Add Method (ColumnDefs collection)

Applies To

ColumnDefs collection

Description

Creates a new **ColumnDef** object, and adds it to the **ColumnDefs** collection.

Usage

object.**Add(***key* **As Variant,** *caption* **Variant)**

The **Add** method has these parts:

Part	*Description*
object	An object expression that evaluates to a ColumnDefs collection.
key	Required. A unique string identifying the newly created ColumnDef object. The key value must be unique.
caption	Required. The caption text associated with this ColumnDef object.

Remarks

The Add method returns a reference to the newly inserted **ColumnDef** object.

Use the index argument to insert a ColumnDef in a specific position.

See Also

Remove method**, Clear** method**, ColumnDef** object.

Example

The following example adds a new item to the **GTList** control's **ColumnDefs** collection:

```
GTList1.ColumnDefs.Add("My Column", "Red Bricks")
```

Add Method (ListImages collection)

Applies To

ListImages collection

Description

Creates a new **ListImage** object, and adds it to the **ListImages** collection.

Usage

object.**Add([***index* **As Variant], [***key* **As Variant], [***picture* **As Variant])**

The **Add** method has these parts:

Part	Description
object	An object expression that evaluates to a SubItems collection.
index	Optional. An integer that uniquely identifies the newly created object in the collection.
key	Optional. A unique string identifying the newly created SubItem. The key value must be unique.
picture	Optional. The picture associated with this SubItem object.

Remarks

The Add method returns a reference to the newly inserted **ListImage** object.

Use the index argument to insert a SubItem in a specific position.

See Also

Remove method**, Clear** method**, ListImage** object, **ListItems** collection.

Example

The following example adds a new SubItem object to the **GTList** control's **ListItems** collection:

```
GTList1.ListImages.Add( , , "Phone.BMP")
```

Add Method (ListItems collection)

Applies To

ListItems collection

Description

Creates a new **ListItem** object, and adds it to the **ListItems** collection.

Usage

object.**Add([***index* **As Variant], [***key* **As Variant], [***text* **As Variant],**
[*image* **As Variant], [***selectedimage* **As Variant])**

The **Add** method has these parts:

Part	**Description**
object	An object expression that evaluates to a ListItems collection.
index	Optional. An integer that uniquely identifies the newly created object in the collection.
key	Optional. A unique string identifying the newly created ListItem object. The key value must be unique.
text	Optional. The text associated with this ListItem object.
image	Optional. An integer value specifying an index into the parent control's **ImageList** control The image is displayed with the **ListItem** object.
selectedimage	Optional. An integer value specifying an index into the parent control's **ImageList** control The image is displayed with the **ListItem** object when the object is selected.

Remarks

The Add method returns a reference to the newly inserted **ColumnDef** object.

Use the index argument to insert a ColumnDef in a specific position.

Before setting either the *image* or *selectedimage* properties, you must first initialize them. You can do this at design time or at run time.

See Also

Remove method**, Clear** method**, ListItem** object, **ListImages** object

Example

The following example adds a new item to the **GTList** control's **ListItems** collection:

```
GTList1.ColumnDefs.Add( , , "Slate Bricks" 1,2)
```

Add Method (SubItems collection)

Applies To

SubItems collection

Description

Creates a new **SubItem** object, and adds it to the **SubItems** collection.

Usage

object.**Add([***index* **As Variant], [***key* **As Variant], [***text* **As Variant])**

The **Add** method has these parts:

Part	Description
object	An object expression that evaluates to a SubItems collection.
index	Optional. An integer that uniquely identifies the newly created object in the collection.
key	Optional. A unique string identifying the newly created SubItem. The key value must be unique.
text	Optional. The text associated with this SubItem object.

Remarks

The Add method returns a reference to the newly inserted **SubItem** object.

Use the index argument to insert a SubItem in a specific position.

See Also

Remove method**, Clear** method**, SubItem** object, **ListItems** collection.

Example

The following example adds a new SubItem object to the **GTList** control's **ListItems** collection:

```
GTList1.ListItems.SubItems.Add( , , "Manufactured 3 x 4 x 6")
```

AddItem Method

Applies To

GTList control**, GTCombo** control

Description

Adds an item to a **GTList** or **GTCombo** control. Doesn't support named arguments..

Usage

object.**AddItem** *item, index*

The **AddItem** method has these parts:

Part	Description
object	Required. An object expression that evaluates to one of the objects or controls in the Applies To list.
item	Required. String expression specifying the item to add to the object.
index	Optional. Integer specifying the position within the object where the new item is placed. For the first item in a **GTList** or **GTCombo** control index is 0.

Remarks

If you supply a valid value for index, item is placed at that position within the object. If index is omitted, item is added at the proper sorted position (if the Sorted property is set to True) or to the end of the list (if **Sorted** is set to **False**).

A **GTList** or **GTCombo** control that is bound to a **DataSource** doesn't support the **AddItem** method.

See Also

ListItems collection, **ListItem** object.

Example

This example adds 100 entries to a **GTList** control.

```
Dim I as Integer
Dim Entry As String

For I = 1 To 100  ' Count from 1 to 100.
      Entry = "List Entry " & I     ' Create entry.
      GTList1.AddItem Entry   ' Add the entry.
Next I
```

AllowColumnDragDrop Property

Applies To

GTList control**, GTCombo** control

Description

Determines if the user can move columns of the control by dragging and dropping the columns.

Usage

object.**AllowColumnDragDrop** [= *boolean*]

The **AllowColumnDragDrop** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the user can move the control's columns by dragging and dropping.

Remarks

Typically the user will drag and drop a column by clicking and holding the mouse button over the desired column, then drag it to the target location, and drop the column object by releasing the mouse button.

See Also

AllowColumnResize property.

Example

The following code changes the **AllowColumnDragDrop** property of a **GTList** control:

```
GTList1.AllowColumnDragDrop = True
```

AllowColumnResize Property

Applies To

GTList control**, GTCombo** control

Description

Determines if the user can resize the columns of the control using the mouse.

Usage

object.**AllowColumnResize** [= *boolean*]

The **AllowColumnResize** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the user can resize the control's columns using the mouse.

Remarks

To resize the width of the control's column, the user puts the mouse over the column header bar at the top of the control. The mouse icon will change to the resize width cursor. When the cursor indicates re-sizing is allowed by clicking and holding the mouse button, and dragging the column to the desired width.

See Also

AllowColumnDragDrop, SubRowsStatic properties.

Example

The following code changes the **AllowColumnResize** property of a **GTList** control:

```
GTList1.AllowColumnResize = False
```

AllowColumnSortClick Property

Applies To

GTList control**, GTCombo** control

Description

Determines if the column will sort the list when the user clicks on the column header.

Usage

object.**AllowColumnSortClick** [= *boolean*]

The **AllowColumnSortClick** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the list is sorted when the column header is clicked by the user.

Remarks

To sort the items of the control's column, the user puts the mouse over the column header bar at the top of the control, and clicks on the column header above the column to sort. If this property is True, the list will be sorted when the column header is clicked according to the **SortKeyXXX** and **SortOrderXXX** properties. This feature only works when the control is operating in non-bound mode. Specifically the control must not be bound to a database by the **DataSource** property, or be in **Virtual** mode.

See Also

SortKey, **SortKey2**, **SortKey3**, **SortOrder**, **SortOrder2**, **SortOrder3** properties.

Example

The following code changes the **AllowColumnSortClick** property of a **GTList** control:

```
GTList1.AllowColumnSortClick = False
```

AutoPositionList Property

Applies To

GTCombo control

Description

Determines if the position of the list items are automatically moved to match the text in the edit portion of the control when the edited text changes.

Usage

*object.***AutoPositionList** [= *boolean*]

The **AutoPositionList** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the list is automatically positioned to match the typed in text.

Remarks

To have the control automatically position the items in the list according to the text that is typed in by the user, set this property to **True**.

See Also

ListIndex property, **PositionList** event.

Example

The following code changes the **AutoPositionList** property of a **GTList** control:

```
GTList1.AutoPositionList = True
```

BackColor Property (ColumnDef, ListItem, SubItem object)

Applies To

ColumnDef object**, ListItem** object**, SubItem** object

Description

Specifies the background color used when drawing the object specified.

Usage

object. **BackColor** [= *color*]

The **Backcolor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
color	A value or constant that determines the background color of the object, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the background color value for each object.

See Also

ForeColor property

Example

The following code sets the **BackColor** property for a **ListItem** object to the default background color of the parent control:

```
GTList1.ListItems.Item(0).BackColor = GList1.DefBackColor
```

BackStyle Property (ColumnDef object, ListItem object, SubItem object)

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies the background style to be used when the object is displayed in the control.

Usage

object.**BackStyle** [= *number*]

The **BackStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
Number	An integer specifying the background style to use when drawing the object in the control's window.

Settings

The settings for number are:

Setting	Value	Description
gtBackStyleTransparent	0	(Default) Transparent. Draws objects transparently.
gtBackStyleOpaque	1	Opaque. Draws objects opaquely.
GtBackStyleUseDefault	2	Use Default. Use the value set in the **DefBackStyle** property.

Remarks

When the object's are drawn by the **GTList** or **GTCombo** control, and the object's **BackStyle** is set to **gtBackStyleUseDefault**, the value in the **DefBackStyle** property is applied to the object resulting in a transparent, or opaque appearance as described above in Settings.

See Also

DefBackColor, DefBackStyle, BackStyle properties

Example

The following code sets the **BackStyle** for a **GTList** control's first ListItem to Opaque:

```
Glist1.ListItems(0).BackStyle = gtBackStyleOpaque
```

CalcRowCountOnLoad Property

Applies To

GTList control**, GTCombo** control

Description

Enables or disables if the control will count the exact number of records in the bound database when the control is loaded into memory.

Usage

*object.***CalcRowCountOnLoad** [= *boolean*]

The **CalcRowCountOnLoad** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the CalcRowCountOnLoad feature is enabled.

Remarks

Setting this value to **True** causes the control to determine the number of records in the bound database when the control is loaded into memory. This provides an accurate representation for the vertical scroll bar's thumb position. On large external databases, determining the number of records can be slow. Setting this value to **False** approximates the number of records in the bound database, but can yield better performance. Default = **False**

See Also

ListCount property

Example

The following code turns the control's **CalcRowCountOnLoad** on by setting the **CountRowCountOnLoad** property to **True**.

```
GTList1.CalcRowCountOnLoad = True
```

CaptionBackColor Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the BackColor used for the ColumnDef object's caption text.

Usage

object.**CaptionBackColor** [= *color*]

The **CaptionBackColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
color	A value or constant that determines the background color of the object's **Caption** text, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

See Also

CaptionForeColor property

Example

The following code sets the **CaptionBackColor** to Yellow using the **RGB** function:

```
GTCombo1.ColumnDefs.Item(0).CaptionBackColor = RGB(255,255,0)
```

CaptionBackStyle Property (ColumnDef object)

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies the background style to be used when the column header caption is displayed.

Usage

object.**CaptionBackStyle** [= *number*]

The **CaptionBackStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
number	An integer specifying the background style to use when drawing the column header's caption.

Settings

The settings for number are:

Setting	Value	Description
gtBackStyleTransparent	0	(Default) Transparent. Draws captions transparently.
gtBackStyleOpaque	1	Opaque. Draws captions opaquely.
gtBackStyleUseDefault	2	Use Default. Use the value set in the **DefCaptionBackStyle** property.

Remarks

When the column header captions are drawn by the **GTList** or **GTCombo** control, and the column header's **CaptionBackStyle** is set to **gtBackStyleUseDefault**, the value in the **DefCaptionBackStyle** property is applied to the caption resulting in a transparent, or opaque appearance as described above in Settings.

See Also

DefColCaptionBackColor, DefColCaptionBackStyle properties

Example

The following code sets the **CaptionBackStyle** for a **GTList** control's first column header to use the default:

```
GList1.ColumnDefs(0).CaptionBackStyle = gtBackStyleUseDefault
```

CaptionBorderStyle Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the border style to be used for the columndef's caption text.

Usage

object.**CaptionBorderStyle** [= *number*]

The **CaptionBorderStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the style to use for the border on the ColumnDef object's caption

Settings

The settings for *number* are:

Setting	Value	Description
gtBorderStyleNone	0	(Default) None. No border is drawn.
gtBorderStyleSingle	1	Fixed Single. A single border is drawn around the item.
gtBorderStyleInset	2	Inset. The border has a sunken edge.
gtBorderStyleRaised	3	Raised. The border has a raised edge.
gtBorderStyleUseDefault	4	Use Default. Use the value stored in the control's **DefBorderStyle** property

Remarks

When a ColumnDef object is drawn the value stored in this property is applied to the caption text border. If the value is set to **gtBorderStyleUseDefault**, the value in the **DefBorderStyle** for the parent control is used.

See Also

None

Example

The following code sets the **ListItems** first object's **CaptionBorderStyle** to the control's **DefBorderStyle** for a **GTList** control:

```
GTList.ListItems(0).CaptionBorderStyle = GList1.DefBorderStyle
```

CaptionFont Property (ColumnDef object)

Applies To

ColumnDef object

Description

Returns the font used for the Caption.

Usage

*object.***CaptionFont**

Returns a font object used for the ColumnDef object's Caption text. The *object* placeholder is a ColumnDef object.

Remarks

When the ColumnDef object's Caption text is displayed this font is used.

See Also

CaptionFont3D property

Example

The following code sets the ColumnDef object's **CaptionFont** bold value **True** for a **GTCombo** control:

```
GTCombo1.ColumnDefs.Item(0).CaptionFont.Bold = True
```

CaptionFont3D Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the 3D font style used to draw the text for the ColumnDef object's Caption.

Usage

object.**CaptionFont3D** [= *number*]

The **CaptionFont3D** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
number	An integer specifying the 3D style to use when drawing the object's caption text.

Settings

The settings for *number* are:

Setting	Value	Description
gtFont3DNone	0	(Default) None. Text is displayed normally.
gtFont3DRaisedLight	1	Raised Light. The text is slightly raised.
gtFont3DRaisedHeavy	2	Raised Heavy. The text is raised.
gtFont3DInsetLight	3	Inset Light. The text is slightly sunken.
gtFont3DInsetHeavy	4	Inset Heavy. The text is sunken.
gtFont3DUseDefault	5	Use Default. Use the value stored in the **DefFont3D** property.

Remarks

When a Caption object is drawn the value of this property is applied to the text. If the value is **gtFont3DUseDefault**, the value of the parent control's **DefFont3D** property is used.

See Also

CaptionFont property, **DefFont3D** property

Example

The following code sets a ColumnDef object's **CaptionFont3D** to Raised Heavy for a **GTList** control:

```
GTList1.ColumnDefs.Item(0).CaptionFont3D = 2
```

CaptionForeColor Property (ColumnDef object)

Applies To

GTList control**, GTCombo** control

Description

Specifies the forecolor used for the ColumnDef object's caption text.

Usage

*object.***CaptionForeColor** [= *color*]

The **CaptionForecolor** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
color	A value or constant that determines the forecolor color of the text displayed for the object's Caption, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the foreground color value for the text displayed for the object's Caption.

See Also

CaptionBackColor property

Example

The following code sets the object's **CaptionForeColor** to the first ListItem's **ForeColor** for a **GTList** control:

```
GTList1.ColumnDefs.Item(0).CaptionForeColor =
      GTList1.ListItems(0).ForeColor
```

CaptionImage Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the image index used for the ColumnDef object's caption.

Usage

object.**CaptionImage** [= *index*]

The **CaptionImage** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
index	An integer index into the parent control's associated **ImageList**.

Remarks

Use this property to get or set the image for the ColumnDef object's caption.

See Also

ListImages collection, **GTCombo** control, **GTList** Control

Example

The following code sets the columndef's **CaptionImage** to the first image in the parent control's image list for a **GTList** control:

```
GTList1.ColumnDefs.Item(0).CaptionImage = 1
```

CaptionPictureAlignment Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the picture alignment used for the ColumnDef object's caption image.

Usage

object.**CaptionPictureAlignment** [= *number*]

The **CaptionPictureAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
number	A number or value specifying the object's picture alignment as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtPictureAlignmentLeftTop	0	Left Top. Image is left/top aligned.
gtPictureAlignmentLeftMiddle	1	Left Middle. Image is left/middle aligned.
gtPictureAlignmentLeftBottom	2	Left Bottom. Image is left/bottom aligned.
gtPictureAlignmentRightTop	3	Right Top. Image is right/top aligned.
gtPictureAlignmentRightMiddle	4	Right Middle. Image is right/middle aligned.
gtPictureAlignmentRightBottom	5	Right Bottom. Image is right/bottom aligned.
gtPictureAlignmentCenterTop	6	Center Top. Image is center/top aligned.
gtPictureAlignmentCenterMiddle	7	Center Middle. Image is center/middle aligned.
gtPictureAlignmentCenterBottom	8	Center Bottom. Image is center/bottom aligned.
gtPictureAlignmentStretch	9	Stretch to fit. Image is stretched to fit image area.
gtPictureAlignmentTile	10	Tile. Image is tiled in the image area.
gtPictureAlignmentUseDefault	15	Use Default. Use the parent control's **DefColCaptionPictureAlignment** property value.

Remarks

Use this property to get or set the picture alignment for the ColumnDef object's caption image. Setting this property to **gtPictureAlignmentUseDefault** uses the value in the parent control's **DefColCaptionPictureAlignment** property.

See Also

GTList control, **GTCombo** control, **ColumnDefs** collection, **DefColCaptionPictureAlignment** property

Example

The following code sets the object's CaptionPictureAlignment property to Center Bottom for a **GTList** control

```
GTList1.ColumnDefs.Item(0)CaptionPictureAlignment = 8
```

CaptionTextAlignment Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the text alignment used for the ColumnDef object's caption text.

Usage

object.**CaptionTextAlignment** [= *number*]

The **CaptionTextAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
number	A number or value specifying the object's text alignment for the object's caption as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtTextAlignmentLeftTop	0	Left Top. Caption text is left/top aligned.
gtTextAlignmentLeftMiddle	1	Left Middle. Caption text is left/middle aligned.
gtTextAlignmentLeftBottom	2	Left Bottom. Caption text is left/bottom aligned.
gtTextAlignmentRightTop	3	Right Top. Caption text is right/top aligned.
gtTextAlignmentRightMiddle	4	Right Middle. Caption text is right/middle aligned.
gtTextAlignmentRightBottom	5	Right Bottom. Caption text is right/bottom aligned.
gtTextAlignmentCenterTop	6	Center Top. Caption text is center/top aligned.
gtTextAlignmentCenterMiddle	7	Center Middle. Caption text is center/middle aligned.
gtTextAlignmentCenterBottom	8	Center Bottom. Caption text is center/bottom aligned.

Remarks

Use this property to get or set the text alignment for the ColumnDef object.

See Also

ColumnDefs collection, **DefTextAlignment** property

Example

The following code sets the text alignment for a GTCombo's ColumnDef object:

```
Dim xColDef As ColumnDef
xColDef = GTCombo1.ColumnDefs.Item(0)
xColDef.CaptionTextAlignment = gtTextAlignmentUseDefault
```

CaptionWordWrap Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies if the column header's caption will word wrap based on the width of the column header.

Usage

object.**CaptionWordWrap** [= *number*]

The **CaptionWordWrap** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
number	A number or value specifying the word wrap feature as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtWordWrapOff	0	Off. Word wrapping is turned off.
gtWordWrapOn	1	On. Word wrapping is turned on.
gtWordWrapUseDefault	2	Use Default. Use the parent control's **DefWordWrap** property value.

Remarks

Use this property to get or set the word wrap feature for the ColumnDef object's caption text. Setting this property to **gtWordWrapUseDefault** uses the value in the parent control's **DefWordWrap** property.

See Also

GTList control**, GTCombo** control**, ColumnDefs** collection**, DefWordWrap** property

Example

The following code sets the object's **CaptionWordWrap** property to **True** for a **GTList** control

```
GTList1.ColumnDefs.Item(0)CaptionWordWrap = gtWordWrapOn
```

Clear Method

Applies To

ColumnDefs collection**, ListItems** collection**, ListImages** collection**, SubItems** collection

Description

Removes all the objects from the collection rendering the collection empty.

Usage

object.**Clear**

The object placeholder represents an object expression that evaluates to one of the collections listed in the Applies To section.

Remarks

To remove one object from a collection, use the **Remove** method.

See Also

Remove method.

Example

The following example clears the contents of the **GTList** control's **ListItems** collection:

```
GTList1.ListItems.Clear
```

ClickColumn Event

Applies To

GTList control**, GTCombo** control

Description

The **ClickColumn** event occurs when the user clicks in one of the list's columns.

Usage

Private Sub *object*_**ClickColumn(ByRef** *ColumnDef* **As ColumnDef**,
button **As Integer)**

The **ClickColumn** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
ColumnDef	An object expression that evaluates to a **ColumnDef** object. The ColumnDef argument is a ByRef reference to the column that was clicked by the user.
button	Returns an integer that identifies the button that was pressed to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.

Remarks

The **Sorted**, **SortKey**, and **SortOrder** properties are commonly used in code to sort the list using the clicked ColumnDef as the **SortKey**.

See Also

Sorted property, **SortKey** property, **SortOrder** property, **ColumnDefs** collection, **ColumnDef** object.

Example

This example sorts the clicked column in Ascending order for a **GTCombo** control.

```
Private Sub GTCombo1_ClickColumn(ByVal ColumnDef As ColumnDef,
                                 button As Integer)
      GTCombo1.SortOrder = gtSortAscending
      GTCombo1.Sorted = True ' Sort the List.
End Sub
```

ClickItem Event

Applies To

GTList control**, GTCombo** control

Description

The **ClickItem** event occurs when the user clicks one of the items in the control's window.

