.
sckSocketShutdown	10058	Socket has been shut down.
sckTimedout	10060	Socket has been shut down.
sckConnectionRefused	10061	Connection is forcefully rejected.
sckNotInitialized	10093	WinsockInit should be called first.
sckHostNotFound	11001	Authoritative answer: Host not found.
sckHostNotFoundTryAgain	11002	Non-Authoritative answer: Host not found.
sckNonRecoverableError	11003	Non-recoverable errors.
sckNoData	11004	Valid name, no data record of requested type.

Visual Basic Reference

Visual Studio 6.0

ExitFocus Event

See Also Example Applies To

Occurs when focus leaves the object. The object itself could be losing focus, or a constituent control could be losing focus.

Syntax

Sub *object*_**ExitFocus()**

The ExitFocus event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks

This event is useful if *object* needs to know that the focus is now leaving it.

The ExitFocus event is raised after any LostFocus event; the LostFocus event will only be raised in *object* or constituent control of *object* that actually loses the focus.

Visual Basic: MSFlexGrid/MSHFlexGrid Controls

Visual Studio 6.0

Expand Event (MSHFlexGrid)

SeeAlso Example Applies To

Occurs when the user expands a row within the **MSHFlexGrid**. The **Col** and **Row** properties of the **MSHFlexGrid** contain the cell used to expand the band.

Syntax

Private Sub *object*_**Expand(***Cancel***)**

The Expand event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
Cancel	A Boolean expression. If the developer sets **Cancel** to **True**, the expand is cancelled.

Visual Basic: Windows Controls

Visual Studio 6.0

Expand Event (TreeView Control)

See Also Example Applies To

Occurs when a **Node** object in a **TreeView** control is expanded, that is, when its child nodes become visible.

Syntax

Private Sub *object*_**Expand(ByVal** *node* **As Node)**

The Expand event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
node	A reference to the expanded **Node** object.

Remarks

The Expand event occurs after the Click and DblClick events.

The Expand event is generated in three ways: when the user double-clicks a **Node** object that has child nodes; when the **Expanded** property for a **Node** object is set to **True**; and when the plus/minus image is clicked. Use the Expand event to validate an object, as in the following example:

```
Private Sub TreeView1_Expand(ByVal Node As Node)
   If Node.Index <> 1 Then
      Node.Expanded = False   ' Prevent expand.
   End If
End Sub
```

Visual Basic: Windows Controls

Expand Event Example

This example adds several **Node** objects to a **TreeView** control. When a **Node** is expanded, the Expand event is generated, and information about the **Node** is displayed. To try the example, place a **TreeView** control on a form and paste the code into the form's Declarations section. Run the example, and expand the nodes.

```
Private Sub Form_Load()
   Dim nodX As Node
   Set nodX = TreeView1.Nodes.Add(, , "RP", "Root Parent")
   Set nodX = TreeView1.Nodes.Add("RP", tvwChild, "C1", "Child1")
   Set nodX = TreeView1.Nodes.Add("C1", tvwChild, "C2", "Child2")
   Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C3", " Child3")
   Set nodX = TreeView1.Nodes.Add("C2", tvwChild, "C4", " Child4")
   TreeView1.Style = tvwTreelinesPlusMinusText   ' Style 6.
   TreeView1.LineStyle = tvwRootLines   ' Style 1
End Sub

Private Sub TreeView1_Expand(ByVal Node As Node)
   Select Case Node.Key Like "C*"
   Case Is = True
      MsgBox Node.Text & " is a child node."
   End Select
End Sub
```

Visual Basic Reference

Visual Studio 6.0

FatalErrorResponse Event

See Also Example Applies To

Occurs when the processing of a **WebClass** object is terminated due to an error.

Syntax

Private Sub *object*_**FatalErrorResponse**(*senddefault* **As Boolean)**

The FatalErrorResponse event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
senddefault	When set to **True**, *senddefault* signals that the default Active Server Pages error message should be returned to the browser. If the **WebClass** object writes its own message, *senddefault* should be set to **False** to prevent default processing.

Remarks

If the **WebClass** is terminated and you choose to write your own error message, such as

```
senddefault=false
```

then the code to write the message must be placed inside this event.

Any Visual Basic error or terminal error will cause a FatalErrorResponse event.

Visual Basic Reference

Visual Studio 6.0

FontChanged Event

See Also Example Applies To

Occurs when a property of a **stdFont** object changes at run time.

Syntax

Private Sub *object*_**FontChanged(ByVal** *PropertyName* **As String)**

The FontChanged event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
PropertyName	A string returning the name of the **stdFont** object property that has changed. Possible values include Bold, Italic, Name, and Size.

