This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

TextHeight Method

See Also Example Applies To

Returns the height of a text string as it would be printed in the current font of a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**TextHeight(***string***)**

The **TextHeight** method syntax has these parts:

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** object with the focus is assumed to be *object*.
String	Required. A string expression that evaluates to a string for which the text height is determined. Parentheses must enclose the string expression.

Remarks

The height is expressed in terms of the **ScaleMode** property setting or **Scale** method coordinate system in effect for *object*. Use **TextHeight** to determine the amount of vertical space required to display the text. The height returned includes the normal leading space above and below the text, so you can use the height to calculate and position multiple lines of text within *object*.

If *string* contains embedded carriage returns, **TextHeight** returns the cumulative height of the lines, including the leading space above and below each line.

© 2018 Microsoft

Visual Basic Reference

TextHeight Method Example

The **TextHeight** method is used to center a line of text vertically on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HalfWidth, HalfHeight, Msg   ' Declare variable.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   BackColor = QBColor(4)   ' Set background color.
   ForeColor = QBColor(15)   ' Set foreground color.
   Msg = "Visual Basic"   ' Create message.
   FontSize = 48   ' Set font size.
   HalfWidth =  TextWidth(Msg) / 2    ' Calculate one-half width.
   HalfHeight = TextHeight(Msg) / 2   ' Calculate one-half height.
   CurrentX = ScaleWidth / 2 - HalfWidth   ' Set X.
   CurrentY = ScaleHeight / 2 - HalfHeight   ' Set Y.
   Print Msg   ' Print message.
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

TextWidth Method

See Also Example Applies To

Returns the width of a text string as it would be printed in the current font of a **Form**, **PictureBox**, or **Printer**. Doesn't support named arguments.

Syntax

object.**TextWidth(***string***)**

The **TextWidth** method syntax has these parts:

Part	**Description**
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.
String	Required. A string expression that evaluates to a string for which the text width is determined. Parentheses must surround the string expression.

Remarks

The width is expressed in terms of the **ScaleMode** property setting or **Scale** method coordinate system in effect for *object*. Use **TextWidth** to determine the amount of horizontal space required to display the text. If *string* contains embedded carriage returns, **TextWidth** returns the width of the longest line.