Usage

Private Sub *object*_**ClickItem(ByVal** *item* **As ListItem**, **ByVal** *SubItem* **As SubItem**, *button* **As Integer)**

The **ClickColumn** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
item	An object expression that evaluates to a **ListItem** object. The item argument is a ByRef reference to the **ListItem** object that was clicked by the user. If a SubItem was clicked this argument will be the ListItem associated with the SubItem object.
SubItem	An object expression that evaluates to a **SubItem** object. The SubItem argument is a ByRef reference to the **SubItem** object that was clicked by the user. This argument will be the first SubItem object in the ListItem object's SubItems collection if the ListItem was clicked.
button	Returns an integer that identifies the button that was pressed to cause the event. The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.

Remarks

Use this event to determine which **ListItem** was clicked. This event is fired before the **Click** event. The standard **Click** event is generated if the mouse is clicked on any part of the control. The **ClickItem** event is generated only when the mouse is clicked on the text or image of a **ListItem** object.

See Also

ListItems collection, **ListItem** object., **SubItems** collection, **SubItem** object.

Example

This example changes the 3D font for the item each time the item is clicked.

```
Private GTList1_ClickItem(ByVal item As ListItem,
                          ByVal SubItem As SubItem,
                          button As Integer)
      Dim nFont3D As Integer
      nFont3D = item.Font3D + 1
      If nFont3D > 3 then
            nFont3D = 0
      End If
      item.Font3D = nFont3D
End Sub
```

ColumnAfterMove Property

Applies To

GTList control**, GTCombo** control

Description

The **ColumnAfterMove** event occurs when the user drops the column he/she is dragging from one column position to another..

Usage

Private Sub *object*_**ColumnAfterMove(ByVal** *ColumnDef* **As ColumnDef**,**)**

The **ColumnAfterMove** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
ColumnDef	An object expression that evaluates to a **ColumnDef** object. The ColumnDef argument is a ByVal argument specifying the column that was moved by the user.

Remarks

This event is generated if the column is moved by the user via drag-drop, or by the **ColumnDef's Move** method.

See Also

Move method **ColumnDefs** collection, **ColumnDef** object., **ColumnBeforeMove** event

Example

This example below displays the column that was moved in a label's caption for a **GTCombo** control.

```
Private Sub GTCombo1_ColumnAfterMove(ByVal ColumnDef As ColumnDef)
      Label1.Caption = Column + ColumnDef.Caption + "was moved."
End Sub
```

ColumnBeforeMove Property

Applies To

GTList control**, GTCombo** control

Description

The **ColumnAfterMove** event occurs when the user drops the column he/she is dragging from one column position to another..

Usage

Private Sub *object*_**ColumnAfterMove(ByVal** *ColumnDef* **As ColumnDef***, Cancel* **As Boolean)**

The **ColumnAfterMove** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
ColumnDef	An object expression that evaluates to a **ColumnDef** object. The ColumnDef argument is a ByVal argument specifying the column that was moved by the user.
Boolean	A boolean value passed by reference indicating if the column move should be canceled.

Remarks

This event is generated if the column is moved by the user via drag-drop, or by the **ColumnDef's Move** method. To cancel the move operation, set the Cancel argument to **True** before returning from the event handler.

See Also

Move method **ColumnDefs** collection, **ColumnDef** object., **ColumnAfterMove** event

Example

This example below cancels the move operation for a GTCombo control:.

```
Private Sub GTCombo1_ColumnBeforMove(ByVal ColumnDef As ColumnDef, Cancel As
Boolean)
      Cancel = True"
End Sub
```

ColumnDefs Collection Methods

Add
Clear
Item
MoveTo (object only)
Remove

ColumnDefs Collection, ColumnDef Object

See Also Properties Methods Constants

The ColumnDefs collection is a standard collection object that holds a series of ColumnDef objects. A ColumnDef object is an item in a GTList or GTCombo control that contains heading text information for the control. The heading text and image appear at the top of the control's window in a window that can be sized by using ColumnDefs properties. The collection supports all of the standard methods associated with collections. Select one of the items above to see the Properties, Methods, and Constants associated with the ColumnDefs collection.

Usage

object.ColumnDefs(*index*)

The syntax above refers to the collection and to individual objects in the collection, respectively, according to the standard collection syntax.

The ColumnDef object defines one column header used in a GTCombo or GTList control. The ColumnDef object has support for:

- Individual background and foreground colors
- Caption background and foreground colors
- Fonts for column objects and column captions

You can define ColumnDef objects both at runtime and at design-time. With a ColumnDef object the user can:

- Click the ColumnDef to trigger a ClickColumn event.
- Grab the objects right border to adjust the width of the column
- Drag and Drop columns to move their position in the control's window.
- Hide the ColumnDef window.

There is always one column in the control window, which is Column 0. This column contains the actual ListItem objects; and not their SubItems. The second column (Column 1) contains SubItems. Therefore, there is always one more ColumnDef object than actual SubItems for the ListItem. The ListItem object's SubItems collection is a 0-based collection. The text in element 0 of the SubItems collection is a mirror of the ListItem object's image and text information.

Listed below are all of the properties and methods supported by the collection and object. Any property or method that is underlined, is standard and not documented here. See the standard Microsoft documentation for information on those elements. Items marked with the symbol () apply only to the collection. All other properties apply only to the object.

Properties

BackColor	BackStyle	Caption
CaptionBackColor	CaptionBackStyle	CaptionBorderStyle
CaptionFont	CaptionFont3D	CaptionForeColor
CaptionImage	CaptionPictureAlignment	CaptionTextAlignment
CaptionWordWrap	Count	DataField
Font	Font3D	ForeColor
Format	Image	Index
Key	MemoryUsed	PictureAlignment
RowHeight	SubColumn	SubItemSource
TagVariant	TextAlignment	Visible
Width	WordWrap	

Methods

Add	Clear	Item

Remove

ColumnDefs Collection, ColumnDef Object Properties

Count (Collection Only)
MemoryUsed (Collection Only)

BackColor
BackStyle
Caption
CaptionBackColor
CaptionBackStyle
CaptionBorderStyle
CaptionFont
CaptionFont3D
CaptionForeColor
CaptionImage
CaptionPictureAlignment
CaptionTextAlignment
CaptionWordWrap
DataField
Font
Font3D
ForeColor
Format
Image
Index
Key
PictureAlignment
RowHeight
SubColumn
SubItemSource
TagVariant
TextAlignment
Visible
Width
WordWrap

ColumnDefs Property

Applies To

GTList control**, GTCombo** control

Description

Returns a reference to a collection of **ColumnDef** objects.

Usage

object.**ColumnDefs**

The object placeholder represents an object expression that evaluates to one of the controls listed in the Applies To section.

Remarks

You can manipulate **ColumnDef** objects using standard collection methods (for example, the **Remove** method). Each **ColumnDef** in the collection can be accessed either by its index or by a unique key, stored in the **Key** property.

See Also

ColumnDefs collection**, ColumnDef** object**, Key** property.

Example

The following example removes the first ColumnDef object from the collection:

```
GTCombo1.ColumnDefs.Remove 1
```

ColumnHeaderHeight Property

Applies To

GTList control, **GTCombo** control

Description

Specifies a fixed height for the column header bar displayed at the top of the control.

Usage

object.**ColumnHeaderHeight** [= *float*]

The **ColumnHeaderHeight** property has these parts:

Part	Description
object	An object expression that evaluates to a **GTList** or **GTCombo** control.
float	A float indicating the height for the column header bar. A value of 0 allows the control to use the **Font** size to determine the height of the column header bar.

Remarks

If the height of this property is not specified, the column header's **Font** property is used to automatically determine the height of the column header. The value for this property is specified using the scale of the container's units of measure.

See Also

ColumnHeaderHeight property, **ColumnDefs** collection, **ColumnDef** object, **Font** property.

Example

The following code changes the **ColumnHeaderHeight** property of a **GTList** control using the default MS Sans Serif font, and specifying the data as Twips:

```
GTList1.ColumnHeaderHeight = 350
```

ColumnResize Event

Applies To

GTList control**, GTCombo** control

Description

The **ColumnResize** event occurs when the size of a ColumnDef object changes.

Usage

Private Sub *object*_**ColumnResize(ByRef** *ColumnDef* **As ColumnDef**
width **As Single** *canel* **As Boolean)**

The **ColumnResize** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
ColumnDef	An object expression that evaluates to a **ColumnDef** object. The ColumnDef argument is a ByRef reference to the column that was moved by the user.
width	Number specifying the new width of the ColumnDef object.
cancel	A boolean expression indicating if the ColumnResize operation should be canceled. Set this value to True if you want to cancel the operation.

Remarks

Use the ColumnResize event to recalculate local variables based on the size of the column, or to repaint objects that may be affected by the resize.

See Also

AllowColumnResize property, **ColumnDefs** collection, **ColumnDef** object.

Example

This example changes the background color of the ColumnDef object to Red if the column's width gets greater than 500 for a **GTCombo** control when the user moves the column.

```
Private Sub GTCombo1_ColumnMove(ColumnDef As ColumnDef,
                                height As Integer,
                                width As Integer)
      If width > 500 then
            columndef.Backcolor = RGB(255,0,0)
      Else
            columndef.Backcolor = GTCombo1.DefBackcolor
      End If
End Sub
```

ColumnResizeElastic Property

Applies To

GTList control**, GTCombo** control

Description

Determines if the column to the right of the column being resized, is automatically resized to the difference between the old size and the new size. This has the net effect of preserving the width of the sum of the columns.

Usage

*object.***ColumnResizeElastic** [= *boolean*]

The **ColumnResizeElastic** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the column to the right of the column being resized, is automatically resized to keep the net width.

Remarks

If this property is true, and the user re-sizes column 1 from 150 to 125, the difference of 25 is added to the width column 2. If the user increases column 1 from 125 to 150, column 2 is resized by subtracting 25 from its width automatically. Having the columns be elastic, preserves the net width of the sum of the control's columns widths.

See Also

AllowColumnResize property, **ColumnResize** event.

Example

The following code changes the **ColumnResizeElastic** property of a **GTList** control:

```
GTList1.ColumnResizeElastic = False
```

ComboStyle Property

Applies To

GTCombo control

Description

Returns or sets a value indicating the type of **GTCombo** control and the behavior of its list box portion. Read only at run time

Usage

*object.***ComboStyle**

The object placeholder represents an object expression that evaluates to a **GTCombo** control.

Settings

The ComboStyle settings are:

Setting	Value	Description
gtStyleCombo	0	(Default) Dropdown Combo. Includes a drop-down list and a text box. The user can select from the list or type in the text box.
gtStyleSimple	1	Simple Combo. Includes a text box and a list, which doesn't drop down. The user can select from the list or type in the text box. The size of a Simple combo box includes both the edit and list portions. By default, a Simple combo box is sized so that none of the list is displayed. Increase the **Height** property to display more of the list..
gtStyleList	2	Dropdown List. This style only allows selection from the drop-down list.

Remarks

Follow these guidelines in deciding which setting to choose:

Use setting 0 (Dropdown Combo) or setting 1 (Simple Combo) to give the user a list of choices. Either style enables the user to enter a choice in the text box. Setting 0 saves space on the form because the list portion closes when the user selects an item.

Use setting 2 (Dropdown List) to display a fixed list of choices from which the user can select one. The list portion closes when the user selects an item.

See Also

DropDown event

Example

The following code gets the **ComboStyle** and displays it as a label's caption:

```
Label1.Caption = "ComboStyle: " + STR$(GTCombo1.ComboStyle)
```

DataList™

What are DataList ActiveX controls?

A description of DataList ActiveX control features.

Quick Tour - Writing Sample Programs

Guides you through some sample programs using the DataList ActiveX controls.

Using GTList Control (Control Reference)

How to use GTList to write applications, including the control's overview and reference.

Using GTCombo (Control Reference)

How to use GTCombo to write applications, including the control's overview, and reference.

Technical Specifications

Technical requirements for running DataList ActiveX controls.

Included Files

Files installed by the DataList Setup program.

Error Messages

Trappable Error messages generated by the DataList ActiveX controls.

Technical Support

Access numbers for GreenTree Technologies Inc. Technical Support.

Copyright (c) 1996, GreenTree Technologies, Inc.

This document was produced using ForeHelp 2.1.1

DataList ActiveX 1.0
August 1996 (First Printing)

Help File Version: 1.01

Count Property
(ColumnDefs, ListItems, ListImages SubItems collection)

Applies To

ColumnDefs collection**, ListItems** collection**, ListImages collection, SubItems** collection

Description

Returns the number of members in the collection.

Usage

object.**Count**

The object placeholder represents an object expression that evaluates to one of the collections listed in the Applies To section.

Remarks

The **Count** property is associated with the collection and not the control itself.

See Also

MemoryUsed property.

Example

The following example removes the objects from a ColumnDefs collection:

```
Dim nCount As Integer

nCount = GTCombo1.ColumnDefs.Count
For I = 0 to nCount
      GTCombo1.ColumnDefs.Remove 1
Next I
```

DataChanged Property

Applies To:

GTCombo control

Description

Returns or sets a value indicating that the data in the bound control has been changed by some process other than that of retrieving data from the current record. Not available at design time.

Usage

object.**DataChanged** [= *value*]

The DataChanged property syntax has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	A Boolean expression that indicates whether data has changed, as described in Settings.

Settings

The settings for value are:

Setting	Description
True	The data currently in the control isn't the same as in the current record.
False	(Default) The data currently in the control, if any, is the same as the data in the current record.

Remarks

When a Data control moves from record to record, it passes data from fields in the current record to controls bound to the specific field or the entire record. As data is displayed in the bound controls, the DataChanged property is set to False. If the user or any other operation changes the value in the bound control, the DataChanged property is set to True. Simply moving to another record doesn't affect the DataChanged property.

When the Data control starts to move to a different record, the Validate event occurs. If DataChanged is True for any bound control, the Data control automatically invokes the Edit and Update methods to post the changes to the database.

If you don't wish to save changes from a bound control to the database, you can set the DataChanged property to False in the Validate event.

Inspect the value of the DataChanged property in your code for a control's Change event to avoid a cascading event. This applies to both bound and unbound controls.

See Also

GTList control, **GTCombo** control, **DataSource** property

Example

The example below indicates to the user that the data in the control has changed from the current record:

```
If DataChanged = True Then
      Label1.Caption = "Data Changed"
Else
      Label1.Caption = "Data is not Changed"
End If
```

DataField Property (ColumnDef object)

Applies To

ColumnDef object

Description

Returns or sets a value that binds a **Field** object to the ColumnDef object.

Usage

object. **DataField** [= *value*]

The **DataField** property has these parts:

Part	Description
object	An object expression that evaluates to one a ColumnDef object.
value	A string expression that evaluates to the name of one of the fields in the Recordset object specified by the parent control's **DataSource** property.

Remarks

The ColumnDef objects of a **GTList** or **GTCombo** control can be bound to a database field. The value of this property specifies the field name from the DataSource record structure. The data from the **DataSource** field will be displayed

See Also

GTList control, **GTCombo** control, **DataSource** property

Example

The example below sets the **DataField** to the `Titles` field. This example assumes that a **DataSource** exists and that the data source has a Titles field in its record structure.

```
GTCombo1.DataField = "Titles"
```

DataFieldDisplay Property

Applies To

GTCombo control

Description

Returns or sets a value that binds a **Field** from the **DataSource** to the edit portion of the combo box.

Usage

object. **DataFieldDisplay** [= *value*]

The **DataFieldDisplay** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	A string expression that evaluates to the name of one of the fields in the Recordset specified by the **DataSource** property. The value of the field is stored in the `GTCombo.Text` property.

Remarks

The **GTCombo** control's edit box portion can be bound to a separate database and database field. This property specifies which field in the current data record is used as the text for the edit box portion of the combo box. This data field is separate from the **DataFieldList** for the list portion of the control. Therefore, you can have a list of descriptive items in the list, and edit a different part of the same record or a related record in a separate database when the list item is selected. You might want to have product names in the list, and edit their unit prices when the product name is selected from the list.

See Also

DataSourceList, DataField properties

Example

The example below sets the **DataFieldDisplay** to the `UnitPrice` field. This example assumes that a **DataSource** exists and that the data source has a UnitPrice field in its record structure.

```
GTCombo1. DataFieldDisplay = "UnitPrice"
```

DataFieldList Property

Applies To

GTCombo control

Description

Returns or sets a value that binds a **Field** object from a data source to the list portion of the combo box.

Usage

object. **DataFieldList** [= *value*]

The **DataFieldList** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	A string expression that evaluates to the name of one of the fields in the Recordset object specified by the **DataSourceList** property. The value of the field is stored in the `GTCombo.DisplayText` property.

Remarks

The **GTCombo** control's list box portion can be bound to a database field. This property specifies which field in the current data record is used as the listitem's text for the list box portion of the combo box. This data field is separate from the DataField in the control. Therefore, you can have a list of descriptive items in the list, and edit a different part of the same record, or a different record in a different database when the list item is selected. You might want to have product names in the list, and edit their unit prices when the product name is selected in the list.

See Also

DataSourceList, DataFieldDisplay properties

Example

The example below sets the **DataFieldList** to the `ProductName` field. This example assumes that a **DataSourceList** exists and that the data source has a ProductName field in its record structure.

```
GTCombo1. DataFieldList = "ProductName"
```

DataSource Property

Applies To

GTList control

Description

Sets a value that specifies the Data control through which the current **DataList** control is bound to a database. Not available at run time.

Usage

*object.***DataSource**

The **DataSource** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.

Remarks

To bind a control to a database at run time, you must specify a Data control in the **DataSource** property at design time using the Properties window.

Unlike the stock **DataSource** property there is no need to complete the connection with a database in the Recordset managed by the Data control, by providing the name of a Field object in a **DataField** property. All fields for the **RecordSet** are added to the control, one for each column.

See Also

DataSourceList, DataChanged properties.

Example

The **DataSourceList** property is not available at runtime, therefore no example is provided.

DataSource and DataSourceList

The **DataSource** property binds the edit box portion of the control to a Data control in your application. The Data control provides access to data stored in databases using any one of three types of Recordset objects. The Data control enables you to move from record to record and to display and manipulate data from the records in bound controls. Without a Data control, data-aware (bound) controls like **GTCombo** can't automatically access data. Once **GTCombo** is bound to a database, the control can display the data in the database records, indicate how many records are in the database through the **ListCount** property, edit, modify, delete, and update the database records, sort the database records using a hierarchy of keys and order the output. Show additional database record fields in the **SubItems** collection. The control does all of this activity for you, you don’t have to write any code. Once the control is bound to a database, it can no longer be virtual.

DataSourceList Property

Applies To

GTCombo control

Description

Sets a value that specifies the Data control through which the current **DataList** control is bound to a database. Not available at run time.

Usage

*object.***DataSourceList**

The **DataSourceList** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.

Remarks

To bind a control to a database at run time, you must specify a Data control in the **DataSourceList** property at design time using the Properties window.

Unlike the stock **DataSource** property there is no need to complete the connection with a database in the Recordset managed by the Data control, by providing the name of a Field object in a **DataField** property. All fields for the **RecordSet** are added to the control, one for each column.

See Also

DataSource, DataChanged properties.

Example

The **DataSourceList** property is not available at runtime, therefore no example is provided.

DisplayText Property

Applies To

GTCombo control

Description

Contains the text that is displayed in the edit portion of the GTCombo control.

Usage

object.**DisplayText** [= *string*]

The DisplayText property syntax has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
string	A string expression specifying text.

Remarks

This property sets and retrieves the text in the edit portion of the control.

See Also

ListItems collection, **ListItem** object, DataSource property

Example

The following code gets the DisplayText for a GTCombo control:

```
Label1.Caption = GTCombo1.DisplayText
```

DefBackColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default BackColor used for all list items.

Usage

*object.***DefBackColor** [= *color*]

The **DefBackColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the background color of the list item, as described in Settings.

Settings

The settings for *color* are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

When a list item object is drawn by the **GTList** or **GTCombo** control, and the object's BackColor is set to Use Default, this properties color is applied to the object. By default all list item's background colors are drawn using this property. However, if the DefBackColorOdd property is set to a different color, all even indexed items are drawn using this color, and all odd indexed items are drawn using the **DefBackColorOdd** property color value.

See Also

DefBackStyle property**, ListItems** collection**, ListItem** object**, ColumnDefs** collection**, ColumnDef** object, **SubItems** collection, **SubItem** object.

Example

The following code sets the **DefBackColor** for a **GTList** control to Green:

```
GTList1.DefBackColor = RGB(0, 255, 0)
```

DefBackColorOdd Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default BackColor used for all list items that appear on odd numbered lines.

Usage

object.**DefBackColorOdd** [= *color*]

The **DefBackColorOdd** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the background color of the list item, as described in Settings.

Settings

The settings for *color* are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.
gtUseDefaultColor	Uses the color set in the **DefBackColor** property for odd items.

Remarks

When an odd indexed list item object is drawn by the **GTList** or **GTCombo** control, and the object's BackColor is set to Use Default, this properties color is applied to the object.

See Also

DefBackColor property.

Example

The following code sets the **DefBackColorOdd** for a **GTList** control to Green:

```
GTList1.DefBackColorOdd = RGB(0, 255, 0)
```

DefBackStyle Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default background style to be used when list item objects are drawn in the control.

Usage

*object.***DefBackStyle** [= *number*]

The **DefBackStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the default background style to use when drawing list item objects in the control's window.

Settings

The settings for number are:

Setting	Value	Description
gtBackStyleTransparent	0	Transparent. Draws list item objects transparently.
gtBackStyleOpaque	1	(Default) Opaque. Draws list item objects opaquely.
gtBackStyleUseDefault	2	Use Default. Applies only to collection objects.

Remarks

When a list item object is drawn by the **GTList** or **GTCombo** control, and the object's BackStyle is set to **gtBackStyleUseDefault**, this property's value is applied to the object resulting in a transparent, or opaque appearance as described above in Settings.

See Also

DefBackColor property**, ListItems** collection**, ListItem** object

Example

The following code sets the **DefBackStyle** for a **GTList** control to Opaque:

```
GList1.DefBackStyle = gtBackStyleOpaque
```

DefColCaptionBackColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default BackColor used for all captions in the column header bar.