Remarks

The **FontChanged** event will occur only if the **stdFont** object is declared using the **WithEvents** keyword and only for the **stdFont** object. An object declared as **Font** rather than as **stdFont** cant use the **WithEvents** keyword and will cause an automation error. The **FontChanged** event is primarily useful for User controls.

Visual Studio 6.0

Visual Basic: MSChart Control

FootnoteActivated Event

See Also Example Applies To

Occurs when the user double clicks the chart footnote.

Syntax

Private Sub *object*_**FootnoteActivated**(*mouseFlags* **As Integer**, *cancel* **As Integer**)

The FootnoteActivated event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
mouseFlags	Integer. Indicates whether a key is held down when the mouse button is clicked, as described in Settings.
cancel	This argument is not used at this time.

Settings

The event handler determines if a key is held down when the mouse button is clicked and sets *mouseFlags* to:

Constants	**Description**
VtChMouseFlagShiftKeyDown	If the SHIFT key is held down.
VtChMouseFlagControlKeyDown	If the CONTROL key is held down.

Visual Studio 6.0

Visual Basic: MSChart Control

FootnoteSelected Event

See Also Example Applies To

Occurs when the user clicks the chart footnote.

Syntax

Private Sub *object*_**FootnoteSelected** (*mouseFlags* **As Integer**, *cancel* **As Integer**)

The FootnoteSelected event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
mouseFlags	Integer. Indicates whether a key is held down when the mouse button is clicked, as described in Settings.
cancel	This argument is not used at this time.

Settings

The event handler determines if a key is held down when the mouse button is clicked and sets *mouseFlags* to:

Constants	Description
VtChMouseFlagShiftKeyDown	If the SHIFT key is held down.
VtChMouseFlagControlKeyDown	If the CONTROL key is held down.

Visual Studio 6.0

Visual Basic: MSChart Control

FootnoteUpdated Event

See Also Example Applies To

Occurs when the chart footnote changes.

Syntax

Private Sub *object*_**FootnoteUpdated** (*updateFlags* **As Integer**)

The FootnoteUpdated event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
updateFlags	Integer. Provides information about the update of the footnote, as described in Settings.

Settings

The following table lists the constants for *updateFlags*.

Constant	Description
VtChNoDisplay	Absence of update flags; the chart display is not affected. (Defined as 0.)
VtChDisplayPlot	Update will cause the plot to repaint.
VtChLayoutPlot	Update will cause the plot to lay out.
VtChDisplayLegend	Update will cause the legend to repaint.
VtChLayoutLegend	Update will cause the legend to lay out.
VtChLayoutSeries	Update will cause the series to lay out.
VtChPositionSection	A chart section has been moved or resized.

Visual Basic: Windows Controls

Visual Studio 6.0

Format Event (DateTimePicker Control)

See Also Example Applies To

Occurs when the control requests text to be displayed in a callback field.

Syntax

Private Sub *object*_**Format([***index* **As Integer],** *CallbackField* **As String,** *FormattedString* **As String)**

The Format event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
CallbackField	A string expression specifying the callback substring.
FormattedStringField	A string expression specifying the formatted string that is to be displayed.

Remarks

The Format event is used to set the text to be displayed in a callback field.

See **CustomFormat** Property for more information on callback processing.

Visual Basic: Windows Controls

CustomFormat Property, Format Event, FormatSize Event Example

The example displays the current date with the Spanish month name in parentheses. The *CallbackField* "XXXX" is set in the Load event by setting the **CustomFormat** property to a string that includes the four "X"s. The FormatSize event once before the Forma event and is used to determine the size of the callback field. The Format event then occurs which returns a formatted string to replace the callback field text. To try the example, place a **DateTimePicker** control on a form and paste the code into the Declarations section.

```
Option Base 1
Private sSpanishMonthLong(12) As String

Private Sub DTPicker1_Format(ByVal CallbackField As String, FormattedString As String)
   If CallbackField = "XXXX" Then
      FormattedString = sSpanishMonthLong(DTPicker1.Month)
   End If
End Sub

Private Sub DTPicker1_FormatSize(ByVal CallbackField As String, Size As Integer)
   Dim iMaxMonthLen As Integer

   If CallbackField = "XXXX" Then
      iMaxMonthLen = 0
      For i = 1 To 12
         If iMaxMonthLen < Len(sSpanishMonthLong(i)) Then
            iMaxMonthLen = Len(sSpanishMonthLong(i))
         End If
      Next
   End If
   Size = iMaxMonthLen
End Sub

Private Sub Form_Load()
   DTPicker1.CustomFormat = "MMMM(XXXX) dd, yyy"
   DTPicker1.Format = dtpCustom

   sSpanishMonthLong(1) = "Enero"
   sSpanishMonthLong(2) = "Febrero"
   sSpanishMonthLong(3) = "Marzo"
   sSpanishMonthLong(4) = "Abril"
   sSpanishMonthLong(5) = "Mayo"
   sSpanishMonthLong(6) = "Junio"
   sSpanishMonthLong(7) = "Julio"
   sSpanishMonthLong(8) = "Agosto"
   sSpanishMonthLong(9) = "Septiembre"
   sSpanishMonthLong(10) = "Octubre"
   sSpanishMonthLong(11) = "Noviembre"
   sSpanishMonthLong(12) = "Diciembre"
End Sub
```