© 2018 Microsoft

Visual Basic Reference

TextWidth Method Example

The **TextWidth** method is used to center a line of text horizontally on a form. To try this example, paste the code into the Declarations section of a form, and then press F5 and click the form.

```
Private Sub Form_Click ()
   Dim HalfHeight, HalfWidth, Msg   ' Declare variables.
   AutoRedraw = -1   ' Turn on AutoRedraw.
   BackColor = QBColor(4)   ' Set background color.
   ForeColor = QBColor(15)   ' Set foreground color.
   Msg = "Visual Basic"   ' Create message.
   FontSize = 48   ' Set font size.
   HalfWidth = TextWidth(Msg) / 2    ' Calculate one-half width.
   HalfHeight = TextHeight(Msg) / 2   ' Calculate one-half height.
   CurrentX = ScaleWidth / 2 - HalfWidth   ' Set X.
   CurrentY = ScaleHeight / 2 - HalfHeight   ' Set Y.
   Print Msg   ' Print message.
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

ToDefaults Method

See Also Example Applies To

Returns the chart to its initial settings.

Syntax

object.**ToDefaults**

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

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Trace Method

See Also Example Applies To

Passes the specified string to the Win32 OutputDebugString API. The string may be captured by a suitable debugging tool, for example, DBMON.

Syntax

object.**Trace**(*traceoutput* **As String**)

The **Trace** method syntax has these parts:

Part	**Description**
object	An object expression that evaluates to an object in the Applies To list.
traceoutput	String written to the Win32 **OutputDebugString** API.

Remarks

The trace string typically contains debugging information during development. For a production **WebClass** the trace string might contain error messages, as well as performance and statistical data.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

TwipsToChartPart Method

See Also Example Applies To

Identifies a chart part by using the x and y set of coordinates on to identify that part.

Syntax

object.**TwipsToChartPart** (*xVal*, *yVal*, *part*, *index1*, *index2*, *index3*, *index4*)

The **TwipsToChartPart** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
xVal,yVal	Long. The horizontal and vertical coordinates of the point.
part	Integer. A VtChPartType constant that identifies the chart part that is located at the *xVal* and *yVal* coordinates.
index1	Integer. If *part* refers to a series or a data point, this argument specifies which series. Series are numbered in the order their corresponding columns appear in the data grid from left to right, beginning with 1. If part refers to an axis or axis label, this argument identifies the axis type using the VtChAxisId constant.
index2	Integer. If *part* refers to a data point, this argument specifies which data point in the series identified by index1. Data points are numbered in the order their corresponding rows appear in the data grid from top to bottom, beginning with 1. If part refers to an axis, axis title, or axis label, this argument refers to the axis index which is currently not used. In this case, the only valid value for this argument is 1.
index3	Integer. If *part* refers to an axis label, this argument refers to the level of the label. Axis label levels are numbered from the axis out, beginning with 1. If *part* is not an axis label, the argument is unused.
index4	Integer. This argument is unused at this time.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Studio 6.0

Visual Basic: MSChart Control

TypeByChartType Method

See Also   Example   Applies To

Returns the series type used to draw a series if the chart type is set to *chType*. This method allows you to get the series type information based on a specified chart type without actually setting the chart type.

Syntax

object.**TypeByChartType** (*chtype*)

The **TypeByChartType** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
chtype	Integer. A VtChChartType constant describing the chart type.

Return Values

Depending on the chart type specified by the *chtype* argument, the **TypeByChartType** method returns one of the **VtChSeriesType** values, as shown below:

Constant	Value	Description
VtChSeriesTypeDefault	-1	Default
VtChSeriesType3dBar	0	3D Bar
VtChSeriesType2dBar	1	2D Bar
VtChSeriesType3dLine	5	3D Line
VtChSeriesType2dLine	6	2D Line
VtChSeriesType3dArea	7	3D Area
VtChSeriesType3dStep	9	3D Step
VtChSeriesType2dStep	10	2D Step

VtChSeriesType2dXY	11	XY
VtChSeriesType2dPie	24	2D Pie

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

Update Method (OLE Container)

See Also Example Applies To

Retrieves the current data from the application that supplied the object and displays that data as a graphic in the **OLE** container control.

Syntax

object.**Update**

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

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

Update Method (Remote Data)

See Also Example Applies To

Saves the contents of the copy buffer row to a specified updatable **rdoResultset** object and discards the copy buffer.

Syntax

object.**Update**

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

Remarks

Use **Update** to save the current row and any changes youve made to it to the underlying database table(s). Changes you make to the **rdoResultset** after using the **AddNew** or **Edit** methods can be lost if you do not use the **Update** method before the application ends.

Note When you use the ClientBatch cursor library, all updates to the base tables are deferred until you use the **BatchUpdate** method. In this case, the **Update** method updates the local **rdoResultset**, but does not update the base tables. These changes can be lost if the application ends before the **BatchUpdate** method has been completed.

Changes to the current row are lost if:

- You use the **Edit** or **AddNew** method, and then reposition the current row pointer to another row without first using **Update**.
- You use **Edit** or **AddNew**, and then use **Edit** or **AddNew** again without first using **Update**.
- You cancel the update with the **CancelUpdate** method.
- You set the **Bookmark** property to another row.
- You close the result set referred to by *object* without first using **Update**.
- The application ends before the **Update** method is executed, as when system power is interrupted.

To edit a row, use the **Edit** method to copy the contents of the current row to the copy buffer. If you dont use **AddNew** or **Edit** first, an error occurs when you use **Update** or attempt to add a new row.

To add a new row, use the **AddNew** method to initialize and activate the copy buffer.

Using **Update** produces an error under any of the following conditions:

- There is no current row.
- The connection or **rdoResultset** is read-only.

- No columns in the row are updatable.
- You do not have an **Edit** or **AddNew** operation pending.
- Another user has locked the row or data page containing your row.
- The user does not have permission to perform the operation.
- Depending on the driver and type of cursor being used, you might not be able to use the cursor to update the primary key.

© 2018 Microsoft

Visual Basic: RDO Data Control

AddNew, Update, CancelUpdate Method Example

The following example illustrates use of the **AddNew** method to add new rows to a base table. This example assumes that you have read-write access to the table, that the column data provided meets the rules and other constraints associated with the table, and there is a unique index on the table. The data values for the operation are taken from three **TextBox** controls on the form. Note that the unique key for this table is not provided here as it is provided automatically it is an *identity* column.