Usage

*object.***DefColCaptionBackColor** [= *color*]

The **DefColCaptionBackColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the background color of the column header bar's captions, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the default background color value for the column header captions used for the control.

See Also

ColumnDefs collection**, ColumnDef** object**, DefColCaptionBorderStyle** property**, DefColCaptionFont** property, **DefColCaptionForeColor** property, **ListItems** collection**, ListItem** object**, GTList** control.

Example

The following code sets the **DefColCaptionBackColor** for a **GTList** control to Green:

```
GList1.DefColCaptionBackColor = RGB(0, 255, 0)
```

DefColCaptionBackStyle Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default background style to be used for all column header captions displayed in the control.

Usage

*object.***DefColCaptionBackStyle** [= *number*]

The **DefColCaptionBackStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the default background style to use when drawing the column header captions.

Settings

The settings for number are:

Setting	Value	Description
gtBackStyleTransparent	0	Transparent. Draws column header captions objects transparently.
gtBackStyleOpaque	1	(Default) Opaque. Draws column header captions opaquely.

Remarks

When a column header caption is drawn by the **GTList** or **GTCombo** control, and the caption's **BackStyle** is set to **gtBackStyleUseDefault**, this property's value is applied to the caption resulting in a transparent, or opaque appearance as described above in Settings.

See Also

DefColCaptionBackColor property**, ColumnDefs** collection**, ColumnDef** object

Example

The following code sets the **DefColCaptionBackStyle** for a **GTList** control to Opaque:

```
GList1.DefColCaptionBackStyle = gtBackStyleOpaque
```

DefColCaptionBorderStyle Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default border style to be used for all column header captions displayed in the control.

Usage

*object.***DefColCaptionBorderStyle** [= *number*]

The **DefColCaptionBorderStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the default style to use when drawing column header captions in the control's window.

Settings

The settings for *number* are:

Setting	Value	Description
gtBorderStyleNone	0	(Default) None. No border is drawn.
gtBorderStyleSingle	1	Fixed Single. A single border is drawn around the captions.
gtBorderStyleInset	2	Inset. The border has a sunken edge.
gtBorderStyleRaised	3	Raised. The border has a raised edge.

Remarks

When a column header caption is drawn by the **GTList** or **GTCombo** control, and the column header's **BorderStyle** is set to **gtBorderStyleUseDefault**, the value of this property is applied to the caption resulting in the appearance described in Settings.

See Also

ColumnDefs collection**, ColumnDefs** object**, DefColCaptionBackColor** property, **DefColCaptionFont** property**, ListItems** collection**, ListItem** object

Example

The following code sets **DefColCaptionBorderStyle** for a **GTList** control to None:

```
GTList1.ColumnDefs(0).BorderStyle = gtBorderStyleNone
```

DefColCaptionFont Property

Applies To

GTList control**, GTCombo** control

Description

Returns the default column headers font used for all the captions.

Usage

*object.***DefColCaptionFont**

Returns a font object used for all the column headers captions. The *object* placeholder can be any control listed in the Applies To section.

Remarks

When a column header caption text is displayed by the **GTList** or **GTCombo** control, this default Font used when drawing the caption.

See Also

ColumnDefs collection**, ColumnDefs** object**, DefColCaptionBackColor** property, **DefColCaptionBorderStyle** property**, ListItems** collection**, ListItem** object.

Example

The following code sets the **DefColCaptionFont**'s bold attribute to **True** for a **GTList** control:

```
GList1.DefColCaptionFont.Bold = True
```

DefColCaptionFont3D Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default 3D font style used when displaying the column headers caption.

Usage

*object.***DefColCaptionFont3D** [= *number*]

The **DefColCaptionFont3D** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the default 3D style to use when drawing column header captions in the control's window.

Settings

The settings for *number* are:

Setting	Value	Description
gtFont3DNone	0	(Default) None. Text is displayed normally.
gtFont3DRaisedLight	1	Raised Light. The text is slightly raised.
gtFont3DRaisedHeavy	2	Raised Heavy. The text is raised.
gtFont3DInsetLight	3	Inset Light. The text is slightly sunken.
gtFont3DInsetHeavy	4	Inset Heavy. The text is sunken.

Remarks

When a column header caption is drawn by the **GTList** or **GTCombo** control, and the column header's **Font3D** property is set to **gtFont3DUseDefault**, the value of this property is applied to the caption's text.

See Also

ColumnDefs collection**, ColumnDefs** object**, DefColCaptionFont** property**, ListItems** collection**, ListItem** object

Example

The following code sets the control's **DefColCaptionFont3D** to Inset Light for a **GTList** control:

```
GTList1.DefColCaptionFont3D = gtFont3DInsetLight
```

DefColCaptionForeColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default forecolor used for all captions in the column header bar.

Usage

*object.***DefColCaptionForeColor** [= *color*]

The **DefColCaptionForeColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the forecolor color of the column header bar's captions, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the default foreground color value for the column header captions used for the control.

See Also

ColumnDefs collection**, ColumnDef** object**, DefColCaptionBorderStyle** property**, DefColCaptionFont** property, **DefColCaptionBackColor** property, **ListItems** collection, **ListItem** object

Example

The following code sets the **DefColCaptionForeColor** for a **GTList** control to Black:

```
GTList1.ColumnDefs(0).ForeColor = RGB(0, 0, 0)
```

DefColCaptionPictureAlignment Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default picture alignment used for all caption images in the column header bar.

Usage

*object.***DefColCaptionPictureAlignment** [= *number*]

The **DefColCaptionPictureAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the caption picture's alignment as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtPictureAlignmentLeftTop	0	Left Top. Image is left/top aligned.
gtPictureAlignmentLeftMiddle	1	Left Middle. Image is left/middle aligned.
gtPictureAlignmentLeftBottom	2	Left Bottom. Image is left/bottom aligned.
gtPictureAlignmentRightTop	3	Right Top. Image is right/top aligned.
gtPictureAlignmentRightMiddle	4	Right Middle. Image is right/middle aligned.
gtPictureAlignmentRightBottom	5	Right Bottom. Image is right/bottom aligned.
gtPictureAlignmentCenterTop	6	Center Top. Image is center/top aligned.
gtPictureAlignmentCenterMiddle	7	Center Middle. Image is center/middle aligned.
gtPictureAlignmentCenterBottom	8	Center Bottom. Image is center/bottom aligned.
gtPictureAlignmentStretch	9	Stretch to fit. Image is stretched to fit image area.
gtPictureAlignmentTile	10	Tile. Image is tiled in the image area.
gtPictureAlignmentLeftOfText	11	(Default) Left Of Text. Image is placed left of the item's text.
gtPictureAlignmentRightOfText	12	Right Of Text. Image is placed right of the item's text.
gtPictureAlignmentAboveText	13	Above Text. Image is placed above the Item's text.
gtPictureAlignmentBelowText	14	Below Text. Image is placed below the item's text.

Remarks

Use this property to get or set the default picture alignment for the column header object images.

See Also

ColumnDefs collection, **ColumnDef** object, **ListImages** collection, **ListImage** object, **ListItems** collection, **ListItem** object

Example

The following code sets the **ColumnDef** object's picture alignment to the control's **DefColCaptionImage** for a **GTList** control. Setting its value to 15 explicitly means use the value in DefColCaptionImage:

```
GTList1.ColumnDefs(0).ColCaptionImageAlignment = 15
```

DefColCaptionTextAlignment Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default text alignment used for all captions in the column header bar.

Usage

*object.***DefColCaptionTextAlignment** [= *number*]

The **DefColCaptionTextAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the caption text alignment as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtTextAlignmentLeftTop	0	Left Top. Caption text is left/top aligned.
gtTextAlignmentLeftMiddle	1	Left Middle. Caption text is left/middle aligned.
gtTextAlignmentLeftBottom	2	Left Bottom. Caption text is left/bottom aligned.
gtTextAlignmentRightTop	3	Right Top. Caption text is right/top aligned.
gtTextAlignmentRightMiddle	4	Right Middle. Caption text is right/middle aligned.
gtTextAlignmentRightBottom	5	Right Bottom. Caption text is right/bottom aligned.
gtTextAlignmentCenterTop	6	Center Top. Caption text is center/top aligned.
gtTextAlignmentCenterMiddle	7	Center Middle. Caption text is center/middle aligned.
gtTextAlignmentCenterBottom	8	Center Bottom. Caption text is center/bottom aligned.

Remarks

Use this property to get or set the default text alignment for the column header object text.

See Also

ColumnDefs collection**, ColumnDef** object**, ListItems** collection**, ListItem** object

Example

The following code sets the default column header's caption alignment to the Center Top for a **GTList** control:

```
GTList1.DefColCaptionTextAlignment = gtTextAlignmentCenterTop
```

DefColCaptionWordWrap Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default value for the column header's caption word wrap feature.

Usage

object.**DefColCaptionWordWrap** [= *number*]

The **DefColCaptionWordWrap** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
number	A number or value specifying the word wrap feature as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtWordWrapOff	0	Off. Column header captions are not word wrapped.
gtWordWrapOn	1	On. Column header captions are word wrapped.

Remarks

Use this property to get or set the default value for the ColumnDef object's word wrap feature. Setting the ColumnDef object's **CaptionWordWrap** property to **gtWordWrapUseDefault** uses the value set in this property.

See Also

ColumnDefs collection**, ColumnDef** object, **CaptionWordWrap** property

Example

The following code sets the object's **DefColCaptionWordWrap** property to **True** for a **GTList** control

```
GTList1.DefColCaptionWordWrap = gtWordWrapOn
```

DefColumnWidth Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default column width used for all list items in the control.

Usage

*object.***DefColumnWidth** [= *float*]

The **DefColColumnWidth** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
float	A float indicating the width for each item in the control. A value of 0 evenly spaces each item.

Remarks

Use this property to get or set the default column width for all the objects displayed in the control. The value for this property is specified using the container's units of measure.

See Also

DefRowHeight property, **ListItems** collection**, ListItem** object

Example

The following code sets the default column header's width to a specific value for a **GTList** control:

```
GTList1.DefColumnWidth = 74.3
```

DefFont Property

Applies To

GTList control**, GTCombo** control

Description

Returns the default font used for all objects in the control.

Usage

*object.***DefFont**

Returns a font object used for all the list items. The *object* placeholder can be one of the controls listed in the Applies To section.

Remarks

When an item displayed by the **GTList** or **GTCombo** control, this default Font used.

See Also

DefFont3D property, **DefColCaptionBorderStyle** property**, ListItems** collection**, ListItem** object

Example

The following code sets the default font's bold value **True** for a **GTList** control:

```
GTList1.DefFont.Bold = True
```

DefFont3D Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default 3D font style used to draw the text for all list items displayed in the control.

Usage

*object.***DefFont3D** [= *number*]

The **DefFont3D** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the default 3D style to use when drawing text for list items.

Settings

The settings for *number* are:

Setting	Value	Description
gtFont3DNone	0	(Default) None. Text is displayed normally.
gtFont3DRaisedLight	1	Raised Light. The text is slightly raised.
gtFont3DRaisedHeavy	2	Raised Heavy. The text is raised.
gtFont3DInsetLight	3	Inset Light. The text is slightly sunken.
gtFont3DInsetHeavy	4	Inset Heavy. The text is sunken.

Remarks

When a list item object is drawn by the **GTList** or **GTCombo** control, and the item's **Font3D** property is set to **gtFont3DUseDefault**, the value of this property is applied to the item's text.

See Also

DefFont property**, ListItems** collection**, ListItem** object

Example

The following code sets a control's default 3D font (**DefFont3D)** to Raised Heavy for a **GTList** control:

```
GTList1.DefFont3D = gtFont3DRaisedHeavy
```

DefForeColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default forecolor used for all list items in the control.

Usage

*object.***DefForeColor** [= *color*]

The **DefForeColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the forecolor color of the text displayed for all list items as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the default foreground color value for the text displayed for each list item. This property value applies to all SubItems in **GTList** controls. If the value in the DefForeColorOdd property is different than this property, the odd indexed list items are displayed using the foreground color defined in the **DefForeColor** property.

See Also

DefBackColor, DefForeColorOdd properties**, ListItems** collection**, ListItem** object

Example

The following code sets the control's **DefForeColor** to Black for a **GTList** control:

```
GTList1.DefForeColor = RGB( 0, 0, 0)
```

DefForeColorOdd Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default forecolor used for all odd numbered list items in the control.

Usage

*object.***DefForeColorOdd** [= *color*]

The **DefForeColorOdd** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the forecolor color of the text displayed for all list items as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.
gtUseDefaultColor	Uses the value in the DefForeColor property as the foreground color for odd numbered list items.

Remarks

Use this property to get or set the default foreground color value for the text displayed for each odd indexed list item. This property value applies to all odd indexed SubItems in **GTList** controls.

See Also

DefBackColor, DefBackColorOdd property**, ListItems** collection**, ListItem** object

Example

The following code sets the control's **DefForeColorOdd** to Black for a **GTList** control:

```
GTList1.DefForeColorOdd = RGB( 0, 0, 0)
```

DefPictureAlignment Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default picture alignment used for all list item images.

Usage

object.**DefPictureAlignment** [= *number*]

The **DefPictureAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the list item object's picture alignment as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtPictureAlignmentLeftTop	0	Left Top. Image is left/top aligned.
gtPictureAlignmentLeftMiddle	1	Left Middle. Image is left/middle aligned.
gtPictureAlignmentLeftBottom	2	Left Bottom. Image is left/bottom aligned.
gtPictureAlignmentRightTop	3	Right Top. Image is right/top aligned.
gtPictureAlignmentRightMiddle	4	Right Middle. Image is right/middle aligned.
gtPictureAlignmentRightBottom	5	Right Bottom. Image is right/bottom aligned.
gtPictureAlignmentCenterTop	6	Center Top. Image is center/top aligned.
gtPictureAlignmentCenterMiddle	7	Center Middle. Image is center/middle aligned.
gtPictureAlignmentCenterBottom	8	Center Bottom. Image is center/bottom aligned.
gtPictureAlignmentStretch	9	Stretch to fit. Image is stretched to fit image area.
gtPictureAlignmentTile	10	Tile. Image is tiled in the image area.
gtPictureAlignmentLeftOfText	11	(Default) Left Of Text. Image is placed left of the item's text.
gtPictureAlignmentRightOfText	12	Right Of Text. Image is placed right of the item's text.
gtPictureAlignmentAboveText	13	Above Text. Image is placed above the Item's text.
gtPictureAlignmentBelowText	14	Below Text. Image is placed below the item's text.

Remarks

Use this property to get or set the default picture alignment for all the list item object images.

See Also

ListImages collection, **ListImage** object, **ListItems** collection**, ListItem** object

Example

The following code sets the control's default picture alignment Center Bottom for a **GTList** control

```
GTList1.DefPictureAlignment = 8
```

DefRowHeight Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default row height used for all list items in the control.

Usage

*object.***DefRowHeight** [= *float*]

The **DefRowHeight** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
float	A float indicating the row height for each row of items in the control. A value of 0 allows the **Font** height to determine the row height.

Remarks

Use this property to get or set the default row height for all the list items displayed in the control. This value is specified in the container's units of measure.

See Also

DefColumnWidth property, **ListItems** collection**, ListItem** object

Example

The following code sets the default row height to a specific twips value for a **GTList** control:

```
GTList1.DefRowHeight = 400
```

DefTextAlignment Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the default text alignment used for all list item objects.

Usage

*object.***DefTextAlignment** [= *number*]

The **DefTextAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the list item's text alignment for all the control's objects as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtTextAlignmentLeftTop	0	Left Top. Caption text is left/top aligned.
gtTextAlignmentLeftMiddle	1	Left Middle. Caption text is left/middle aligned.
gtTextAlignmentLeftBottom	2	Left Bottom. Caption text is left/bottom aligned.
gtTextAlignmentRightTop	3	Right Top. Caption text is right/top aligned.
gtTextAlignmentRightMiddle	4	Right Middle. Caption text is right/middle aligned.
gtTextAlignmentRightBottom	5	Right Bottom. Caption text is right/bottom aligned.
gtTextAlignmentCenterTop	6	Center Top. Caption text is center/top aligned.
gtTextAlignmentCenterMiddle	7	Center Middle. Caption text is center/middle aligned.
gtTextAlignmentCenterBottom	8	Center Bottom. Caption text is center/bottom aligned.

Remarks

Use this property to get or set the default text alignment all the list items in the control.

See Also

ListItems collection**, ListItem** object, **TextAlignment** property

Example

The following code sets the default text alignment for all of the control's list items to the Right Top for a **GTList** control:

```
GTList1.DefTextAlignment = gtTextAlignmentRightTop
```

DefWordWrap Property

Applies To

GTCombo control, **GTList** control

Description

Specifies the default word wrap mode for object's data when the object's WordWrap property is set to gtWordWrapUseDefault.

Usage

object. **DefWordWrap** [= *number*]

The **DefWordWrap** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the default word wrap value as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtWordWrapOff	0	Off. No word wrapping take place.
gtWordWrapOn	1	On. Word wrapping is enabled.

Remarks

Use this property to get or set the default word wrap value used when an object has their WordWrap property set to gtWordWrapUseDefault. This property cannot use the value gtWordWrapUseDefault.

See Also

WordWrap property

Example

The following code sets the control's Def**WordWrap** property to **True** for a **GTList** control

```
GTList1.DefWordWrap = gtWordWrapOn
```

Drag Method

Applies To

GTList control**, GTCombo** control

Description

Begins, ends, or cancels a drag operation on the **GTList** or **GTCombo** control. Doesn't support named arguments.

Usage

object.**Drag** *action*

The **Drag** method has these parts:

Part	Description
object	Required. An object expression that evaluates to one of the objects or controls in the Applies To list.
action	Optional. A constant or value that specifies the action to perform, as described in Settings. If action is omitted, the default is to begin dragging the object.

Settings

The settings for *action* are:

Value	Description
0	Cancel the current dragging operation for the control.
1	(Default) Begin a dragging operation for the control.
2	End the dragging operation for the control.

Remarks

These constants are listed in the object library in the Object Browser.

Using the **Drag** method to control a drag-and-drop operation is required only when the **DragMode** property of the object is set to Manual (0). However, you can use Drag on an object whose **DragMode** property is set to Automatic (1).

If you want the mouse pointer to change shape while the object is being dragged, use either the **DragIcon** or **MousePointer** property. The **MousePointer** property is only used if no **DragIcon** is specified.

See Also

DragMode property, **DragIcon** property, **MousePointer** property.

Example

This example begins a drag operation for a **GTCombo** control.

```
GTCombo1.Drag 1
```

DropDown Event

Applies To

GTCombo control

Description

Occurs when the list portion of a **GTCombo** control is about to drop down; this event doesn't occur if a **GTCombo** control's **ComboStyle** property is set to 1 (Simple).

Usage

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

The **DropDown** event has these parts:

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

Remarks

Use a **DropDown** event procedure to make final updates to a GTCombo list before the user makes a selection. This enables you to add or remove items from the list using the **AddItem** or **RemoveItem** methods

See Also

AddItem method, **RemoveItem** method, **ClickItem** event, **ClickColumn** event.

Example

This example removes each item from the control, then places new items based on an option button selected by the user for a **GTCombo** control.

```
Private Sub GTCombo1_DropDown ()
      While GTCombo1.ListCount      ' Delete existing items.
            GTCombo1.RemoveItem 0
      Wend
      If OptionGroup(0).Value = True Then
            GTCombo1.AddItem "Slate Bricks", 0
            GTCombo1.AddItem "Terra Cotta Bricks", 1
      Else
            GTCombo1.AddItem "Slate Stacking Stone"
            GTCombo1.AddItem "Terra Cotta Stacking Stone"
      End If
End Sub
```

DropDownWidth Property

Applies To

GTCombo control

Description

Returns or sets the width of the drop down window for the GTCombo control.

DropDownWidth is always in the same unit of measure as the container for the object.

Usage

object. **DropDownWidth** [= *number*]

The **DropDownWidth** property has these parts:

Part	Description
object	An object expression that evaluates to a GTCombo control..
number	A number or value specifying the width of the drop down window specified in the container's units of measure..

Remarks

The minimum DropDownWidth is 1 pixel. Use this property to get or set the width of the drop down window for the control. Setting this property to 0 causes the control to calculate the width of the drop down window automatically.

See Also

MinDropDownItems, **MaxDropDownItems** properties

Example

The following code sets the DropDownWidth property to a specific value in twips for a GTCombo control:

```
GTCombo1.DropDownWidth = 2400
```

DynamicCols Property

Applies To

GTList control**, GTCombo** control

Description

Specifies if the columns in the list are automatically re-added to the list when a new data source is introduced to the control.

Usage

object.**DynamicCols** [= *number*]

The **DynamicCols** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying if the columns in the list are automatically resized as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtNever	0	Never. Never Re-add the columns on a data source change.
gtInitialOnly	1	Initial DataSource only. Re-add the columns when the data source changes initially.
gtAlways	2	Every DataSource change. Re-add the columns whenever the data source changes.

Remarks

Use this property to control when columns are re-added to the list based on a data source change.

See Also

ColumnDefs collection**, ColumnDef** object, **ColumnResizeElastic** property.

Example

The following code sets the dynamic columns property to initial only for a **GTList** control:

```
GTList1.DynamicCols = gtDynamicColsInitialOnly
```

Error Messages

The following is a list of trappable errors that could occur at runtime when using the DataList custom controls.

Error Number	Description
30001	“Cannot retrieve SubItem(0)”. SubItem 0 has been deleted by a previous operation.
30002	“Failed to get external ImageList data”. The external image list no longer exists.