Visual Basic Reference

Visual Studio 6.0

Format Event (StdDataFormat Object)

See Also Example Applies To

Occurs after the **StdDataFormat** object formats the value.

Syntax

Sub *object*_**Format**(**ByRef** *datavalue* **As StdDataValue**)

The Format event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
datavalue	**StdDataValue** object.

Remarks

The Format event allows you to do formatting that the standard settings of the **StdDataFormat** object cannot accomplish.

Visual Basic: Windows Controls

Visual Studio 6.0

FormatSize Event

See Also Example Applies To

Occurs after the CustomFormat property changes, and before the Format event occurs. The event allows you to set the maximum allowable size of the formatted string so the control can be painted in the screen with ample space for the user-formatted string.

Syntax

Private Sub *object*_**FormatSize([***index* **As Integer],** *CallbackField* **As String,** *Size* **As Long)**

The FormatSize event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
CallbackField	A string expression specifying the callback substring.
Size	A long integer specifying the size of the string that will be returned in the Format event.

Remarks

The FormatSize event is used to set the size of the text to be displayed in a callback field.

Visual Basic: Windows Controls

CustomFormat Property, Format Event, FormatSize Event Example

The example displays the current date with the Spanish month name in parentheses. The *CallbackField* "XXXX" is set in the Load event by setting the **CustomFormat** property to a string that includes the four "X"s. The FormatSize event once before the Forma event and is used to determine the size of the callback field. The Format event then occurs which returns a formatted string to replace the callback field text. To try the example, place a **DateTimePicker** control on a form and paste the code into the Declarations section.

```
Option Base 1
Private sSpanishMonthLong(12) As String

Private Sub DTPicker1_Format(ByVal CallbackField As String, FormattedString As String)
   If CallbackField = "XXXX" Then
      FormattedString = sSpanishMonthLong(DTPicker1.Month)
   End If
End Sub

Private Sub DTPicker1_FormatSize(ByVal CallbackField As String, Size As Integer)
   Dim iMaxMonthLen As Integer

   If CallbackField = "XXXX" Then
      iMaxMonthLen = 0
      For i = 1 To 12
         If iMaxMonthLen < Len(sSpanishMonthLong(i)) Then
            iMaxMonthLen = Len(sSpanishMonthLong(i))
         End If
      Next
   End If
   Size = iMaxMonthLen
End Sub

Private Sub Form_Load()
   DTPicker1.CustomFormat = "MMMM(XXXX) dd, yyy"
   DTPicker1.Format = dtpCustom

   sSpanishMonthLong(1) = "Enero"
   sSpanishMonthLong(2) = "Febrero"
   sSpanishMonthLong(3) = "Marzo"
   sSpanishMonthLong(4) = "Abril"
   sSpanishMonthLong(5) = "Mayo"
   sSpanishMonthLong(6) = "Junio"
   sSpanishMonthLong(7) = "Julio"
   sSpanishMonthLong(8) = "Agosto"
   sSpanishMonthLong(9) = "Septiembre"
   sSpanishMonthLong(10) = "Octubre"
   sSpanishMonthLong(11) = "Noviembre"
   sSpanishMonthLong(12) = "Diciembre"
End Sub
```

Visual Basic Reference

Visual Studio 6.0

GetDataMember Event

See Also Example Applies To

Occurs when a data consumer requests a new data source.

Syntax

Private Sub *object*_**GetDataMember(***DataMember* **As String,** *Data* **As Object)**

The GetDataMember event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
DataMember	A string containing the name of the data member to be bound as a data source.
Data	An object reference to an ADO RecordSet object.

Remarks

The GetDataMember event is available only when an objects **DataSourceBehavior** property is set to **vbDataSource**. You can add code to the GetDataMember event procedure to initialize a data member or to select from multiple data members within an object.