```
Option Explicit
Dim er As rdoError
Dim cn As New rdoConnection
Dim qy As New rdoQuery
Dim rs As rdoResultset
Dim col As rdoColumn

Private Sub AddNewJob_Click()
On Error GoTo ANEH

With rs
   .AddNew
   !job_desc = JobDescription
   !min_lvl = MinLevel
   !max_lvl = MaxLevel
   .Update
End With
Exit Sub

UpdateFailed:
MsgBox "Update did not suceed."
rs.CancelUpdate
Exit Sub
A
NEH:
Debug.Print Err, Error
For Each er In rdoErrors
   Debug.Print er
Next
Resume UpdateFailed

End Sub

Private Sub Form_Load()

cn.CursorDriver = rdUseOdbc
cn.Connect = "uid=;pwd=;server=sequel;" _
   & "driver={SQL Server};database=pubs;dsn=;"
cn.EstablishConnection
With qy
   .Name = "JobsQuery"
   .SQL = "Select * from Jobs"
   .RowsetSize = 1
   Set .ActiveConnection = cn
   Set rs = .OpenResultset(rdOpenKeyset, _
      rdConcurRowver)
   Debug.Print rs.Updatable
End With

Exit Sub
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Extensibility Reference

Visual Studio 6.0

Update Method

See Also Example Applies To

Refreshes the contents of the **AddIns** collection from the add-ins listed in the Vbaddin.ini file in the same manner as if the user had opened the Add-In Manager dialog box.

Syntax

object.**Update**

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

Remarks

All add-ins listed in the Vbaddin.ini file must be registered ActiveX components in the Registry before they can be used in Visual Basic.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

UpdateControls Method

See Also Example Applies To

Gets the current record from a **Data** control's Recordset object and displays the appropriate data in controls bound to a **Data** control. Doesn't support named arguments.

Syntax

object.**UpdateControls**

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

Remarks

Use this method to restore the contents of bound controls to their original values, as when a user makes changes to data and then decides to cancel the changes.

This method creates the same effect as making the current record current again, except that no events occur.

The **UpdateControls** method terminates any pending **Edit** or **AddNew** operation.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

UpdateControls Method (BindingCollection Object)

See Also Example Applies To

Gets the current row from the **BindingCollection** object's data source and resets the contents of controls bound through the object.

Syntax

object.**UpdateControls**

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

Remarks

Use this method to restore the contents of bound controls to their original values, as when a user makes changes to data and then decides to cancel the changes.

Calling **UpdateControls** has the same effect as resetting the **DataChanged** property on each **Binding** object to **False**, then making the current row current again, with the exception that no events occur when calling **UpdateControls**. By not invoking any events, this method can be used to simplify an update operation because no additional validation or change event procedures are triggered.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

UpdateControls Method (Remote Data)

See Also Example Applies To

Gets the current row from a RemoteData controls **rdoResultset** object and displays the appropriate data in controls bound to a **RemoteData** control.

Syntax

object.**UpdateControls**

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

Remarks

Use this method to restore the contents of bound controls to their original values, as when a user makes changes to data and then decides to cancel the changes.

This method creates the same effect as making the current row current again, except that no events occur. By not invoking any events, this method can be used to simplify an update operation because no additional validation or change event procedures are triggered.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

UpdateRecord Method

See Also Example Applies To

Saves the current values of bound controls. Doesn't support named arguments.

Syntax

object.**UpdateRecord**

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

Remarks

Use this method to save the current contents of bound controls to the database during the Validate event without triggering the Validate event again. Using this method avoids creating a cascading event.

The **UpdateRecord** method has the same effect as executing the **Edit** method, changing a field, and then executing the **Update** method, except that no events occur.

You can use this method to avoid triggering the Validate event.

Whenever you attempt to update a record in the database, any validation rules must be satisfied before the record is written to the database. These rules are established by setting the **ValidationRule** property or, in the case of Microsoft SQL Server, by Transact SQL defaults, rules, and triggers written to enforce referential and data integrity.

In some cases, the update may not occur because the operation violates referential integrity constraints, the page containing the record is locked, the database or **Recordset** object isn't updatable, or the user doesn't have permission to perform the operation. Any of these conditions generates a trappable error.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RDO Data Control

Visual Studio 6.0

UpdateRow Method (Remote Data)

See Also Example Applies To

Saves the current values of bound controls to the database.

Syntax

object.**UpdateRow**

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

Remarks

Use this method to save the current contents of bound controls to the database during the Validate event, but without triggering the Validate event again. You can use this method to avoid triggering the Validate event. Using this method avoids a cascading event.

The **UpdateRow** method has the same effect as executing the **Edit** method, changing a column, and then executing the **Update** method, except that no events occur.