Exercise 2: Customizing the look and feel

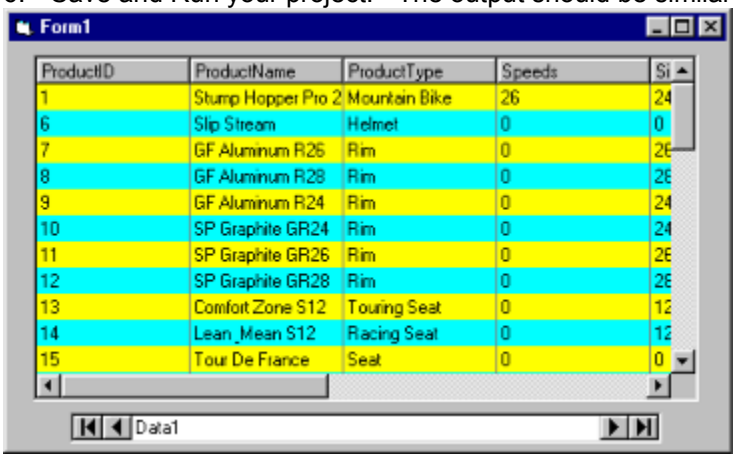
See Also

If you haven't completed exercise one, do so now, it only takes about 2 minutes, and will give you a taste of how powerful *GreenTree's DataList* controls really are. This exercise expands on what you accomplished in exercise one.

1. For this exercise, start a new Visual Basic project and place a **GTList** control and a **Data** control on the default Form.
2. Set the **DatabaseName** property of the **Data** control to:

   ```
   C:\DATALIST\TOPROCK.MDB
   ```

   This directory may be different if you have installed the *GreenTree DataList* controls into a different directory other than the default given in the *DataList* setup program.
3. Set the Data control's RecordSource property to "Products". This identifies what table in the database will contain the database records.
4. Click on the **GTList** control, and set the **DataSource** property to "`Data1`".
5. Set **Backcolor** and **DefBackColor** to Yellow (H0000FFFF). Set **DefBackColorOdd** to Cyan (H00FFFF00).
6. Save and Run your project. The output should be similar to the following:



Even and Odd rows take on an alternating appearance, making your data easier to read. You still haven't had to write a line of code.

In the next excercise you will learn how to add items to your list at run-time, and sort the list.

Exercise 2: Customizing the user's interface

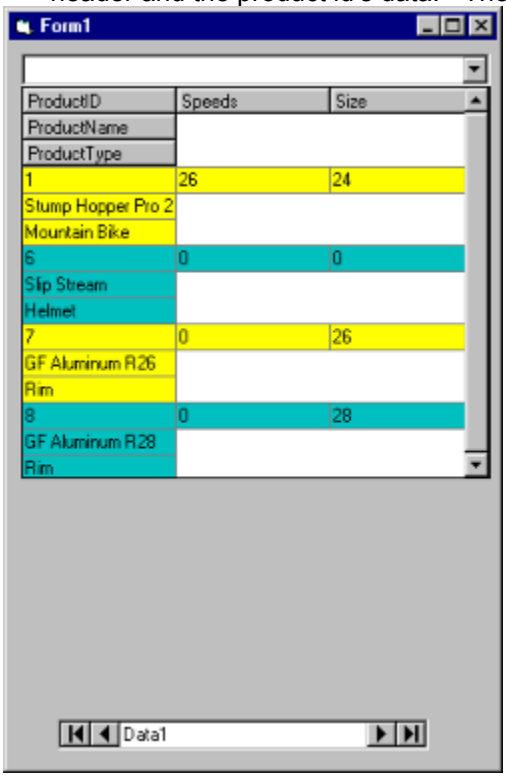
See Also

If you haven't completed exercise one, do so now, it only takes about 2 minutes, and will give you a taste of how powerful *GreenTree's DataList* controls really are. This exercise expands on what you accomplished in exercise one.

1. For this exercise, start a new Visual Basic project and place a **GTCombo** control and a **Data** control on the default Form.
2. Set the **DatabaseName** property of the **Data** control to:

   ```
   C:\DATALIST\TOPROCK.MDB
   ```

   This directory may be different if you have installed the *GreenTree DataList* controls into a different directory other than the default given in the *DataList* setup program.
3. Set the **Data** control's **RecordSource** property to "`Products`". This identifies what table in the database will contain the database records.
4. Click on the **GTCombo** control, and set the **DataSourceList** property to "`Data1`".
5. Set the **SubRows** property to the value `3` in the properties window.
6. Set the **DefBackColor** property to Yellow (H0000FFFF) and the **DefBackColorOdd** property to Cyan (H00C0C000).
7. Save and Run your project. Click on the down-arrow button at the right of the control. The list will be pulled down showing the contents of the `TOPROCK` database.
8. Click on the *ProductName* column header and drag the column header to the space between the *ProductId* column header and the product id's data. As you drag and drop the column an outline of the column will be displayed with the cursor. Drop the *ProductName* column header. Drag and Drop the *ProductType* column header into the remaining space between the *ProductName* column header and the product id's data. The output should look similar to this:



Wow! Now you can see more of your data in the same amount of space. Experiment moving and re-sizing the column headers to give just the appearance you want. Notice that even and odd rows take on an alternating appearance, making your data easier to read. You still haven't had to write a line of code. There are many options related to changing the colors of the column headers, listitems, and subitems allowing you to present your data in virtually any style you choose.

In the next excercise you will learn how to add pictures to the controls in your application.

Exercise 3: Adding Pictures to your application.

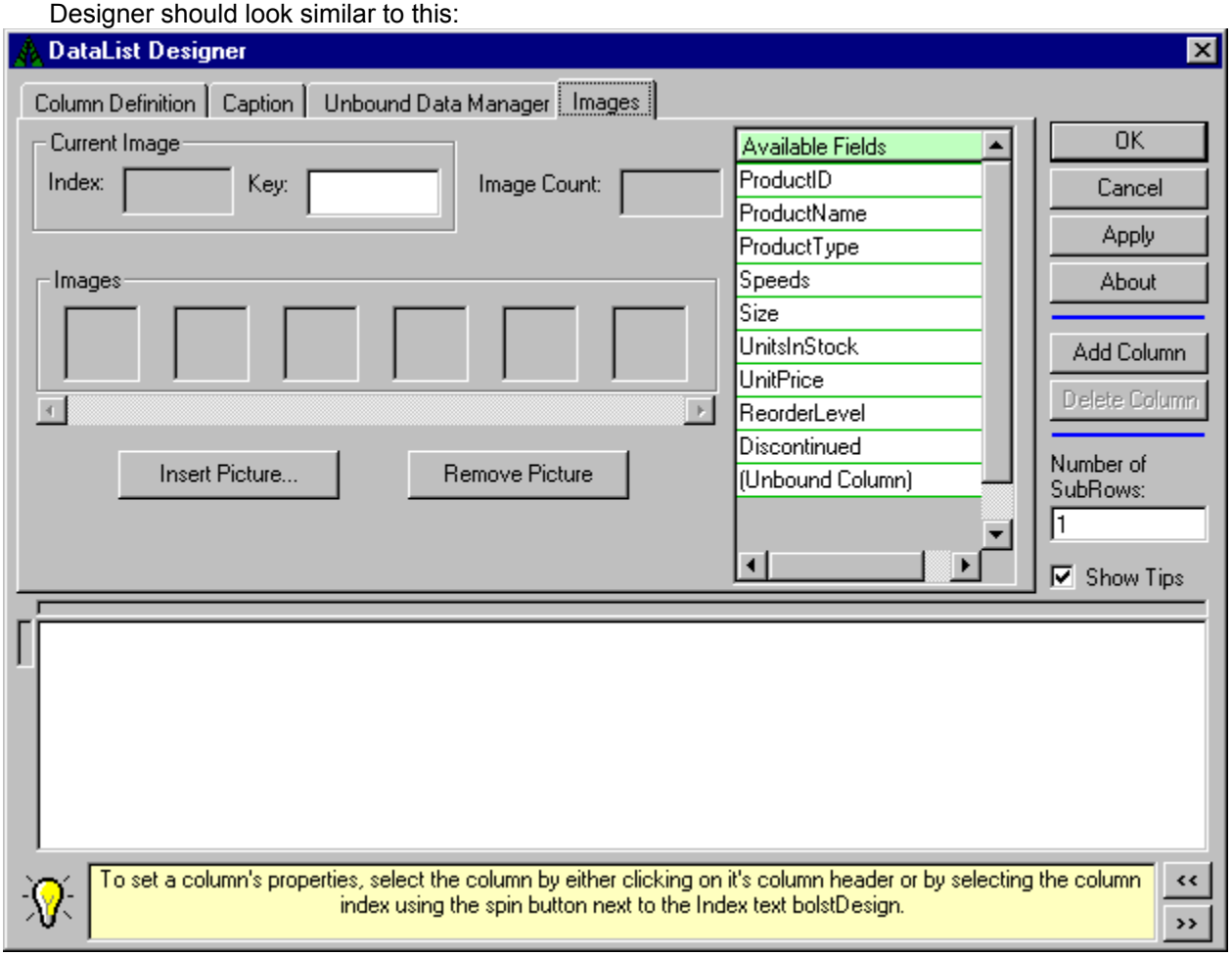
See Also

For this exercise, you will use the DataList designer tool to add some images to the internal image list and write a little code to add those images to the items as they are displayed from the TOPROCK database.

1. For this exercise, start a new Visual Basic project and place a **GTCombo** control and a **Data** control on the default Form.
2. Set the **DatabaseName** property of the **Data** control to:

   C:\DATALIST\TOPROCK.MDB

   This directory may be different if you have installed the *GreenTree DataList* controls into a different directory other than the default given in the *DataList* setup program.
3. Set the **Data** control's RecordSource property to "Products". This identifies what table in the database will contain the database records.
4. Click on the **GTCombo** control, and set the **DataSourceList** property to "Data1".
5. Right click over the GTCombo control, and select the Retrieve Fields from the context sensitive menu. This adds all of the fields to the list designer's collection of column definitions.
6. Right click again over the GTCombo control, and select the List Designer menu option. The list designer tool appears. Select the Image Tab at the top of the List Designer window. The List Designer should look similar to this:



7. Insert five pictures into the internal image list by clicking on the Insert Picture... button, and selecting

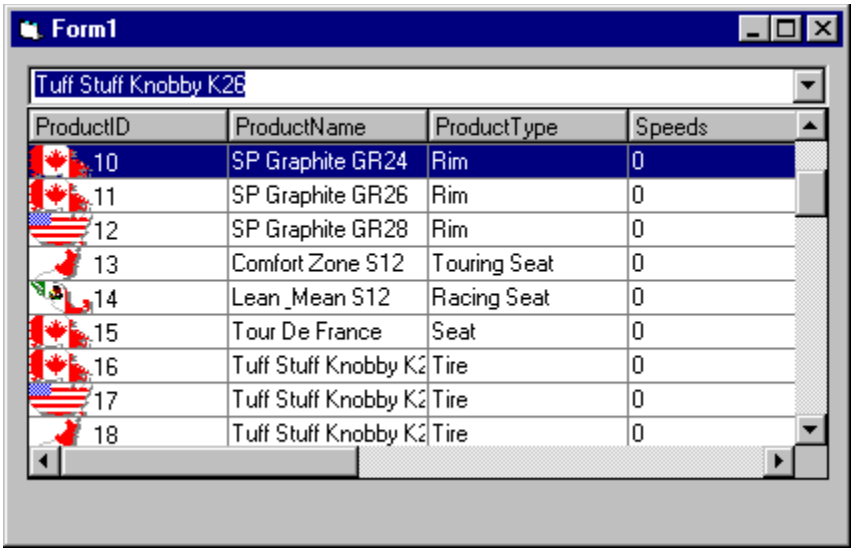
some pictures from the directory Icons\Flags within the Visual Basic directory structure.

8. Select the OK button from the DataList designer window to save your changes and close the DataList designer.
9. Double click on the GTCombo control, and select the **ListItemDataRequest** event from the pull down **Proc** list.
10. Insert the following code into the ListItemDataRequest function body:

```
Dim Idx As Integer

Idx = ListItem.SubItems(0).Text
Idx = Idx Mod 5
If Idx = 0 Then
    Idx = 1
End If
ListItem.SubItems(0).Image = Idx
```

9. Save and Run your project. Click on the down-arrow button at the right of the control. The list will be pulled down showing the contents of the TOPROCK database. The output should look similar to this:



You may want to add more meaningful images to the list of items in your application.

Exercise 3: Adding list items, and sorting.

See Also

Are you ready to write a little code? For this exercise, start a new Visual Basic project and place a **GTList** control on the default Form.

1. Set the GTList's **Sorted** property to **True.**
2. Double Click on the project's Form, to get to the **Form_Load** event handler routine. Add the following lines of code to your **Form_Load** routine:

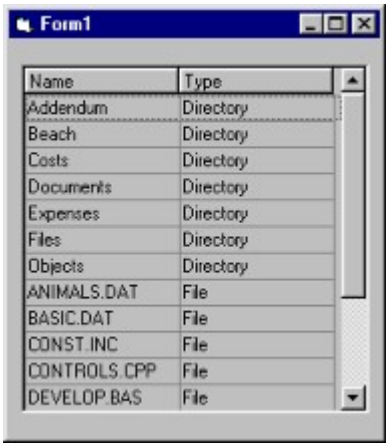
```
Dim MyPath As String
Dim MyName As String

GTList1.ColumnDefs.Add , "Name"
GTList1.ColumnDefs.Add , "Type"

MyPath = "c:\"  ' Set the path.
MyName = Dir(MyPath, vbDirectory)   ' Retrieve the first entry.

Do While MyName <> ""   ' Start the loop.
    ' Ignore the current directory and the encompassing directory.
    If MyName <> "." And MyName <> ".." Then
        ' Use bitwise comparison to determine the file type.
        GTList1.AddItem (MyName)
        If (GetAttr(MyPath & MyName) And vbDirectory) = vbDirectory Then
            GTList1.ListItems(GTList1.NewIndex).SubItems.Add , , "Directory"
        ElseIf (GetAttr(MyPath & MyName) And vbNormal) = vbNormal Then
             GTList1.ListItems(GTList1.NewIndex).SubItems.Add , , "File"
        End If  ' get file type.
    End If
    MyName = Dir    ' Get next entry.
Loop
```

3. Save and Run your project. Notice the list is sorted based on the Name column. Click on the Name column and see that the file names get sorted and displayed in descending order. Click on the Name column again and see that the list is automatically re-sorted in ascending order. Click on the Type column and the list is sorted based on the directory type. The output should look similar to the following, but will differ some based on the file and directory names on your system:



Note that while you drag the vertical scrollbar's thumb, the ScrollTips appear, showing the file name. If you drop the thumb by releasing the left mouse button, the file displayed in the ScrollTip will become the list item in the center row of the window.

Also be aware that automatic sorting of the data only occurs when the control is not bound to a

database, and not operating in Virtual mode.

In the next excercise you will learn how easy it is to create Virtual lists using the GTList control.

Exercise 4: Virtual Lists

See Also

GreenTree's GTList control makes creating a virtual list of items is simple. In this exercise, you will create 10,000 telephone numbers. For this exercise, start a new Visual Basic project and place a **GTList** control on the default Form.

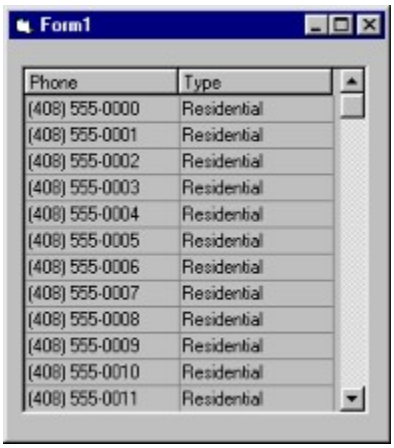
1. Set the **GTList**'s **Virtual** property to **True.** Set the **VirtualItemCount** property to 10000.
2. Double Click on the project's Form, to get to the **Form_Load** event handler routine. Add the following line of code to your **Form_Load** routine:

```
GTList1.ColumnDefs.Add , "Phone"
GTList1.ColumnDefs.Add , "Type"
```

3. Pull down the Object: combo box in Visual Basic's code window, and select the GTList1 object from the list. Pull down the Proc: combo box and select the ListItemDataRequest event procedure from the list: Add the following line of code to the **ListItemDataRequest** event procedure:

```
ListItem.Text = "(408) 555-" + Format(Index, "0000")
If Index < 5000 Then
   ListItem.SubItems.Add , , "Residential"
Else
   ListItem.SubItems.Add , , "Business"
End If
```

4. Save and Run your project. The output should look similar to the following:



Congratulations, you have experienced just some of the powerful features provided by *GreenTree Technology's Active-X Controls* for Microsoft(r) Windows(tm). For more information on specific properties, methods, events, collections, or constants see the GTList Control Reference.

ExtendTipBackColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the BackColor used for all ExtendTip items.

Usage

*object.***ExtendTipBackColor** [= *color*]

The **ExtendTipBackColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the background color of the ExtendTip item, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

See Also

ExtendTipForeColor property, **ScrollTipBackColor** property

Example

The following code sets the **ExtendTipBackColor** to Yellow using the **RGB** function:

```
GTCombo1.ExtendTipBackColor = RGB(255,255,0)
```

ExtendTipBegin Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the extend tip of the control is about to be displayed over the control's ListItem or SubItem.

Usage

Private Sub *object*_**ExtendTipBegin(**Cancel **As Boolean)**

The **ExtendTipBegin** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
Cancel	A reference to a boolean that determines if the extended tip should be displayed..

Remarks

To prevent the extended tip from being displayed, set Cancel to **True**. Leaving Cancel unmodified or setting Cancel to **False** will allow the extend tip to be displayed. If the extend tip is canceled, this event will be fired when the next **ExtendTipDelay** time-out occurs.

See Also

ExtendTipEnd, ScrollTipBegin, ScrollTipEnd events.

Example

This example cancels the extend tip for a **GTCombo** control.

```
Private Sub GTCombo1_ExtendTipBegin (Cancel As Boolean)
      Cancel = True
End Sub
```

ExtendTipDataField Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets a value that binds a **Field** object to the ExtendTip.

Usage

*object.***ExtendTipDataField** [= *value*]

The **ExtendTipDataField** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	A string expression that evaluates to the name of one of the fields in the Recordset object specified by the **DataSource** property.

Remarks

The ExtendTip of a GTList or GTCombo control can be bound to a database field. This property specifies which field in the current data record is used as the text for the ExtendTip item.

See Also

DataSource property, **ScrollTipDataField** property

Example

The example below sets the **ExtendTipDataField** to the `Authors` field. This example assumes that a **DataSource** exists and that the data source has an Author field in its record structure.

```
GTCombo1.ExtendTipDataField = "Authors"
```

ExtendTipDelay Property

Applies To

GTList control**, GTCombo** control

Description

Gets or Sets the amount of time (in milliseconds) that the mouse has to pause over the ExtendTip item before the ExtendTip is displayed.

Usage

*object.***ExtendTipDelay** [= *number*]

The **ExtendTipDelay** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A long integer specifying the number of milliseconds the mouse has to be stationary over the ExtendTip before the ExtendTip text is displayed.

Remarks

The ExtendTip item may not appear if the one or more of the following conditions exists:

- The mouse is not stationary over the ExtendTip object for the **ExtendTipDelay** time.
- The **ExtendTipTimeout** is so short, the ExtendTip is removed before the user sees the ExtendTip.
- The application containing the ExtendTip is not the active application.

See Also

ExtendTipTimeout property, **ExtendTipBegin, ExtendTipEnd** events

Example

The following code sets the **ExtendTipDelay** to half a second for a **GTList** control.

```
GTList1.ExtendTipDelay = 500
```

ExtendTipEnd Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the extend tip is removed from the display.

Usage

Private Sub *object*_**ExtendTipEnd()**

The *object* expression evaluates to one of the objects or controls in the Applies To list.

Remarks

This event is fired when the **ExtendTipTimeout** has expired and the extend tip is removed from the display.

See Also

ExtendTipBegin, ScrollTipBegin, ScrollTipEnd events.

Example

This example shows the syntax of the ExtendTipEnd function.

```
Private Sub GTCombo1_ExtendTipEnd ()

End Sub
```

ExtendTipFont Property

Applies To

GTList control**, GTCombo** control

Description

Returns the font used for the ExtendTip.

Usage

*object.***ExtendTipFont**

Returns a font object used for the ExtendTip objects. The *object* placeholder can be any of the controls listed in the Applies To section.

Remarks

When the ExtendTip's text is displayed this font is used.

See Also

ScrollTipFont property

Example

The following code sets the **ExtendTipFont**'s bold value **True** for a **GTCombo** control:

```
GTCombo1.ExtendTipFont.Bold = True
```

ExtendTipForeColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the forecolor used for the text for ExtendTip.

Usage

*object.***ExtendTipForeColor** [= *color*]

The **ExtendTipForecolor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the forecolor color of the text displayed for all ExtendTip, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the foreground color value for the text displayed for the ExtendTip.

See Also

ExtendTipBackColor property, **ScrollTipForeColor** property

Example

The following code sets the control's **ExtendTipForeColor** to the first ListItem's **ForeColor** for a **GTList** control:

```
GTList1.ExtendTipForeColor = GTList1.ListItems(0).ForeColor
```

ExtendTipTimeout Property

Applies To

GTList control**, GTCombo** control

Description

Gets or Sets the amount of time (in milliseconds) that ExtendTip is displayed before it disappears if the user does not move the mouse.

Usage

*object.***ExtendTipTimeout** [= *number*]

The **ExtendTipTimeout** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A long integer specifying the number of milliseconds the ExtendTip text is to be displayed before disappearing.

Remarks

If the value of this property is set to 0, the ExtendTip remains displayed until the user moves the mouse.

See Also

ExtendTipDelay property, **ExtendTipBegin, ExtendTipEnd** events

Example

The following code sets the **ExtendTipTimeout** to five seconds for a **GTList** control.

```
GTList1.ExtendTipTimeout = 5000
```

ExtendTips Property

Applies To

GTList control**, GTCombo** control

Description

Enables or disables ExtendTips for the control's list items.