Visual Basic Reference

GetDataMember Event Example

This example uses the **GetDataMember** event to determine what data will be provided by a data source class.

```
Option Explicit
Private rsFirst As ADODB.Recordset
Private rsSecond As ADODB.Recordset
Private rsDefault As ADODB.Recordset

Private Sub Class_GetDataMember(DataMember As String, Data As Object)
   Select Case DataMember
      Case "First"
         Set Data = rsFirst
      Case "Second"
         Set Data = rsSecond
      Case ""    default
         Set Data = rsDefault
      Case Else
         Err.Raise 99999, "DataSource", "Invalid DataMember"
   End Select
End Sub
```

Visual Basic: Windows Controls

Visual Studio 6.0

GetDayBold Event

See Also Example Applies To

Occurs when the control needs to display a date, in order to get bold information.

Syntax

Private Sub *object*_**GetDayBold([***index* **As Integer],** *StartDate* **As Date,** *Count* **As Integer,** *State( )* **As Array)**

The DateDblClick event syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.
StartDate	A date expression specifying the first date that is displayed.
Count	A numeric expression specifying the number of days that are displayed.
State( )	An array of boolean values that specify if a date is bold.

Remarks

The GetDayBold event can be used to set the boldness of days as they are brought into view.

Visual Basic: Windows Controls

GetDayBold Event Example

The following code sets all Fridays to bold. To try the example, place a **MonthView** control on a form, and paste the code into the Declarations section. Then run the project.

```
Private Sub MonthView1_GetDayBold(ByVal StartDate As Date, ByVal Count As Integer, State() As Boolean)
   ' Presuming the start of the week is Sunday, set the variable intBold
   ' to the fifth Friday).  Then set the State of
   ' every Friday to True.
   Dim intBold As Integer
   intBold = mvwFriday
   While intBold < Count
      State(intBold - 1) = True
      intBold = intBold + 7
   Wend
End Sub
```

Visual Basic Reference

Visual Studio 6.0

GetQueryText Event

See Also Example Applies To

Occurs when the source of a Command object is changed.

Syntax

Sub *object*_**GetQueryText(***value***)**

Parameters

The GetQueryText event syntax has these parts:

Part	Description
object	An object expression that evaluates to an item in the Applies To list.
value	A value that specifies the DECommand object whose source has changed.

Remarks

This property enables communication between Data View and the Data Environment designer.

Visual Basic Reference

Visual Studio 6.0

GotFocus Event (UserControl Object and UserDocument Object)

See Also Example Applies To

Occurs in the object or constituent control when focus enters it.

Syntax

Sub *object*_**GotFocus()**

The GotFocus event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.

Remarks

This GotFocus event is not the same GotFocus extender event that the developer who uses *object* handles. This GotFocus event is for the author of *object*, and is internal to *object*.

This event is useful if *object* needs to know that the focus is now on it.

Object itself can get focus only when the **CanGetFocus** property is **True** and there are no constituent controls that can receive the focus.

The EnterFocus event is raised before the GotFocus event.

Do not raise the GotFocus extender event from this event.

Visual Basic Reference

Visual Studio 6.0

GotFocus Event

See Also Example Applies To

Occurs when an object receives the focus, either by user action, such as tabbing to or clicking the object, or by changing the focus in code using the **SetFocus** method. A form receives the focus only when all visible controls are disabled.

Syntax

`Private Sub Form_GotFocus( )`

Private Sub *object*_**GotFocus(**[*index* **As Integer**]**)**

The GotFocus event syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
index	An integer that uniquely identifies a control if it's in a control array.

Remarks

Typically, you use a GotFocus event procedure to specify the actions that occur when a control or form first receives the focus. For example, by attaching a GotFocus event procedure to each control on a form, you can guide the user by displaying brief instructions or status bar messages. You can also provide visual cues by enabling, disabling, or showing other controls that depend on the control that has the focus.

Note An object can receive the focus only if its **Enabled** and **Visible** properties are set to **True**. To customize the keyboard interface in Visual Basic for moving the focus, set the tab order or specify access keys for controls on a form.

Visual Basic Reference

GotFocus Event Example

This example displays a status bar message when a button in an **OptionButton** group gets the focus. To try this example, paste the code into the Declarations section of a form that contains two **OptionButton** controls and a **Label** control. Set the **Name** property for both **OptionButton** controls to OptionGroup, and then press F5 and click the **OptionButton** controls.

```
Private Sub Form_Load ()
   Label1.AutoSize = True
End Sub

Private Sub OptionGroup_GotFocus (Index As Integer)
   Select Case Index
      Case 0
         Label1.Caption = "Option 1 has the focus."
      Case 1
         Label1.Caption = "Option 2 has the focus."
   End Select
End Sub

Private Sub OptionGroup_LostFocus (Index As Integer)
   Label1.Caption = ""
End Sub
```