Note When you use the ClientBatch cursor library, all updates to the base tables are deferred until you use the **BatchUpdate** method. In this case, the **UpdateRow** method updates the local **rdoResultset**, but does not update the base tables. These changes can be lost if the application ends before the **BatchUpdate** method has been completed.

Whenever you attempt to update a row in the database, any validation rules must be satisfied before the row is written to the database. In the case of Microsoft SQL Server, these rules are established by Transact SQL defaults, rules, and triggers written to enforce referential and data integrity.

An update may not occur because of any of the following reasons, which can also trigger a trappable error:

- The page containing the row or the row itself is locked.
- The database or **rdoResultset** object isnt updatable.
- The user doesnt have permission to perform the operation.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic: RichTextBox Control

Visual Studio 6.0

Upto Method

See Also Example Applies To

Moves the insertion point up to, but not including, the first character that is a member of the specified character set in a **RichTextBox** control.

Syntax

object.**Upto**(*characterset*, *forward*, *negate*)

The **Upto** method syntax has these parts:

Part	**Description**
object	Required. An object expression that evaluates to an object in the Applies To list.
characterset	Required. A string expression that specifies the set of characters to look for when moving the insertion point, based on the value of *negate*.
forward	Optional. A Boolean expression that determines which direction the insertion point moves, as described in Settings.
negate	Optional. A Boolean expression that determines whether the characters in *characterset* define the set of target characters or are excluded from the set of target characters, as described in Settings.

Settings

The settings for *forward* are:

Setting	**Description**
True	(Default) Moves the insertion point forward, toward the end of the text.
False	Moves the insertion point backward, toward the start of the text.

The settings for *negate* are:

Setting	Description
True	The characters not specified in the *characterset* argument are used to move the insertion point.
False	(Default) The characters specified in the *characterset* argument are used to move the insertion point.

© 2018 Microsoft

Visual Basic: RichTextBox Control

Upto Method Example

This example defines a pair of keyboard shortcuts that moves the insertion point in a **RichTextBox** control to the end of a sentence (ALT+S) or the end of a word (ALT+W). To try this example, put a **RichTextBox** control on a form. Paste this code into the KeyUp event of the **RichTextBox** control. Then run the example.

```
Private Sub RichTextBox1_KeyUp (KeyCode As Integer, Shift As Integer)
   If Shift = vbAltMask Then
      Select Case KeyCode
         ' If Alt+S:
         Case vbKeyS
            ' Move insertion point to the end of the sentence.
               RichTextBox1.Upto ".?!:", True, False
         ' If Alt+W:
         Case vbkeyW
            ' Move insertion point to the end of the word.
               RichTextBox1.Upto " .?!:", True, False
      End Select
   End If
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

URLFor Method

See Also Example Applies To

Used to specify the uniform resource locator (URL) that the system needs to reference a webclasss HTML template or WebItem in the browser.

Syntax

object.**URLFor**(*WebItemobject* **As WebItem**,[*eventname*])

The **URLFor** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
WebItemobject	**WebItem** object for which you want to generate a URL and fire an event.
eventname	String that refers to an event in the **WebItem** if omitted it refers to the Respond event. The returned URL can be embedded in the response to the client and, when referenced, will trigger the related event in the WebClass.

Remarks

Also used to generate a string containing a URL at run time that refers to a **WebItem** and specifies a custom event to fire.When receiving an event name that was not defined at design time, the URLFor method returns a URL which when processed by the client fires the UserEvent event on the respective WebItem. The UserEvent event will be passed the name of the run-time defined event.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

ValidateControls Method

See Also Example Applies To

Ensures that the contents of the last control on the form are valid before exiting the form.

Syntax

object.**ValidateControls**

The *object* placeholder represents an object expression that evaluates to an object in the Applies To list. If *object* is omitted, the **Form** with the focus is assumed to be *object*.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

WhatsThisMode Method

See Also Example Applies To

Causes the mouse pointer to change into the What's This pointer and prepares the application to display What's This Help on the selected object.

Syntax

object.**WhatsThisMode**

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

Remarks

Executing the **WhatsThisMode** method places the application in the same state you get by clicking the What's This button in the title bar. The mouse pointer changes to the What's This pointer. When the user clicks an object, the **WhatsThisHelpID** property of the clicked object is used to invoke context-sensitive Help. This method is especially useful when invoking Help from a menu in the menu bar of your application.

© 2018 Microsoft

Visual Basic Reference

WhatsThisMode Method Example

This example uses a command in a menu to change the mouse pointer to the What's This pointer and enable context-sensitive Help. To try the example, create a menu, and paste the code into the Click event of one of the **Menu** controls. Press F5, and click the menu command to toggle the application into the What's This state.