Usage

object.**ExtendTips** [= *boolean*]
The **ExtendTips** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the ExtendTips are enabled.

Remarks

Setting this value to **True** enables the ExtendTips on the control's list items.

See Also

ExtendTips property

Example

The following code turns the control's ExtendTips off by setting the **ExtendTips** property to **False**.

```
GTList1.ExtendTips = False
```

Font3D Property (ColumnDef, ListItem, SubItem object)

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies the 3D font style used to draw the object.

Usage

object.**Font3D**[= *number*]

The **Font3D** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
number	An integer specifying the 3D style to use when drawing text for the specified object.

Settings

The settings for *number* are:

Setting	Value	Description
gtFont3DNone	0	(Default) None. Text is displayed normally.
gtFont3DRaisedLight	1	Raised Light. The text is slightly raised.
gtFont3DRaisedHeavy	2	Raised Heavy. The text is raised.
gtFont3DInsetLight	3	Inset Light. The text is slightly sunken.
gtFont3DInsetHeavy	4	Inset Heavy. The text is sunken.
gtFont3DUseDefault	5	Use Default. Use the value stored in the parent control's **DefFont3D** property.

Remarks

When an object is drawn the value of this property is applied to the item's text. If the value of this property is set to **gtFont3DUseDefault**, the value in **DefFont3D** is used.

See Also

Font property, **GTList** control, **GTCombo** control, **DefFont3D** property

Example

The following code sets the **Font3D** to Raised Heavy for a **GTList** control:

```
GTList1.ColumnDefs.Item(0).Font3D = 2
```

ForeColor Property (ColumnDef, ListItem, SubItem object)

Applies To

ColumnDef object**, ListItem** object**, SubItem** object

Description

Specifies the forecolor used when drawing text for the object specified.

Usage

object. **ForeColor** [= *color*]

The **Forecolor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
color	A value or constant that determines the forecolor color of the text displayed for the object, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the foreground color value for the text displayed for each object.

See Also

BackColor property

Example

The following code sets the **ForeColor** property for a **ListItem** object to the default foreground color of the parent control:

```
GTList1.ListItems.Item(0).ForeColor = GList1.DefForeColor
```

Format Property

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies the appearance of numbers, dates, times, and text when they are displayed.

Usage

object.**Format** [= *posformat$; negformat$; zeroformat$; nullformat$*]

The **Format** property has these parts:

Part	Description
object	An object expression that evaluates to a one of the controls in the **Applies To** list.
posformat$	An Expression used to display positive values.
negformat$	An Expression used to display negative values.
zeroformat$	An Expression used to display zero values.
nullformat$	An Expression used to display **Null** or empty values.

Remarks

The **Format** property defines the format expressions used to display the contents of the control. You can use the same format expressions as defined by the Visual Basic Format$ function, with the exception that named formats ("On/Off") can't be used.

This property can have from one to four parameters separated by semicolons. If one of the parameters is not specified, the format specified by the first parameter is used. If multiple parameters appear, the appropriate number of separators must be used. For example, to specify *posformat*$ and *nullformat*$, use the syntax
[form.]GTList1.ColumnDefs(0).Format = *posformat*$;;; *nullformat*$

The following table shows a number of standard formats available to the user; however, any valid *Format*$ expression may be defined.

Data type	Value	Description
Number	(Default) Empty string	General Numeric format. Displays as entered.
Number	$#,##0.00;($#,##0.00)	Currency format. Uses thousands separator; displays negative numbers enclosed in parentheses.
Number	0	Fixed number format. Displays at least one digit.
Number	#,##0	Commas format. Uses commas as thousands separator.
Number	0%	Percent format. Multiplies value by 100 and appends a percent sign.
Number	0.00E+00	Scientific format. Uses standard scientific notation.
Date/Time	(Default) c	General Date and Time format. Displays date, time, or both.
Date/Time	dddddd	Long Date format. Same as the Long Date setting in the International section of the Microsoft Windows Control Panel. Example: Tuesday, May 26, 1992.
Date/Time	dd-mmm-yy	Medium Date format. Example: 26-May-92.
Date/Time	ddddd	Short Date format. Same as the Short Date setting in the International section of the Microsoft Windows Control Panel. Example: 5/26/92.
Date/Time	ttttt	Long Time format. Same as the Time setting in the International section of the Microsoft Windows Control Panel. Example: 05:36:17 A.M.
Date/Time	hh:mm AM/PM	Medium Time format. Example: 05:36 A.M.
Date/Time	hh:mm	Short Time format. Example: 05:36.

See Also

Text, Caption properties

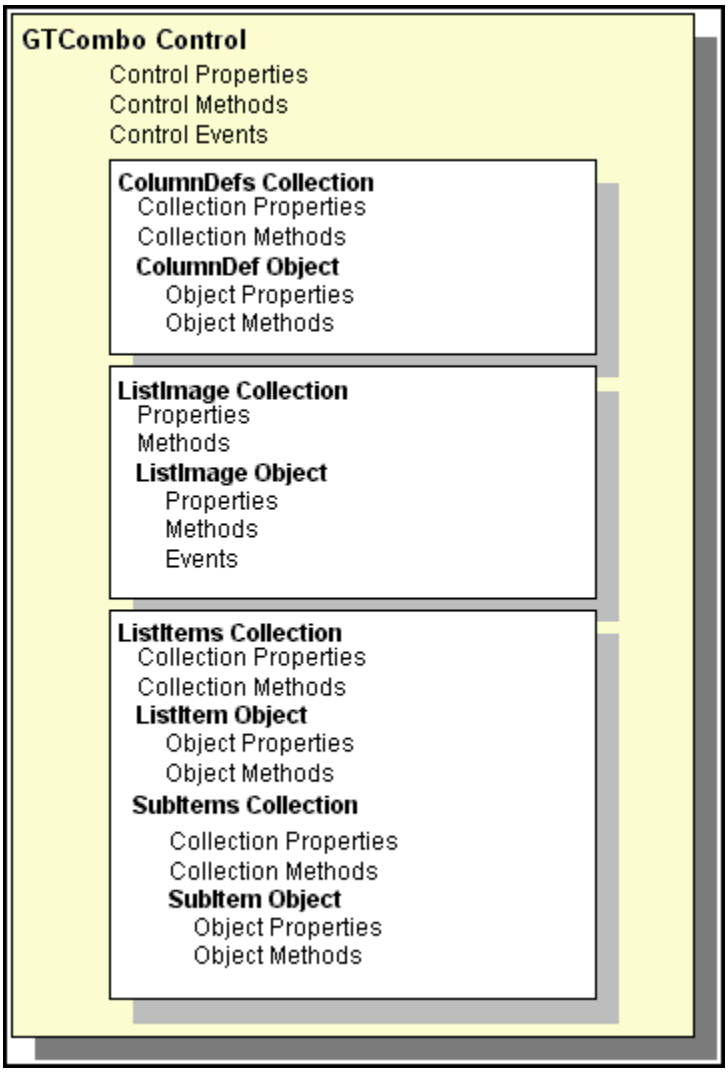
Example

The following code sets the **Format** to a Date/Time format for a **GTList** control's column header object:

```
GTList1.ColumnHeaders(0).Format = "dd-mmm-yyy"
```

GTCombo Control Hierarchy

The following diagram depicts the hierarchy for the GTCombo control:



GTCombo Control Quick Tour

See Also

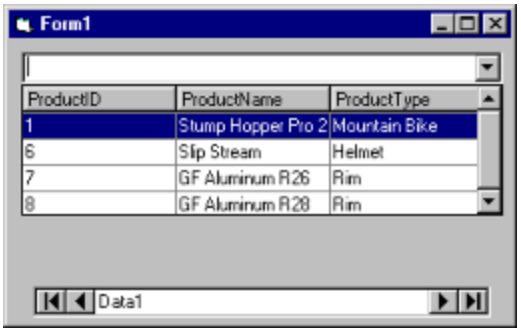
Exercise 1: Binding Data to the Control

This exercise binds the **GTCombo** control to the TOPROCK.MDB database. TopRock is a fictitious bicycle manufacturing company, and TOPROCK.MDB is the database of their product inventory.

1. For this exercise, start a new Visual Basic project and place a **GTCombo** control and a **Data** control on the default Form.
2. Set the **DatabaseName** property of the **Data** control to:

   `C:\DATALIST\TOPROCK.MDB`

   This directory may be different if you have installed the *GreenTree DataList* controls into a different directory other than the default given in the *DataList* setup program.
3. Set the **Data** control's RecordSource property to "`Products`". This identifies what table in the database will contain the database records.
4. Click on the **GTCombo** control, and set the **DataSourceList** property to "`Data1`".
5. Save and Run your project. Click on the down-arrow button at the right of the control. The list will be pulled down showing the contents of the `TOPROCK` database. The output should look similar to this:



The **GTCombo** control has gathered the records in the database, created column headers, list items, and SubItems that it displays in the control. It has created this view of your data with no coding!

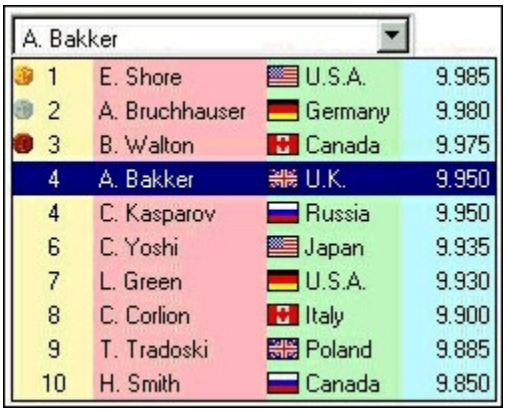
6. As you select an item in the list, the drop down list is rolled up, and the *ProductID* field is displayed in the edit box portion of the control. As a user of the control, you probably do not want to edit the *ProductId.* Let's change the default activity of the control, so you can edit the *UnitPrice.*
7. Stop running the program by closing the window or pressing the Visual Basic Stop button. Click on the **GTCombo** control to select it.
8. Change the **DataFieldList** property to UnitPrice by pulling down the list of fields in the properties window, and selecting **UnitPrice** from the list.
9. Save and Run your project. Click on the down-arrow button at the right of the control. The list will be pulled down showing the contents of the TOPROCK database. Select the *Slip Stream* helmet from the list of items. The *Slip Stream's* unit price is displayed in the edit box portion of the control.
10. Change the unit price from 59.99 to 49.99. Click on the **Data** control to change the row, and the database is updated with the new *UnitPrice* data.

Now that you have experienced how easy it is to view and edit your data using **GTCombo,** in the next exercise you will see how easy it is to customize and use **GTCombo**'s properties to provide the user with a more compelling interface.

GTCombo Control Reference

See Also Properties Methods Events Collections Constants

The GTCombo Control is a greatly enhanced version of the standard ComboBox control. The GTCombo control combines the features of a TextBox control and a **GTList** control. Users can enter information in the TextBox portion of the control, or they can select information from either the list or the drop down list portion of the control. Unlike the **GTList** control, only a single item in the **GTCombo** list can be selected at any one time. Virtual lists of items can be displayed in the control, by using the **Virtual**, **VirtualItemCount** properties and the **ListItemDataRequest** event to add items to the list when needed by the control.



Usage

GTCombo

Remarks

To add or delete items in a GTCombo control, use the **AddItem** or **RemoveItem** methods. The control has some special collections associated with is that provide expanded capabilities over a standard ComboBox control:

- Use the **ColumnDefs Add** method to add column header definitions
- Use the **ListItems** collection to customize ListItem objects.
- The **SubItems** collection adds associative text or graphics to each list item.

Additional Topics

The **GTCombo** control is defined by its properties, methods, events, and collections. See the Control Hierarchy.

The control's properties, manage all the global aspects of the control's appearance, its' default behavior, and which features are enabled or disabled. Here are just a few of the properties that are meaningful to developing compelling applications using GTCombo:

- DataSource and DataSourceList
- Virtual and VirtualItemCount
- SubRows and SubRowsStatic

The List Designer tool can help you design your combo control at design time.

GTCombo Events

Underlined items are stock events

Click
ClickColumn
ClickItem
ColumnAfterMove
ColumnBeforeMove
ColumnResize
DblClick
DragDrop
DragOver
DropDown
ExtendTipBegin
ExtendTipEnd
GotFocus
KeyDown
KeyPress
KeyUp
ListItemDataRequest
LostFocus
MouseDown
MouseMove
MouseUp
PositionList
ScrollTipBegin
ScrollTipEnd

GTCombo Properties

Underlined items are stock properties

AllowColumnDragDrop
AllowColumnResize
AllowColumnSortClick
Appearance
AutoPositionList
BackColor
BorderStyle
CalcRowCountOnLoad
ColumnCaptions
ColumnDefs
ColumnHeaderHeight
ColumnResizeElastic
ComboStyle
Container
DataChanged
DataField
DataFieldDisplay
DataFieldList
DataSource
DataSourceList
DefBackColor
DefBackColorOdd
DefBackStyle
DefColCaptionBackColor
DefColCaptionBackStyle
DefColCaptionBorderStyle
DefColCaptionFont
DefColCaptionFont3D
DefColCaptionForeColor
DefColCaptionPictureAlignment
DefColCaptionTextAlignment
DefColCaptionWordWrap
DefColumnWidth
DefFont
DefFont3D
DefForeColor
DefForeColorOdd
DefPictureAlignment
DefRowHeight
DefTextAlignment
DefWordWrap
DisplayText
DragIcon
DragMode
DropDownWidth
DynamicCols
Enabled
ExtendTipBackColor
ExtendTipDataField
ExtendTipDelay
ExtendTipFont
ExtendTipForeColor

ExtendTips
ExtendTipTimeout
ForeColor
GridLineColor
GridLineStyle
GridLineType
Height
HelpContextID
HideSelection
HorzScrollBar
hWnd
ImageList
Index
ItemData
Left
LeftColumn
ListCount
ListImages
ListIndex
ListItems
MaskColor
MaskColorEnabled
MaxDropDownItems
MinDropDownItems
MouseIcon
MousePointer
Name
NewIndex
Object
Parent
Picture
PictureAlignment
ScrollTipBackColor
ScrollTipDataField
ScrollTipFont
ScrollTipForeColor
ScrollTips
SelCount
Selected
SelectedItem
SelLength
SelStart
SelText
Sorted
SortKey
SortKey2
SortKey3
SortOrder
SortOrder2
SortOrder3
SubRows
SubRowsStatic
TabIndex
TabStop
Tag
TagVariant

Text
Top
TopIndex
VertScrollBar
Virtual
VirtualItemCount
Visible
WhatsThisHelpID
Width
ZOrder

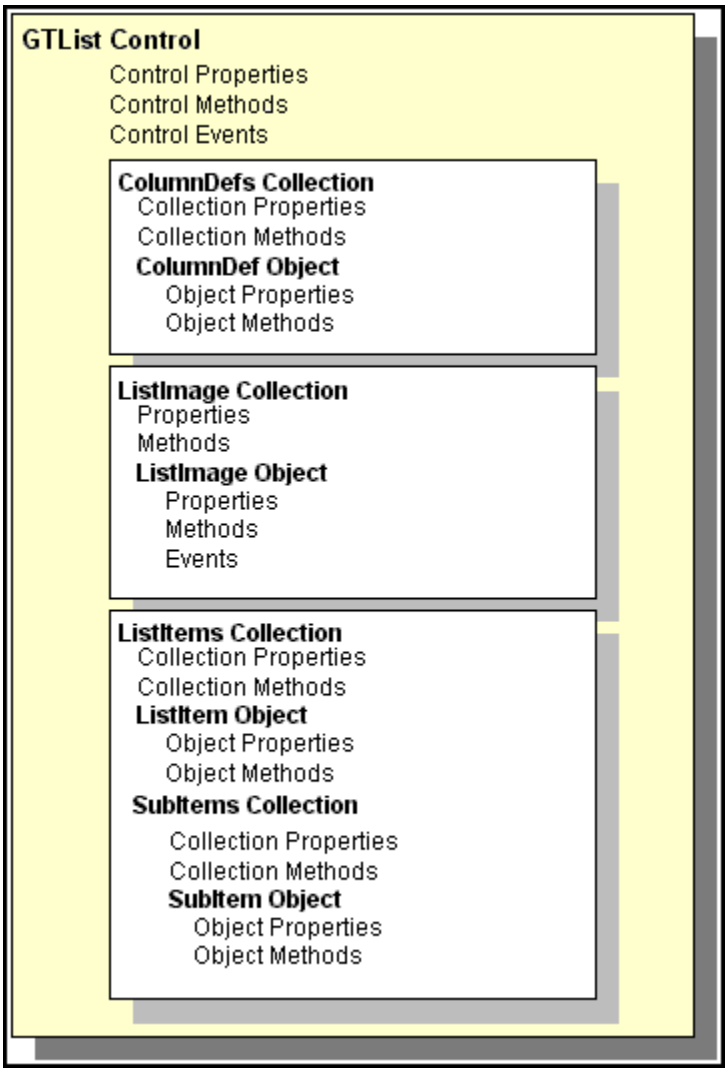
GTCombo Reference:See Also

GTCombo Control Overview
GTCombo Control Quick Tour

GTCombo and GTList Collections

GTList Control Hierarchy

The following diagram depicts the hierarchy for the GTList control:



GTList Control Quick Tour

See Also

This quick tour guides you through the creation of some sample programs using the *GreenTree* **GTList** control. The first exercise creates a simple list that is bound to an external database.

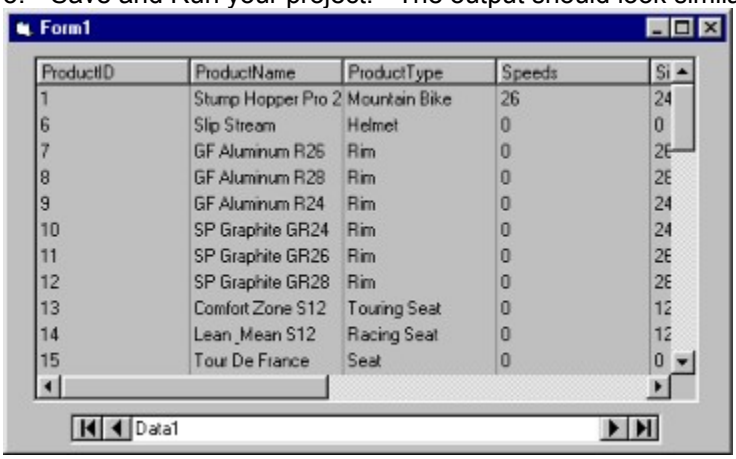
Exercise 1: Binding Data to the Control

This exercise binds the **GTList** control to the TOPROCK.MDB database. TopRock is a fictitious bicycle manufacturing company, and TOPROCK.MDB is the database of their product inventory.

1. For this exercise, start a new Visual Basic project and place a **GTList** control and a **Data** control on the default Form.
2. Set the **DatabaseName** property of the **Data** control to:

   `C:\DATALIST\TOPROCK.MDB`

   This directory may be different if you have installed the *GreenTree DataList* controls into a different directory other than the default given in the *DataList* setup program.
3. Set the **Data** control's **RecordSource** property to "*Products*". This identifies what table in the database will contain the database records.
4. Click on the **GTList** control, and set the **DataSource** property to "*Data1*".
5. Save and Run your project. The output should look similar to this:



The **GTList** control has gathered the records in the database, created column headers, list items, and **SubItems** that it displays in the control. It has created this view of your data with no coding!

6. Move the mouse over the column headers between "*ProductId*" and "*ProductName*". The cursor will change to reflect that re-sizing of the columns is permitted. Click on the column bar and reduce the width of the "ProductId" column. More data can now be seen in the window. Resize other columns to get a complete picture of the data.
7. Click on the control's vertical scroll bar buttons, and see how the control scrolls through each item in the list as you view the TOPROCK product database. Click on the control's horizontal scroll bar buttons to see additional information for each record.
8. Reorder the appearance of the columns by dragging and dropping. Click on a column header, and while holding the left mouse button down, move the column to another position. The order of the columns is changed as you drag and drop each column.

Now that you have experienced how easy it is to view your data using **GTList,** in the next exercise you will see how easy it is to customize and use **GTList**'s properties to enhance the user's experience.

GTList Control Quick Tour See Also

GTList Control Overview
GTList Control Reference

ColumnDefs collection, ColumnDef object
ListImages collection, ListImage object
ListItems collection, ListItem object
SubItems collection, SubItem object

Excercise 2: Customizing the look and feel
Excercise 3: Adding list items and sorting
Excercise 4: Virtual Lists

GTList Control Reference

See Also Properties Methods Events Collections Constants

The **GTList** Control is a greatly enhanced version of the standard ListBox control. The **GTList** control displays a list of items from which the user can select one or more. If the number of items in the list exceeds the number that can be displayed in the control's window, a vertical scroll bar can be automatically added to the control. Since the control can have **SubItems** associated with each list item, if the width of the **SubItems** exceeds the width of the control, a horizontal scroll bar can be automatically added to the bottom of the control's window. **Virtual** lists of items can be displayed in the control, by using the **Virtual**, **VirtualItemCount** properties and the **ListItemDataRequest** event to add items to the list when needed by the control.

Rank	Name	Country	Beam	Bars	Floor	Vault	Total
1	E. Shore	U.S.A.	9.985	9.885	9.930	9.945	39.745
2	A. Bruchhauser	Germany	9.980	9.685	9.795	9.985	39.445
3	C. Kasparov	Russia	9.950	9.990	9.855	9.640	39.435
4	A. Bakker	U.K.	9.950	9.715	9.925	9.825	39.415
5	B. Walton	Canada	9.975	9.955	9.675	9.750	39.355

Usage

GTList

Remarks

To add or delete items in a **GTList** control, use the **AddItem** or **RemoveItem** methods. The control has some special collections associated with is that provide expanded capabilities over a standard ListBox control:

- Use the **ColumnDefs Add** method to add column header definitions
- Use the **ListItems** collection to customize ListItem objects.
- The **SubItems** collection adds associative text or graphics to each list item.

Additional Topics

The **GTList** control is defined by its properties, methods, events, and collections. See the Control Hierarchy.

The control's properties, manage all the global aspects of the control's appearance, its' default behavior, and which features are enabled or disabled. Here are just a few of the properties that are meaningful to developing compelling applications using GTCombo:

- DataSource and DataSourceList
- Virtual and VirtualItemCount
- SubRows and SubRowsStatic

The List Designer tool can help you design your combo control at design time.

GTList Control Reference: See Also

GTList Control Overview
GTList Control Quick Tour

ColumnDefs collection, ColumnDef object
ListImages collection, ListImage object
ListItems collection, ListItem object
SubItems collection, SubItem object

GTList Events

Underlined items are stock events

Click
ClickColumn
ClickItem
ColumnAfterMove
ColumnBeforeMove
ColumnResize
DblClick
DragDrop
DragOver
ExtendTipBegin
ExtendTipEnd
GotFocus
KeyDown
KeyPress
KeyUp
ListItemDataRequest
LostFocus
MouseDown
MouseMove
MouseUp
PositionList
ScrollTipBegin
ScrollTipEnd

GTList Properties

Underlined items are stock properties

AllowColumnDragDrop
AllowColumnResize
Appearance
AllowColumnSortClick
AutoPositionList
BackColor
BorderStyle
ColumnCaptions
ColumnHeaderHeight
ColumnResizeElastic
Container
DataSource
DefBackColor
DefBackColorOdd
DefBackStyle
DefColCaptionBackColor
DefColCaptionBackStyle
DefColCaptionBorderStyle
DefColCaptionFont
DefColCaptionFont3D
DefColCaptionForeColor
DefColCaptionPictureAlignment
DefColCaptionTextAlignment
DefColCaptionWordWrap
DefColumnWidth
DefFont
DefFont3D
DefForeColor
DefForeColorOdd
DefPictureAlignment
DefRowHeight
DefTextAlignment
DefWordWrap
DragIcon
DragMode
DynamicCols
Enabled
ExtendTipBackColor
ExtendTipDataField
ExtendTipDelay
ExtendTipFont
ExtendTipForeColor
ExtendTips
ExtendTipTimeout
FitColumnsToList
GridLineColor
GridLineStyle
GridLineType
Height
HelpContextID
HideSelection
HorzScrollBar

hWnd
ImageList
Index
ItemData
Left
LeftColumn
ListCount
ListIndex
MaskColor
MaskColorEnabled
MouseIcon
MousePointer
MultiSelect
Name
NewIndex
Object
Parent
Picture
PictureAlignment
ScrollTipBackColor
ScrollTipDataField
ScrollTipFont
ScrollTipForeColor
ScrollTips
SelCount
Selected
SelectedItem
Sorted
SortKey
SortKey2
SortKey3
SortOrder
SortOrder2
SortOrder3
SubRows
SubRowsStatic
TabIndex
TabStop
Tag
TagVariant
Text
Top
TopIndex
CalcRowCountOnLoad
VertScrollBar
Virtual
VirtualItemCount
Visible
WhatsThisHelpID
Width

GTList and GTCombo Constants

Constants_Appearance

gtFlat	0
gt3D	1

Constants_BackStyle

gtBackStyleTransparent	0
gtBackStyleOpaque	1
gtBackStyleUseDefault	2

Constants_BorderStyle

gtBorderStyleNone	0
gtBorderStyleSingle	1
gtBorderStyleInset	2
gtBorderStyleRaised	3
gtBorderStyleUseDefault	4

Constants_BorderStyleStock

gtBorderStyleStockNone	0
gtBorderStyleStockSingle	1

Constants_ComboStyle (applies to GTCombo only)

gtStyleCombo	0
gtStyleSimple	1
gtStyleList	2

Constants_Font3D

gtFont3DNone	0
gtFont3DRaisedLight	1
gtFont3DRaisedHeavy	2
gtFont3DInsetLight	3
gtFont3DInsetHeavy	4
gtFont3DUseDefault	5

Constants_DynamicCols

gtNever	0
gtInitial	1
gtAlways	2

Constants_GridLineStyle

gtGridLineStyleSolid	0
gtGridLineStyleDotted	1
gtGridLineStyleRaised	2
gtGridLineStyleInset	3

Constants_GridLineType

gtGridLineTypeNone	0
gtGridLineTypeHoriz	1
gtGridLineTypeVert	2
gtGridLineTypeBoth	3

Constants_ColumnCaptions

gtColumnCaptionsFalse	0
gtColumnCaptionsTrue	1
gtColumnCaptionsUseDefault	2

Constants_Lookup

gtLookup_Index	0
gtLookup_Key	1

Constants_LookupType

gtTextToText	0
gtNumToText	1
gtTextToImage	2
gtNumToImage	3

Constants_MultiSelect (applies to GTList only)

gtMultiSelectNone	0
gtMultiSelectSimple	1
gtMultiSelectExtended	2

Constants_PictureAlignment

gtPictureAlignmentLeftTop	0
gtPictureAlignmentLeftMiddle	1
gtPictureAlignmentLeftBottom	2
gtPictureAlignmentRightTop	3
gtPictureAlignmentRightMiddle	4
gtPictureAlignmentRightBottom	5
gtPictureAlignmentCenterTop	6
gtPictureAlignmentCenterMiddle	7
gtPictureAlignmentCenterBottom	8
gtPictureAlignmentStretch	9
gtPictureAlignmentTile	10
gtPictureAlignmentLeftOfText	11
gtPictureAlignmentRightOfText	12
gtPictureAlignmentAboveText	13
gtPictureAlignmentBelowText	14
gtPictureAlignmentUseDefault	15

Constants_SelectionType

gtSelectionTypeBar	0
gtSelectionTypeItemOnly	1

Constants_Sorting

gtSortAscending	0
gtSortDescending	1

Constants_TextAlignment

gtTextAlignmentLeftTop	0
gtTextAlignmentLeftMiddle	1
gtTextAlignmentLeftBottom	2
gtTextAlignmentRightTop	3
gtTextAlignmentRightMiddle	4
gtTextAlignmentRightBottom	5

gtTextAlignmentCenterTop	6
gtTextAlignmentCenterMiddle	7
gtTextAlignmentCenterBottom	8
gtTextAlignmentUseDefault	9

Constants_WordWrap

gtWordWrapOff	0
gtWordWrapOn	1
gtWordWrapUseDefault	2

GTList and GTCombo Methods

Underlined items are stock methods

AddItem
Clear
Drag
HitTest
RemoveItem
SetFocus
Refresh
ShowWhatsThis
SetFocus
ZOrder

GridLineColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the color for the control's grid lines.

Usage

object.**GridLineColor** [= *color*]

The **GridLineColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the color of the grid lines, as described in Settings.

Settings

The settings for *color* are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

If the **GridLineStyle** is vertical, horizontal, or both, grid lines are drawn around the list items in the control. This property determines the color for the grid lines.

See Also

GridLineStyle, GridLineType properties.

Example

The following code sets the **GridLineColor** for a **GTList** control to Black:

```
GTList1.GridLineColor = RGB(0, 0, 0)
```

GridLineStyle Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the style of grid lines to display around the control's list items.

Usage

*object.***GridLineStyle** [= *number*]

The **GridLineStyle** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the style of grid lines to display around the control's list items as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtGridLineStyleSolid	0	Solid. A solid line is drawn as the grid line
gtGridLineStyleDotted	1	Dotted. A dotted line is drawn as the grid line.
gtGridLineStyleRaised	2	Raised. The grid lines have a raised appearance.
gtGridLineStyleInset	3	Inset. The grid lines have a sunken appearance.

Remarks

Use this property to control the style of the grid lines drawn around the control's list items. Use the **GridLineType** property to disable the grid lines.

See Also

GridLineColor, GridLineType properties

Example

The following code sets the grid line style to **gtGridLineStyleSolid** for a **GTList** control:

```
GTList1.GridLineStyle = gtGridLineStyleSolid
```

GridLineType Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the line type for the grid lines displayed around the control's list items.

Usage

object.**GridLineType** [= *number*]

The **GridLineType** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the line type for the grid lines displayed around the control's list items as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtGridLineTypeNone	0	None. No grid lines are drawn.
gtGridLineTypeVertical	1	Vertical. Vertical grid lines are displayed between the columns.
gtGridLineTypeHorizontal	2	Horizontal. Horizontal grid lines are displayed between each row.
gtGridLineTypeBoth	3	Both. Both horizontal and vertical lines are displayed between the rows and columns respectively.

Remarks

Use this property to control the type of grid lines drawn around the control's list items.

See Also

GridLineColor, GridLineStyle properties

Example

The following code sets the grid line type to **gtGridLineTypeBoth** for a **GTList** control:

```
GTList1.GridLineType = gtGridLineTypeBoth
```

ColumnCaptions Property

Applies To **GTList** control, **GTCombo** control

Description

Returns or sets whether column headers are hidden.

Usage

object.**ColumnCaptions** [= *number*]

The **ColumnCaptions** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer or constant specifying if the column header captions are to be hidden or displayed..

Remarks

Setting this property to **gtColumnCaptionsFalse** hides the column header bar. The default value for this property is **gtColumnCaptionsTrue**. Set this property to gtColumnCaptionsUseDefault to use the value set in the DefColumnCaptions property.

See Also

ColumnDefs collection, **GTList** control., **DefColumnCaptions** property

Example

The following code sets the **ColumnCaptions** property to the current value of a check box control:

```
GTList1.ColumnCaptions = Check1.Value
```

HideSelection Property

Applies To

GTCombo control

Description

Returns or sets a value that specifies if the selected list item remains highlighted when the control loses focus.

Usage

object.**HideSelection** [= *boolean*]

The **HideSelection** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying how list items are displayed when the control loses the focus, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) The items in the control are no longer selected when the control loses the focus.
False	The items are still selected after the control loses focus.

Remarks

Normally, the selected items in the control are hidden when the control loses focus. This is the default action of the property.

If you want the selected items to remain selected after the control loses focus, set the **HideSelection** property to **False**.

See Also

MultiSelect property, **Selected** property

Example

The following code sets the **HideSelection** property to the current value of a check box control:

```
GTCombo1.HideSelection = Check1.Value
```

HitTest Method

Applies To

GTList control**, GTCombo** control

Description

Returns a *short* indicating what object is located at the coordinates of **x** and **y**. Most often used with drag-and-drop operations to determine if a drop target item is available at the present location.

Usage

object.**HitTest(***x* **As Single,** *y* **As Single, [***ListItem* **As Variant], [***SubItem* **as Variant]) As Short**

The **HitTest** method has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
x, y	Coordinates of a target object, which is a **ListItem** object.
ListItem	Optional. Object returned if the return value equals gtHitTestListItem, otherwise NULL.
SubItem	Optional. Object returned if the return value equals gtHitTestListItem; otherwise NULL.

Remarks

If no object exists at the specified coordinates, the **HitTest** method returns gtHitTestError.

The **HitTest** method is most frequently used with the **MousePointer** and **MouseIcon** properties to highlight an object as the mouse is dragged over it.

Settings

The Settings for the *short* return value are:

Setting	Value	Description
gtHitTestError	0	HitTest failed or hit empty space on the control.
gtHitTestListItem	1	HitTest hit a ListItem or SubItem object.
gtHitTestCaption	2	HitTest hit the column headers bar above the list items.
gtHitTestVScroll	3	HitTest hit the vertical scroll bar.
gtHitTestHScroll	4	HitTest hit the horizontal scroll bar.

See Also

ListItems collection**, ListItem** object**, MousePointer** property**, MouseIcon** property.

Example

This example changes the background color of the **ListItem** object to Red when the user moves the mouse over the ListItem object for a **GTList** control.

```
Private Sub GTCombo1_MouseMove(button As Integer,
                               shift As Integer,
                                x As Single, y As Single)
      Dim xItem As ListItem
      Dim nRet As Short

      nRet = GTCombo1.HitTest(x, y, xItem)
      If nRet = gtHitTestListItem Then
            xItem.BackColor = RGB( 255, 0, 0)
      End If
End Sub
```

HorzScroll Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the list portion of the control is scrolled in the horizontal direction.

Usage

Private Sub *object*_**HorzScroll()**

The *object* expression evaluates to one of the controls in the Applies To list.

Remarks

This event is fired each time thelist is scrolled in the horizontal direction. You may want to check the TopIndex property to determine which ListItem is at the top of the list inside this event handler.

See Also

HorzScrollBar, **TopIndex** propterties, **VertScroll** event

Example

This example displays the first visible list item's text when the list is scrolled for a **GTList** control:

```
Private Sub GTList1_HorzScroll ()
      Label1.Caption = GTList1.ListItems(GTList1.TopIndex).Text
End Sub
```

HorzScrollBar Property

Applies To

GTList control **GTCombo** control

Description

Gets or Sets the horizontal scrolling mode and the appearance of the scroll bars for the control's window.

Usage

object.**HorzScrollBar** [= *mode*]

The **HorzScrollBar** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
mode	An integer specifying the horizontal scroll bar setting for the control.

Settings

The settings for *number* are:

Setting	Value	Description
gtSBNever	0	Never. No horizontal scroll bar is displayed in the control's window.
gtSBAlways	1	Always. A horizontal scroll bar is always present on the bottom of the control's window.
gtSBNeeded	2	(Default) Needed. A horizontal scroll bar will be added to bottom of the control's window automatically when the width of the ListItems and its associated SubItems exceeds the width that can be displayed in the window or the width of the ColumnDefs exceeds the width of the control's window.

Remarks

This property controls the appearance of the control's horizontal scroll bars, and the attributes of the scroll bar's thumb. If the ColumnDefs window is hidden the width of the ListItems and its SubItems is used to calculate the width. If the ColumnDefs window is present, the total width of the ColumnDefs is used to calculate the width. Remember that the width of a column may be less than the text contained in the column.

See Also

VertScrollBar property**, ColumnDefs** collection

Example

The following code sets the **HorzScrollBar** to a value set by a user's ListBox control:

```
GTList1.HorzScrollBar = ListBox1.ListIndex
```

Image Property (ColumnDef, ListItem, SubItem object)

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies the image index used for the specified object.

Usage

object.**Image** [= *index*]

The **Image** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
index	A Long integer index into the control's associated **ImageList**.

Remarks

Use this property to get or set the image for the specified object. The index must already exist in the control's image list prior to assigning it to this property.

See Also

ListImages collection, **ListImage** object

Example

The following code sets a ListItem's **Image** to first image for a **GTList** control:

```
GTList1.ListItems.Item(0). Image = 1
```

ImageList Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets the **ImageList** control, if any, that is associated with the **GTList** or **GTCombo** control.

Usage

object.**ImageList** [= *imagelist*]

The **ImageList** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
imagelist	A string expression that is the name reference specifying an **ImageList** control to associate with the control.

Remarks

For the control to use the **ImageList** property, you must put an **ImageList** control on the form. Then, at design time, you can set the **ImageList** property in the associated control's properties dialog box from the combo box containing the names of all the **ImageList** controls currently on the form. To associate an **ImageList** with the control at run time, set the control's **ImageList** property to the **ImageList** control you want to use.

See Also

ListImages collection, **ListImage** object

Example

The following code associates an **ImageList** control (ImageList1) to a **GTList** control:

```
GTList1.ImageList = ImageList1
```

Included Files

The Setup program will place OCX files in the directories you have specified. Sample applications from the manual are located in the \SAMPLES subdirectory of the DataList ActiveX home directory.

The following table gives a brief description of the files that may have been installed on your hard disk during the Setup process. DataList ActiveX selectively installs support files based on the version numbers of files already installed on your system.

Filename(s)	**Description**
MFC40.DLL	Support DLL (Microsoft Foundation Class DLL)
MFCO40.DLL	Support DLL (Microsoft Foundation Class DLL)
MFCANS32.DLL	Support DLL (Microsoft Foundation Class DLL)
MSVCRT40.DLL	Support DLL (Microsoft VC DLL)
OC30.DLL	Support DLL (32-Bit OLE Control Support DLL)
README.TXT	DataList ActiveX README file containing information that may not be in the help file.
GT-LD16.EXE	16-Bit List Designer OLE Server
GT-LD32.DLL	32-Bit List Designer OLE Server
GTLIST16.OCX	16-Bit DataList ActiveX OLE Controls
GTLIST32.OCX	32-Bit DataList ActiveX OLE Controls
LICENSE.TXT	GreenTree License information
DATALIST.HLP	DataList ActiveX Online Help
UNWISE.EXE	DataList uninstall program
VB40016.DLL	16-Bit Visual Basic Runtime DLL
VB40032.DLL	32-Bit Visual Basic Runtime DLL
TOPROCK.MDB	Sample database file used in sample programs
\SAMPLES	Directory containing Visual Basic 4.0 sample projects.

Item Method (ColumnDefs, ListItems, ListImages, SubItems collection)

Applies To

ColumnDefs collection**, ListItems** collection**, ListImages** collection**, SubItems** collection

Description

Returns a reference to an object in a collection.

Usage

object.**Item**(*index)*

The **Item** method has these parts:

Part	**Description**
object	An object expression that evaluates to a SubItems collection.
index	Required. An integer that uniquely identifies the object in the collection to be returned.

Remarks

Use the collection's **Count** property to determine how many objects are in the collection.

See Also

Count property.

Example

The following example returns the first object of the **GTList** control's **ListItems** collection:

```
Dim xListItem As ListItem
xListItem = GTList1.ListItems.Item(0)
```

ItemData Property

Applies To

GTList control, **GTCombo** control, **ListItem** object

Description

Returns or sets a specific number for each list item in a **GTList** control.

Usage

object.**ItemData(** *index* **)** [= *number*]

The **ItemData** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
index	A one-based index of a specific item in the **GTList** control, **GTCombo** control, or **ListItem** object.
number	The number to be associated with the specified item.

Remarks

The **ItemData** property is an array of long integer values with the same number of items as the control's **ListCount** property. You can use the numbers associated with each item to identify the items. For example, you can use a person's social security number (SSN) to identify each person in a **GTList** control. When you fill the **GTList**, also fill the corresponding elements in the **ItemData** array with the SSN numbers.

The **ItemData** property is often used as an index for an array of data structures associated with list items in a **GTList** control.

ItemData appears in the **ListItem** object for ListBox compatibility.

Note

When you insert an item into a **GTList** with the **AddItem** method, an item is automatically inserted in the **ItemData** array as well. However, the value isn't reinitialized to zero; it retains the value that was in that position before you added the item to the list. When you use the **ItemData** property, be sure to set its value when adding new items to a **GTList.**

See Also

ListItems collection**,** **ListItem** object**,** **GTList** control, **GTCombo** control.

Example

This example fills a **GTList** control with a person's name and fills the **ItemData** property array with their SSN number. The example uses the **NewIndex** property to keep the indices synchronized with the newly added items.

```
GTList1.AddItem "Jeff Spalding"
GTList1.ItemData(GTList1.NewIndex) = 555259423
```

Key Property (ColumnDef, ListItem, ListImage SubItem object)

Applies To

ColumnDef object, **ListItem** object, **ListImage** object, **SubItem** object.

Description

Specifies a unique Key used to identify the object

Usage

object.**Key** [= *string*]
The **Font3D** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
string	A string that uniquely identifies the object.

Remarks

Specifying a non-unique key string results in an error.

See Also

Index property, **GTList** control, **GTCombo** control

Example

The following code sets columndef's **Key** property to the string "`William`":

```
GTList1.ColumnDefs.Item(0).Key = "William"
```

LeftColumn Property

Applies To

GTList control**, GTCombo** control

Description

Sets or gets the left most visible column in the list. LeftColumn is a property array.

Usage

object.**LeftColumn([** *SubRow* **As Variant]) As Short**

The **LeftColumn** property array has these parts:

Part	Description
object	Required. An object expression that evaluates to one of the objects or controls in the Applies To list.
SubRow	Optional. A Variant specifying the subrow of the column object.

Remarks

This property returns or sets the left most visible column for the optional **SubRow.** If **SubRow** is omitted from the call, **SubRow** 0 is assumed. The return value can be used as an index into the **ColumnDefs** collection.