```
Private Sub mnuContextHelp_Click ()
   Form1.WhatsThisMode
End Sub
```

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

Write Method

See Also Example Applies To Specifics

Description

Writes a specified string to a **TextStream** file.

Syntax

object.**Write(***string***)**

The **Write** method syntax has these parts:

Part	Description
object	Required. Always the name of a **TextStream** object.
string	Required. The text you want to write to the file.

Remarks

Specified strings are written to the file with no intervening spaces or characters between each string. Use the **WriteLine** method to write a newline character or a string that ends with a newline character.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

WriteBlankLines Method

See Also Example Applies To Specifics

Description

Writes a specified number of newline characters to a **TextStream** file.

Syntax

object.**WriteBlankLines(***lines***)**

The **WriteBlankLines** method syntax has these parts:

Part	Description
object	Required. Always the name of a **TextStream** object.
lines	Required. Number of newline characters you want to write to the file.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic for Applications Reference

Visual Studio 6.0

WriteLine Method

See Also Example Applies To Specifics

Description

Writes a specified string and newline character to a **TextStream** file.

Syntax

object.**WriteLine(**[*string*]**)**

The **WriteLine** method syntax has these parts:

Part	Description
object	Required. Always the name of a **TextStream** object.
string	Optional. The text you want to write to the file. If omitted, a newline character is written to the file.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

WriteProperty Method

See Also Example Applies To

Writes a value to be saved to a **PropertyBag** class object.

Syntax

object.**WriteProperty(***DataName*, *Value*[, *DefaultValue*]**)**

The **WriteProperty** method syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
DataName	A string expression to represents the data value to be placed in the property bag.
Value	The data value to save in the property bag.
DefaultValue	The default value for the data.

Remarks

The **WriteProperty** method will write a data value in the property bag, and associate it with the string value in *DataName*. This string value will be used to access the data value when the **ReadProperty** method is called to retrieve a saved data value from the property bag.

Note Specifying a default value reduces the size of the file belonging to the container of the control. A line for the property is written to the file only if the value to be written is different from the default. Wherever possible, you should specify default values for the properties of the control when initializing, saving, and retrieving property values.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

WriteTemplate Method

See Also Example Applies To

Processes an HTML template attached to a **WebItem** object and returns it to the browser.

Syntax

object.**WriteTemplate**(*Template* **As Variant**)

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
Template	Optional. Indicates the template to return to the browser.

Remarks

Use this method to send the HTML template to the client. The ReplaceToken event is fired during processing of a template if that template contains replacement tokens. The template is written to the **Response** object.

The **WriteTemplate** method does not fire the Respond event for the associated **WebItem**. It is recommended that the **WriteTemplate** method only be called from within the Respond event of the **WebItem** and that the Respond event be triggered by setting the **NextItem** property.

© 2018 Microsoft

This documentation is archived and is not being maintained.

Visual Basic Reference

Visual Studio 6.0

ZOrder Method

See Also Example Applies To

Places a specified **MDIForm**, **Form**, or control at the front or back of the z-order within its graphical level. Doesn't support named arguments.

Syntax

object.**ZOrder** *position*

The **ZOrder** method syntax has these parts:

Part	Description
object	Optional. An object expression that evaluates to an object in the Applies To list. If *object* is omitted, the form with the focus is assumed to be *object*.
Position	Optional. Integer indicating the position of *object* relative to other instances of the same *object*. If position is 0 or omitted, *object* is positioned at the front of the z-order. If position is 1, *object* is positioned at the back of the z-order.

Remarks

The z-order of objects can be set at design time by choosing the Bring To Front or Send To Back menu command from the Edit menu.

Within an **MDIForm** object, **ZOrder** sends MDI child forms to either the front or the back of the MDI client area, depending on the value of *position*. For an **MDIForm** or **Form** object, **ZOrder** sends the form to either the front or the back of the screen, depending on the value of *position*. As a result, forms can be displayed in front of or behind other running applications.

Three graphical layers are associated with forms and containers. The back layer is the drawing space where the results of the graphics methods are displayed. Next is the middle layer where graphical objects and **Label** controls are displayed. The front layer is where all nongraphical controls like **CommandButton**, **CheckBox**, or **ListBox** are displayed. Anything contained in a layer closer to the front covers anything contained in the layer(s) behind it. **ZOrder** arranges objects only within the layer where the object is displayed.

© 2018 Microsoft