See Also

ColumnDefs collection, **ColumnDef** object, **SubRows** property, **SubColumn** property

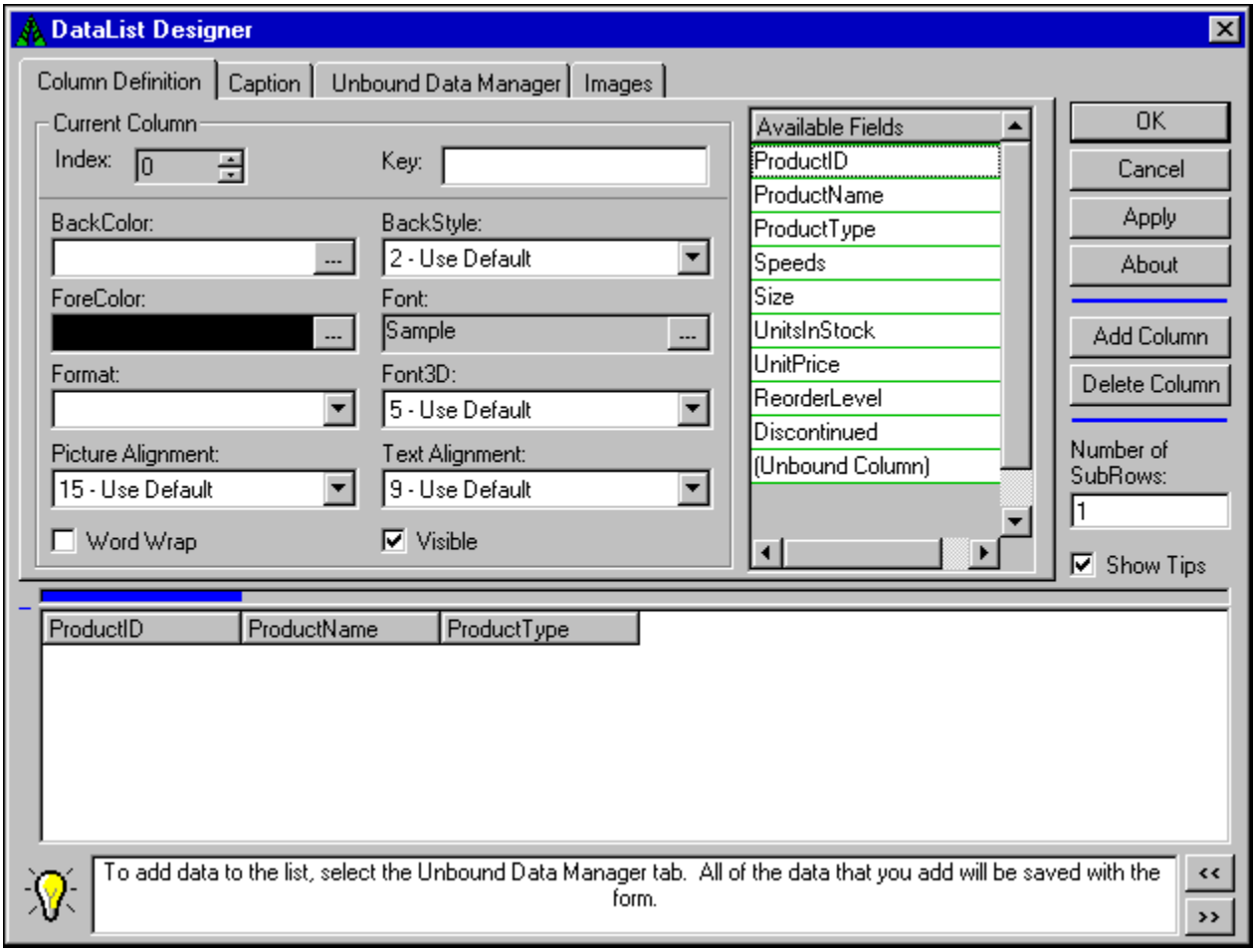
Example

This example sets the left most column to 5 for **SubRow** 2 in a **GTList** control.

```
GTList1.LeftColumn(5) = 2
```

DataList Designer

At design time, you can design and develop the entire look and feel of your **GTCombo** control by using GreenTree's **List Designer**. Use the List Designer to add column headers, sub rows, and view the results of data bound to the control at design time is a click and drag away. To access the **List Designer**, simply select the control by clicking on it, then using the right mouse button, click over the control. A context sensitive menu appears giving you several options related to the List Designer at the bottom of the menu. If you select the **List Designer...** menu option you will invoke the list designer tool. The screen looks similar to the one below:



Inside the DataList designer you can define columns, view column definitions that are bound to the control, and change all of the properties assoicated with the ColumnDefs collection. All of the tasks you normally would have to write code to produce can all be configured from within the DataList designer.

From within the DataList designer you can also add images to the control's internal image list.

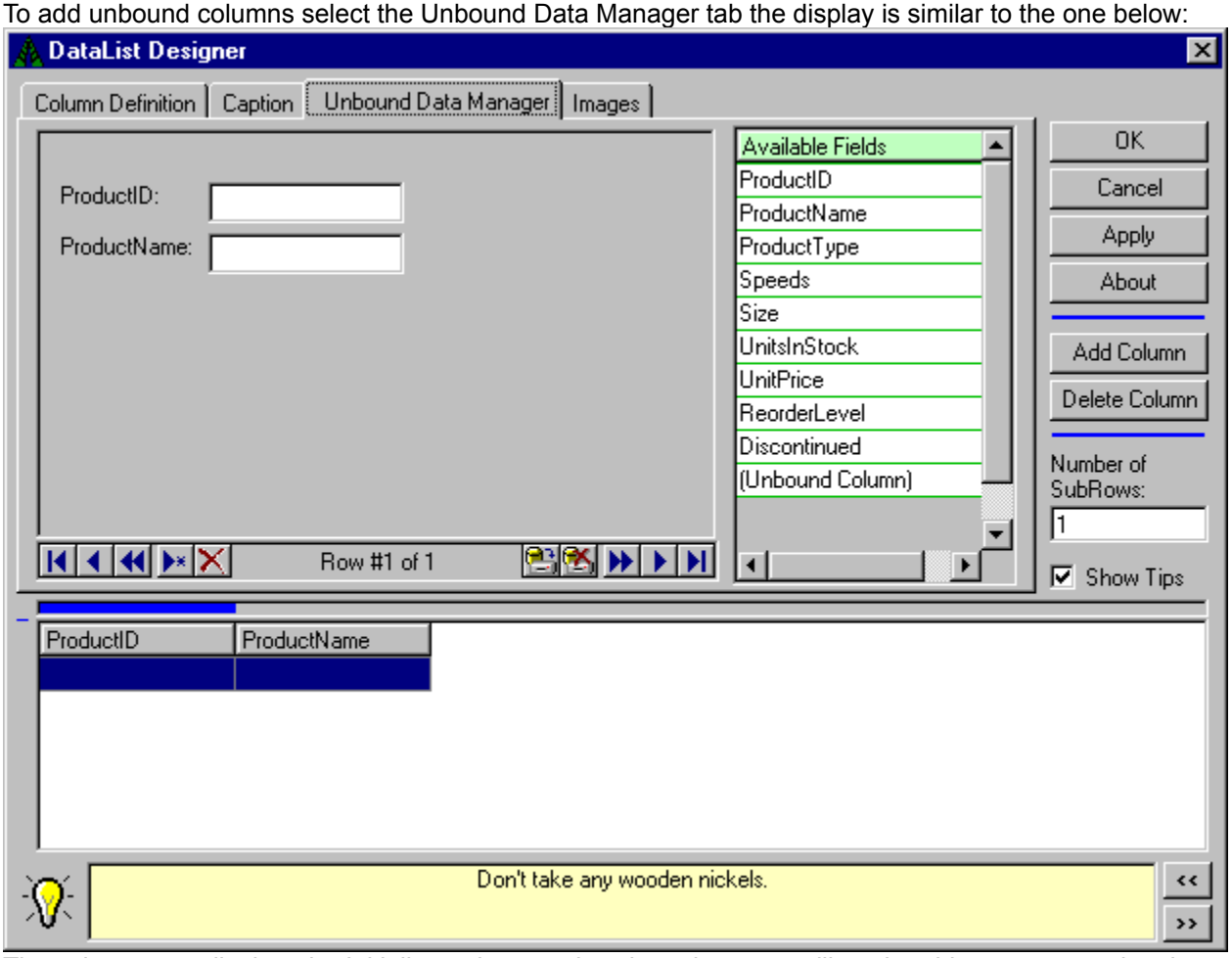
The context sensitive menu option **Retrieve Fields** retreives all of the field names contained in any database bound to the control. Column headers are automatically created, and added to the ColumnDefs collection for you. You may remove any column you wish by selecting the column from the columns displayed, and clicking the **Delete Column** button. When the column is selected a blue highlight appears above the column header button.

The context sensitive menu option **Clear Fields** removes all the columns defined in the list designer.

The DataList designer can mix and match bound and unbound column definitions, for display in your

application.

To add unbound columns select the Unbound Data Manager tab the display is similar to the one below:



The columns are displayed. Initially no data row is selected, so you will not be able to type any data into the text areas next to the columns. Press the Add Row button to select and edit a row of data. Any changes you make to bound data items will result in an update to the database.

ListCount Property

Applies To

GTList control, **GTCombo** control

Description

Returns the number of list items in a **GTList** control.

Usage

*object.***ListCount**

Returns the number of list item objects in the control. The *object* placeholder is a **GTList** or **GTCombo** control object.

Remarks

ListCount is always one more than the largest **ListItems** index value since the ListItems collection is a 0-based collection.

See Also

ListItems collection, **GTList** control.

Example

This example uses the **ListCount** property to iterate over the collection of **ListItems** in a **GTList** control, adding each item's text to a **ListBox** control.

```
For I = 0 to I < GTList.ListCount
      List1.AddItem GTList.ListItems(I).Text
Next I
```

ListImages Collection, ListImage Object

See Also Properties Methods Constants

The ListImages collection is a standard collection object that holds a series of ListImage objects. A ListImage object contains a picture to be associated with a ListItem object. The collection supports all of the standard methods associated with collections. Select one of the items above to see the Properties, Methods, and Constants associated with the ListImages collection.

The ListImages object defines each ListImage as it appears in the GTCombo or GTList window. Each item has support for:

- A unique key and index.
- A graphical image or picture

Usage

object.ListImages(*index*)

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

Remarks

ListImage objects can contain a unique index, key, and pictures. However, to use the pictures in the ListImages collection the parent control must have its **ImageList** property set to the string '`Internal`'. The internal ImageList can be filled with images during design time using the **ListDesigner** tool.

Listed below are all of the properties and methods supported by the collection and object. Any property or method that is underlined, is standard and not documented here. See the standard Microsoft documentation for information on those elements. Items marked with the symbol () apply only to the collection. All other properties apply only to the object.

Properties

Count	Index	Key
MemoryUsed	Picture	

Methods

Add	Clear	Item
Remove		

ListImages Collection, ListImage Object Methods

Add (Collection only)
Clear (Collection only)
Item (Collection only)
Remove (Collection only)

ListImages Collection, ListImage Object Properties

Underlined items are stock properties

Count (Collection Only)
MemoryUsed (Collection Only)

Index
Key
Picture

ListImages Property

Applies To

GTCombo control, GTList control

Description

Returns a reference to a collection of ListImage objects in an ImageList control. May refer to either the Internal image list or an external image list.

Usage

object.**ListImages**

The object placeholder represents an object expression that evaluates to an ImageList control.

Remarks

You can manipulate ListImage objects using standard collection methods (for example, the Add and Clear methods). Each member of the collection can be accessed by its index or unique key. These are stored in the Index and Key properties, respectively, when ListImage is added to a collection.

See Also

ImageList property, **ListImages** collection, **ListImage** object

Example

This example gets the key value for the first item in the ImageList collection:

```
Dim sKey As String
sKey = GTList1.ListImages.Item(0).Key
```

ListImages: See Also

GTList control reference
GTCombo control reference

ColumnDefs collection, ColumnDef object
ListItems collection, ListItem object
SubItems collection, SubItem object

ListIndex Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets the index of the currently selected item in the control. Not available at design time.

Usage

*object.***ListIndex** [= *number*]

The **ListIndex** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A long integer specifying the zero based index of the currently selected item as described in Settings.

Settings

The settings for *number* are:

Setting	Description
-1	(Default) Indicates no item is currently selected. For a GTCombo control, indicates the user has entered new text into the edit box portion.
n	A zero based index of the first item currently selected in the list.

Remarks

The expression `GTList1.ListItems(GTList1.ListIndex)` returns the string for the currently selected item.

The first item in the list is **ListIndex** = 0, and **ListCount** is always one more than the largest **ListIndex** value.

If users can make multiple selections (**MultiSelect** property), this property's behavior depends on the number of items selected. If only one item is selected, **ListIndex** returns the index of that item. In a multiple selection, **ListIndex** returns the index of the item contained within the focus rectangle, whether or not that item is actually selected.

See Also

NewIndex, ListCount, MultiSelect, TopIndex properties

Example

This example gets the text for the first selected list item, and assigns the text to a label's caption property.

```
Label1.Caption = GTList1.ListItems(GTList1.ListIndex).Text
```

ListItemDataRequest Event

Applies To

GTList control**, GTCombo** control

Description

The **ListItemDataRequest** event occurs when data is needed for a virtual ListItem object. The application needs to fill in the contents of the ListItem object. The index argument indicates the relative index of the virtual item contents

Usage

Private Sub *object*_**ListItemDataRequest(ByRef** *ListItem* **As ListItem**
ByVal *index* **As Long)**

The **ListItemDataRequest** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
ListItem	An object expression that evaluates to a **ListItem** object. The ListItem argument is a reference to the ListItem that needs the data.
index	The zero-based virtual index of the ListItem object that needs to be filled in with data from the user's application.

Remarks

The **VirtualItemCount** property specifies the virtual number of ListItem objects in the control. The **Virtual** property turns this mode on or off. The user specifies the number of virtual items the control will contain. Each time a ListItem object needs to be filled in with data, this event is fired to the application. The application needs to fill in the ListItem's data associated with the value passed in the *index* argument.

See Also

ListItem object, **VirtualItemCount** property, **Virtual** property.

Example

This example fills in the ListItem object and its SubItems collection with data calculated by several subroutines that calculate text strings and image indices.

```
Private Sub GTCombo1_ListItemDataRequest( ByRef ListItem As ListItem,
                                          ByVal index As Long)
      listitem.Text = CalcText( index )
      listitem.Image = CalcImage( index )
      For I = 0 to nSubItemCount
            listitem.SubItems(I).Text = CalcSubText(I, index)
            listitem.SubItems(I).Image = CalcSubImage(I, index)
      Next I
End Sub
```

ListItems Collection, ListItem Object

See Also Properties Methods Constants

The ListItems collection is a standard collection object that holds a series of ListItem objects. A ListItem object consists of text, an index of an associated icon, and a collection of SubItems for displaying additional information about the ListItem. The collection supports all of the standard methods associated with collections. Select one of the items above to see the Properties, Methods, and Constants associated with the ListItems collection.

The ListItem object defines each item as it appears in the GTCombo or GTList window. Each item has support for:

- Individual BackColor and ForeColor per item.
- Individual font and styles
- Alignment of text and images
- A SubItems collection that contains additional data about the item, and a mirror of the list item's properties in SubItem(0) of the SubItems collection.

Usage

object.ListItems(*index*)

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

Remarks

ListItem objects can contain both text and pictures. However, to use the pictures the parent control, must have an **ImageList** associated with it.

Listed below are all of the properties and methods supported by the collection and object. Any property or method that is underlined, is standard and not documented here. See the standard Microsoft documentation for information on those elements. Items marked with the symbol () apply only to the collection. All other properties apply only to the object.

Properties

BackColor	BackStyle	Count
Font	Font3D	ForeColor
Format	Height	Image
Index	ItemData	Key
MemoryUsed	PictureAlignment	Selected
SelectedImage	SubItems	TagVariant
Text	TextAlignment	WordWrap

Methods

Add	Clear	Item
Remove		

ListItems Collection, ListItem Object Methods

Add (Collection Only)
Clear (Collection Only)
Item (Collection Only)
Remove (Collection Only)

ListItems Collection, ListItem Object Properties

Underlined items are stock properties

Count (Collection Only)
MemoryUsed (Collection Only)

BackColor
BackStyle
Font
Font3D
ForeColor
Format
Height
Image
Index
ItemData
Key
PictureAlignment
Selected
SelectedImage
SubItems
TagVariant
Text
TextAlignment
WordWrap

ListItems Property

Applies To

GTList control**, GTCombo** control

Description

Returns a reference to a collection of **ListItem** objects.

Usage

object.**ListItems**

The object placeholder represents an object expression that evaluates to one of the controls listed in the Applies To section.

Remarks

You can manipulate **ListItem** objects using standard collection methods (for example, the **Remove** method). Each **ListItem** in the collection can be accessed either by its index or by a unique key, stored in the **Key** property.

See Also

ListItems collection**, ListItem** object**, Key** property.

Example

The following example removes the first **ListItem** object from the collection:

```
GTCombo1.ListItems.Remove 1
```

ListItems: See Also

GTCombo control reference
GTList control reference

ColumnDefs collection, ColumnDef object
ListImages collection, ListImage object
SubItems collection, SubItem object

MaskColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the masking color used when masking transparent bitmaps for selected items in the list.

Usage

object.**MaskColor** [= *color*]

The **MaskColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the color used to mask the transparent bitmap for listitems when the items are selected in the list, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

See Also

MaskColorEnabled property

Example

The following code sets the **MaskColor** to Yellow using the **RGB** function:

```
GTCombo1.MaskColor = RGB(255,255,0)
```

MaskColorEnabled Property

Applies To

GTList control

Description

Specifies if transparent bitmaps are masked using the MaskColor property when selected.

Usage

object.**MaskColorEnabled** [= *boolean*]

The **MaskColorEnabled** property has these parts:

Part	Description
object	An object expression that evaluates to a **GTList** control.
boolean	A boolean expression specifying if ListItem bitmaps are to be masked with the MaskColor property when selected in the list.

Remarks

Setting this property to True will cause the ListItem bitmaps to be masked with the MaskColor property when selected in the list.

See Also

MaskColor, Selected properties

Example

The following code sets the **MaskColorEnabled** property to the value of a CheckBox control for a **GTList** control:

```
GTList1.MaskColorEnabled = CheckBox1.Value
```

MaxDropDownItems Property

Applies To

GTCombo control

Description

Determines the maximum number of items allowed to be shown in the control's drop down box.

Usage

object.**MaxDropDownItems** [= *number*]

The **MaxDropDownItems** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the maximum number of items to show when the drop down box is activated.

Settings

The settings for *number* are:

Value	Description
>0	The number of items to show in the drop down list box. Default = 10

Remarks

Use this property to control how many items are displayed when the drop down list box is shown for the control. The value in this property must be greater than or equal to the value in MinDropDownItems. If the number of items is too high, the bottom of the list box is clipped at the bottom of the screen. This may prevent users from navigating the list with parts of the control's scrollbar.

See Also

DropDownWidth property, **MinDropDownItems** property

Example

The following code sets the MaxDropDownItems to 15 for a GTCombo control:

```
GTCombo1.MaxDropDownItems = 15
```

MemoryUsed Property (ColumnDefs, ListItems, ListImages, SubItems collection)

Applies To

ColumnDefs collection**, ListItems** collection**, ListImages** collection**, SubItems** collection

Description

Returns amount of memory being used by the collection.

Usage

object.**MemoryUsed**

The object placeholder represents an object expression that evaluates to one of the collections listed in the Applies To section.

Remarks

The **MemoryUsed** property is associated with the collection and not the control itself. This prepay value indicates the amount of memory being used by the collection and all of its associated objects.

See Also

Count property.

Example

The following example displays the amount of memory being used by a **ListItems** collection:

```
Label1.Caption = "Memory Used: " +
                         STR$(GTList1.ListItems.MemoryUsed)
```

MinDropDownItems Property

Applies To

GTCombo control

Description

Determines the minimum number of lines to be shown in the control's drop down box.

Usage

object.**MinDropDownItems** [= *number*]

The **MinDropDownItems** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the minimum number of lines to show when the drop down box is activated. Default = 5.

Settings

The settings for *number* are:

Value	Description
>0	The minimum number of lines to show in the drop down list box.

Remarks

Use this property to control how many lines are displayed when the drop down list box is shown for the control. The value must be equal to or less than MaxDropDownItems. If the number of items in the list is less than MinDropDownItems, the list is padded with blank lines.

See Also

DropDownWidth property, **MaxDropDownItems** property

Example

The following code sets the MinDropDownItems to 3 for a GTCombo control:

```
GTCombo1.MinDropDownItems = 3
```

MoveTo Method (ColumnDefs collection)

Applies To

ColumnDef object

Description

Moves a column to a new SubColumn/SubRow location.

Usage

object.**MoveTo** (*SubColumn*, *SubRow*)

The **MoveTo** method has these parts:

Part	Description
object	An object expression that evaluates to a SubItems collection.
SubColumn	The new sub-column index of the ColumnDef object in the **ColumnDefs** collection.
SubRow	Specifies the desired new SubRow index of the ColumnDef object in the ColumnDefs collection.

Remarks

Use this method to move the ColumnDef object to a new location. As the column is moved from one location to another, the index of that column changes in the ColumnDefs collection. If the source of the column is lower than the dest, all columns between the source and the dest are demoted one index number. If the source is larger than the dest, all columns between the dest and the source are promoted one index number. Before the column object is moved, the **ColumnBeforeMove** Event is fired. After the column object is moved, the **ColumnAfterMove** Event is fired.

See Also

Count , SubColumn, SubRows properties, **ColumnBeforeMove, ColumnAfterMove** events.

Example

The example below moves a ColumnDef object ColumnDefs collection to SubColumn 4, SubRow 2 for a GTList control:

```
GTList1.ColumnDefs(1).MoveTo 4, 2
```

MultiSelect Property

Applies To

GTList control

Description

Specifies the type of multiple selecting is allowed by the user.

Usage

object.**MultiSelect** [= *number*]

The **MultiSelect** property has these parts:

Part	Description
object	An object expression that evaluates to a **GTList** control.
number	A number or value specifying the type of multiple selecting the user is allowed to perform for all the control's objects as specified in Settings.

Settings

The settings for *number* are:

Setting	Description
0	(Default) None. Multiple selection is not allowed.
1	Simple. Simple multiple selection. A mouse click or pressing the SPACEBAR selects or deselects an item in the list. (Arrow keys move the focus.)
2	Extended. Extended multiple selection. Pressing SHIFT and clicking the mouse or pressing SHIFT and one of the arrow keys (UP ARROW, DOWN ARROW, LEFT ARROW, and RIGHT ARROW) extends the selection from the previously selected item to the current item. Pressing CTRL and clicking the mouse selects or deselects an item in the list.

Remarks

Use this property to specify how the user can select items in the control.

See Also

ListItems collection, **ListItem** object, **Selected** property

Example

The following code sets the MultiSelect property to the value of a ListBox index for a **GTList** control:

```
GTList1.MultiSelect = ListBox1.ListIndex
```

NewIndex Property

Applies To

GTList control**, GTCombo** control

Description

Returns the index of the item most recently added to a **GTList** or **GTCombo** control.

Usage

*object.***NewIndex**

The object placeholder represents any of the controls listed in the Applies To section.

Remarks

You can use this property with sorted GTLists when you need a list of values that correspond to each item in the **ItemData** property array. As you add an item in a sorted **GTList**, the control inserts the item in the **GTList** in alphabetic order. This property tells you where the item was inserted so that you can insert a corresponding value in the **ItemData** property at the same index.

The **NewIndex** property returns -1 if there are no items in the list or if an item has been deleted since the last item was added.

See Also

ListItems collection, **ListItem** object, **Selected** property

Example

This example fills a **GTList** control with a person's name and fills the **ItemData** property array with their SSN number. The example uses the **NewIndex** property to keep the indices synchronized with the newly added items.

```
GTList1.AddItem "Jeff Spalding"
GTList1.ItemData(GTList1.NewIndex) = 555259423
```

Picture Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets a graphic to be displayed in the background of a **GTList** or **GTCombo** control.

Usage

*object.***Picture** [= *picture*]

The **Picture** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
picture	A string expression that designates a bitmap or icon to display in the control's background, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
None	No picture is displayed in the background of the control.
(Bitmap, icon, metafile)	A string expression that designates a bitmap or icon to in the control's background.

Remarks

At design time, you set the **Picture** property for a control by setting the property in the Properties window. At run time, you can set the **Picture** property using the **LoadPicture** function or the **Picture** property of another control or of a Form object.

When setting the **Picture** property at design time, the graphic is saved and loaded with the **GTList** or **GTCombo** control.

See Also

Backcolor , PictureAlignment properties

Example

This example sets the background **Picture** property of a **GTList** control to a picture loaded at runtime using the **LoadPicture** function.

```
GTList1.Picture = LoadPicture("marble.bmp")
```

Picture Property (ListImage object)

Applies To

ListImage object

Description

Returns or sets a graphic to be stored in the listimage object.

Usage

object.**Picture** [= *picture*]

The **Picture** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
picture	A string expression that designates a bitmap or icon to store in the object, as described in Settings.

Settings

The settings for *picture* are:

Setting	Description
None	No picture is displayed in the background of the control.
(Bitmap, icon, metafile)	A string expression that designates a bitmap or icon to store in the listimage object.

Remarks

At design time, you set the **Picture** property for a listimage by setting the property in the custom property page. You can fill in the listimages collection this way. At run time, you can set the object's **Picture** property using the **LoadPicture** function or the **Picture** property of another control or of a Form object.

When setting the **Picture** property at design time, the graphic is saved and loaded with the **GTList** or **GTCombo** control.

See Also

PictureAlignment property

Example

This example sets the listimage object's **Picture** property to a picture loaded at runtime using the **LoadPicture** function.

```
GTList1.ListImages(0).Picture = LoadPicture("phone.bmp")
```

PictureAlignment Property

Applies To

GTList control**, GTCombo** control**, ColumnDef** Object, **ListItem** Object, **SubItem** Object,

Description

Specifies the picture alignment used for all list item images.

Usage

object.**PictureAlignment** [= *number*]

The **PictureAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A number or value specifying the picture alignment as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtPictureAlignmentLeftTop	0	Left Top. Image is left/top aligned.
gtPictureAlignmentLeftMiddle	1	Left Middle. Image is left/middle aligned.
gtPictureAlignmentLeftBottom	2	Left Bottom. Image is left/bottom aligned.
gtPictureAlignmentRightTop	3	Right Top. Image is right/top aligned.
gtPictureAlignmentRightMiddle	4	Right Middle. Image is right/middle aligned.
gtPictureAlignmentRightBottom	5	Right Bottom. Image is right/bottom aligned.
gtPictureAlignmentCenterTop	6	Center Top. Image is center/top aligned.
gtPictureAlignmentCenterMiddle	7	Center Middle. Image is center/middle aligned.
gtPictureAlignmentCenterBottom	8	Center Bottom. Image is center/bottom aligned.
gtPictureAlignmentStretch	9	Stretch to fit. Image is stretched to fit image area.
gtPictureAlignmentTile	10	Tile. Image is tiled in the image area.

Object Only Settings	Value	Description
gtPictureAlignmentLeftOfText	11	Picture is aligned to the left of the object's text.
gtPictureAlignmentRightOfText	12	Picture is aligned to the right of the object's text.
gtPictureAlignmentAboveText	13	Picture is aligned above the object's text.
gtPictureAlignmentBelowText	14	Picture is aligned below the object's text.
gtPictureAlignmentUseDefault	15	Use the value in DefPictureAlignment property.

Remarks

Use this property to get or set the picture alignment for all the control and object images. Only the objects can use the values between 11 and 15 inclusive. Trying to apply a value above 10 for a control object results in a run-time error.

See Also

DefPictureAlignment property, **ListItems** collection**, ListItem** object

Example

The following code sets the control's picture alignment to Center Bottom for a **GTList** control

```
GTList1.PictureAlignment = 8
```

PositionList Event

Applies To

GTCombo control

Description

Occurs when the list items are automatically positioned by the user typing text in the edit portion of the control.

Usage

Private Sub *object*_**PositionList()**

The *object* expression evaluates to one of the objects or controls in the Applies To list.

Remarks

This event is fired when the user types text in the edit portion of the control, and the list is automatically positioned to match the characters typed by the user.

See Also

AutoPostitionList property, **ListIndex** property, **TopIndex** property.

Example

This example displays the text from the first listitem in the list when the position of the list automatically changes:

```
Private Sub GTCombo1_PositionList()
      Label1.Caption = GTCombo1.ListItems(GTCombo1.TopIndex).Text
End Sub
```

Quick Note: About writing sample programs

All of the sample programs have been written using Microsoft Visual Basic 4.0. Each of the following exercises assumes that you will be using Visual Basic 4.0 in either a 16-bit or 32-bit environment. It is assumed that you have loaded the DataList ActiveX controls into the Visual Basic environment.

Throughout the exercises the TOPROCK.MDB database file is used. This file is a Microsoft Access database file that is shipped with the DataList ActiveX package. The Setup program installs this file onto your system for you in the SAMPLE directory.

Quick Tour - Writing Sample Programs

Quick Note: About writing sample programs

GTList Control

- Excercise 1: Binding Data to the control
- Excercise 2: Customizing the user's interface
- Excercise 3: Adding pictures to your application.

GTCombo Control

- Excercise 1: Binding Data to the control
- Excercise 2: Customizing the look and feel
- Excercise 3: Adding list items, and sorting.
- Excercise 4: Virtual lists.

Refresh Method

Applies To

GTList control**, GTCombo** control

Description

Forces a complete repaint of a control.

Usage

object.**Refresh**

The object placeholder represents an object expression that evaluates to one of the objects or controls listed in the Applies To section.

Remarks

Generally, painting a control is handled automatically while no events are occurring. However, there may be situations where you want the control updated immediately. For example, if you use the GTList as a file list box, or a directory list box to show the current files in a directory structure, you can use Refresh to update the list whenever a change is made to the directory structure.

See Also

ListItems collection**, ListItem** object.

Example

This example calls the **Refresh** method for a **GTList** control:

```
GTList1.Refresh
```

Remove Method (ColumnDefs, ListItems, ListImages, SubItems collection)

Applies To

ColumnDefs collection**, ListItems** collection**, ListImages collection, SubItems** collection

Description

Removes all the objects from the collection rendering the collection empty.

Usage

object.**Remove** *index*

The **Remove** method has these parts:

Part	**Description**
object	An object expression that evaluates to a SubItems collection.
index	Required. An integer that uniquely identifies the object in the collection to be removed.

Remarks

To remove all the objects from a collection, use the **Clear** method.

See Also

Clear method.

Example

The following example removes the first object of the **GTList** control's **ListItems** collection:

```
GTList1.ListItems.Remove 1
```

RemoveItem Method

Applies To

GTList control**, GTCombo** control

Description

Removes an item from a GTList or GTCombo control. Doesn't support named arguments.

Usage

object.**RemoveItem** *index*

The **RemoveItem** method has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
index	Required. Integer representing the position within the object of the item or row to remove. For the first item in a **GTList** or **GTCombo** control, index = 0..

Remarks

A **GTList** or **GTCombo** that is bound to a Data control doesn't support the **RemoveItem** method.

See Also

TopIndex property**, AddItem** method**, Clear** method.

Example

The following example removes the first visible item from a **GTList** control:

```
Dim nIdx As Integer
nIdx = GTList1.GetFirstVisible().Index
GTList1.RemoveItem nIdx
```

RowHeight Property (ColumnDef object)

Applies To

ColumnDef object

Description

Returns or sets the height of the specified list row (record) in twips. Not available at design time.

RowHeight is always in the same unit of measure as the container for the object.

Usage

object. **RowHeight** [= *number*]

The **RowHeight** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
number	A number or value specifying the height of the column's rows as specified in Settings.

Remarks

The minimum RowHeight is 1 pixel. Use this property to get or set the height of the ColumnDef object's rows. Setting this property to 0 causes the ColumnDef object to use the value in the **DefRowHeight** property as the row height.

See Also

DefRowHeight property

Example

The following code sets the list item object's RowHeight property to a specific value in twips for a ColumnDef object:

```
GTList1.ColumnDefs(0).RowHeight = 400
```

ScrollTipBackColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the BackColor used for all ScrollTip items.

Usage

object.**ScrollTipBackColor** [= *color*]

The **ScrollTipBackColor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the background color of the ScrollTip item, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

See Also

ScrollTipForeColor property

Example

The following code sets the **ScrollTipBackColor** to Yellow using the **RGB** function:

```
GTCombo1.ScrollTipBackColor = RGB(255,255,0)
```

ScrollTipBegin Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the scroll tip of the control is about to be displayed next to the control's scrollbar thumb.

Usage

Private Sub *object*_**ScrollTipBegin(**Cancel **As Boolean)**

The **ScrollTipBegin** event has these parts:

Part	Description
object	An object expression that evaluates to one of the objects or controls in the Applies To list.
Cancel	A reference to a boolean that determines if the scroll tip should be displayed..

Remarks

To prevent the scroll tip from being displayed, set Cancel to **True**. Leaving Cancel unmodified or setting Cancel to **False** will allow the scroll tip to be displayed. If the scroll tip is canceled, this event will be fired when the next **ScrollTipDelay** time-out occurs.

See Also

ScrollTipEnd, ExtendTipBegin, ExtendTipEnd events.

Example

This example cancels the scroll tip for a **GTCombo** control.

```
Private Sub GTCombo1_ScrollTipBegin (Cancel As Boolean)
      Cancel = True
End Sub
```

ScrollTipDataField Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets a value that binds a **Field** object to the ScrollTip.

Usage

object.**ScrollTipDataField** [= *value*]

The **ScrollTipDataField** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	A string expression that evaluates to the name of one of the fields in the Recordset object specified by the **DataSource** property.

Remarks

The ScrollTips of a GTList or GTCombo control can be bound to a database field. This property specifies which field in the current data record is used as the text for the ScrollTip item.

See Also

DataSource property

Example

The example below sets the **ScrollTipDataField** to the `Titles` field. This example assumes that a **DataSource** exists and that the data source has a Titles field in its record structure.

```
GTCombo1.ScrollTipDataField = "Titles"
```

ScrollTipEnd Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the scroll tip is removed from the display.

Usage

Private Sub *object*_**ScrollTipEnd()**

The *object* expression evaluates to one of the objects or controls in the Applies To list.

Remarks

This event is fired when the **ScrollTipTimeout** has expired and the scroll tip is removed from the display.

See Also

ScrollTipBegin, ExtendTipBegin, ExtendTipEnd events.

Example

This example shows the syntax of the ScrollTipEnd function.

```
Private Sub GTCombo1_ScrollTipEnd ()

End Sub
```

ScrollTipFont Property

Applies To

GTList control**, GTCombo** control

Description

Returns the font used for the ScrollTip.

Usage

object.**ScrollTipFont**

Returns a font object used for the ScrollTip objects. The *object* placeholder can be any of the controls listed in the Applies To section.

Remarks

When the ScrollTip's text is displayed this font is used.

See Also

ScrollTips property

Example

The following code sets the **ScrollTipFont**'s bold value **True** for a **GTCombo** control:

```
GTCombo1.ScrollTipFont.Bold = True
```

ScrollTipForeColor Property

Applies To

GTList control**, GTCombo** control

Description

Specifies the forecolor used for the text for ScrollTips.

Usage

object.**ScrollTipForeColor** [= *color*]

The **ScrollTipForecolor** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
color	A value or constant that determines the forecolor color of the text displayed for all ScrollTips, as described in Settings.

Settings

The settings for color are:

Setting	Description
Normal RGB colors	Colors specified by using the Color palette or by using the RGB or QBColor functions in code.
System default colors	Colors specified by system color constants listed in the Visual Basic (VB) object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

Remarks

Use this property to get or set the foreground color value for the text displayed for the ScrollTips.

See Also

ScrollTipBackColor property

Example

The following code sets the control's **ScrollTipForeColor** to the first ListItem's **ForeColor** for a **GTList** control:

```
GTList1.ScrollTipForeColor = GTList1.ListItems(0).ForeColor
```

ScrollTips Property

Applies To

GTList control**, GTCombo** control

Description

Enables or disables ScrollTips for the control's scroll bars.

Usage

object.**ScrollTips** [= *boolean*]
The **ScrollTips** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression specifying if the ScrollTips are enabled.

Remarks

Setting this value to True enables the ScrollTips on the control's scroll bars.

See Also

ExtendTips property

Example

The following code turns the control's ScrollTips off by setting the **ScrollTips** property to **False**.

```
GTList1.ScrollTips = False
```

See Also (ColumnDefs)

GTCombo control reference

GTList control reference

ListItems Collection, ListItem Object

ListImages Collection, ListImage Object

SubItems Collection, SubItem Object

DataList Controls Overview: See Also

GTCombo Control Reference
GTCombo Control Quick Tour

GTList Control Reference
GTList Control Quick Tour

ColumnDefs collection, ColumnDefs object
ListImages collection, ListImage object
ListItems collection, ListItem object
SubItems collection, SubItem object

See Also (GTCombo Excercise 2)

GTCombo control reference

Excercise 1: Binding data to the control
Excercise 3: Adding pictures to your application

See Also (GTCombo Excercise 3)

GTCombo control reference

Excercise 1: Binding data to the control
Excercise 2: Customizing the user's interface

GTCombo Quick Tour: See Also

GTCombo control reference
GTCombo Control Reference

ColumnDefs collection, ColumnDef object
ListImages collection, ListImage object
ListItems collection, ListItem object
SubItems collection , SubItem object

Excercise 2: Customizing the user's Interface
Excercise 3: Adding pictures to your application

See Also (GTList Excercise 2)

GTList control reference

Excercise 1: Binding data to the control
Excercise 3: Adding list items and sorting
Excercise 4: Virtual lists

See Also (GTList Excercise 3)

GTList control reference

Excercise 1: Binding data to the control
Excercise 2: Customizing the look and feel
Excercise 4: Virtual lists

See Also (GTList Excercise 4)

GTList control reference

Excercise 1: Binding data to the control
Excercise 2: Customizing the look and feel
Excercise 3: Adding list items, and sorting

SelCount Property

Applies To

GTList control**, GTCombo** control

Description

Returns the number of list items selected in the control.

Usage

*object.***SelCount**

The object placeholder represents an object expression that evaluates to any of the controls listed in the Applies To section.

Remarks

The **SelCount** property returns 0 if no items are selected. Otherwise, it returns the number of list items currently selected. This property is particularly useful when users can make multiple selections.

See Also

Selected property, **MultiSelect** property

Example

The following code displays the number of selected items in a Label for a **GTList** control

```
Label1.Caption = STR$(GTList1.SelCount) + "Items Selected."
```

SelLength SelStart SelText Properties

Applies To

GTCombo control

Description

SelLength returns or sets the number of characters selected.

SelStart returns or sets the starting point of text selected; indicates the position of the insertion point if no text is selected.

SelText returns or sets the string containing the currently selected text; consists of a zero-length string ("") if no characters are selected.

These properties are not available at design time.

Usage

object.**SelLength** [= *number*]

object.**SelStart** [= *index*]

object.**SelText** [= *value*]

The **SelLength, SelStart,** and **SelText** properties have these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
mode	A numeric expression specifying the number of characters selected. For **SelLength** and **SelStart**, the valid range of settings is 0 to text length the total number of characters in the edit area of a **GTCombo** control.
index	A numeric expression specifying the starting point of the selected text, as described in Settings.
value	A string expression containing the selected text.

Remarks

Use these properties for tasks such as setting the insertion point, establishing an insertion range, selecting sub-strings in a control, or clearing text. Used in conjunction with the Clipboard object, these properties are useful for copy, cut, and paste operations.

When working with these properties:

Setting **SelLength** less than 0 causes a run-time error.

Setting **SelStart** greater than the text length sets the property to the existing text length; changing **SelStart** changes the selection to an insertion point and sets **SelLength** to 0.

Setting **SelText** to a new value sets **SelLength** to 0 and replaces the selected text with the new string.

See Also

Text property

Example

The following code sets the selection's text (**SelText**) to a new value for a **GTCombo** control:

```
GTCombo1.SelText = "Skis, Boots, Poles"
```

Selected Property

Applies To

GTList control, **GTCombo** control, **ListItem** Object

Description

Returns the number of list items selected in the control.

Usage

object.**Selelected(** *index* **)** [= *boolean*]
The **Selected** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
index	A zero based index specifying the list item in the control.
boolean	A boolean expression determining if the list item is selected.

Remarks

The index will always be at least one less than the GTList's **ListCount** property.

See Also

ListCount property, **ListItems** collection, **ListItem** object

Example

The following code sets a listbox control to the text in each Selected item for a **GTList** control

```
For I = 0 to GTList1.ListCount
      If GTList1.Selected(I)
            ListBox1.Add = ListItems(I).Text
      End If
Next I
```

SelectedImage Property (ListItem object, SubItem object)

Applies To

ListItem object, SubItem object

Description

Returns or sets the index or key value of a ListImage object in an associated ImageList control; the ListImage is displayed when a ListItem or SubItem object is selected.

Usage

object.**SelectedImage** [= *index*]

The SelectedImage property syntax has these parts:

Part	Description
object	An object expression that evaluates to a Node object.
index	An integer or unique string that identifies a ListImage object in an associated ImageList control. The integer is the value of the ListImage object's **Index** property; the string is the value of the **Key** property.

Remarks

If **Null,** the mask of the default image specified by the Image property is used.

See Also

ListImages collection**, ListImages** object**, ImageList** property.

Example

The following example changes the selected image for the most recently selected list item:

```
GTCombo1.SelectedItem.SelectedImage = 1
```

SelectedItem Property

Applies To

GTList control**, GTCombo** control

Description

Returns a reference to the most recently selected **ListItem** object.

Usage

object.**ListItems**

The object placeholder represents an object expression that evaluates to one of the controls listed in the Applies To section.

Remarks

Use this property to determine the most recently selected list item. You can manipulate **ListItem** objects using standard object properties (for example, the **Format** property).

See Also

ListItems collection**, ListItem** object**, Selected** property.

Example

The following example changes the text alignment for the most recently selected list item:

```
GTCombo1.SelectedItem.TextAlignment = gtTextAlignmentLeftTop
```

SetDefaults Method (SubItem object)

Applies To

SubItem object

Description

Sets the SubItem object properties to their Use Default value as described by the **DefXXXX** properties in the parent control.

Usage

*object.***SetDefaults**

The object placeholder represents an object expression that evaluates to one of the objects listed in the Applies To section.

Remarks

Sets the value of each of the properties below to it's associated **gtXXXXUseDefault** value. This causes the property to have the characteristics of the **DefXXXX** proeprty defaults for all of the following properties

Property	Default Property
TextAlignment	DefTextAlignment
BackColor	DefBackColor
BorderStyle	DefBorderStyle
ForeColor	DefForeColor
Font	DefFont
Font3D	DefFont3D
Image	DefImage
PictureAlignment	DefPictureAlignment

See Also

None

Example

This example sets the defaults for the first SubItem used by the control's ListItems collection:

```
GTList1.ListItems.SubItems.Item(0).SetDefaults
```

SortKey Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a value that determines how the **ListItem** objects in a **GTList** or **GTCombo** control are sorted.

Usage

object.**SortKey** [= *number*]

The **SortKey** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the sortkey as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Sort using the ListItem object's Text property.
> 0	Sort using this SubItem number.

Remarks

The **Sorted** property must be set to **True** before the change takes place.

The control can automatically sorts the list when the column header is clicked based on the **Sorted**, **SortOrder** and **SortKey** properties. You may want to change the sort order based on external criteria, and resort the list in a ClickColumn event. The list is only sorted when the control is non-bound and non-virtual. Specifically, the control can not be bound via a DataSource and the Virtual property must be False for sorting to occur.

Note: Sorting by SubItem number is not the same as sorting by column number. Each column can be connected to any SubItem, thus sorting is done via the SubItem number. For instance, if ColumnDefs(2) has 15 as its' SubItemSource, SortKey needs to be set to 15 not 2. for the desired sorting.

See Also

Sorted, SortKey, SortOrder, SubItemSource properties, **ClickColumn** event

Example

The following code sets the **SortKey** and **SortOrder** based on the external criteria from a **GTList** control**,** for a **GTCombo** control:

```
Private Sub GTCombo1_ClickColumn (ByVal ColumnHeader As ColumnHeader)
      If Check1.value = CHECKED Then
            GTCombo1.SortKey= GTList2.ListIndex
            GTCombo1.SortOrder = GTList2.SortOrder
      End If
End Sub
```

SortKey2 Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a secondary key value that determines how the **ListItem** objects in a **GTList** or **GTCombo** control are sub-sorted.

Usage

object.**SortKey2** [= *number*]

The **SortKey2** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the secondary sort key as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Perform secondary sort using the ListItem object's Text property.
> 0	Perform secondary sort using this SubItem number.

Remarks

The **Sorted** property must be set to **True** before the change takes place. The list is only sorted when the control is non-bound and non-virtual. Specifically, the control can not be bound via a DataSource and the Virtual property must be False for sorting to occur.

Note: Sorting by SubItem number is not the same as sorting by column number. Each column can be connected to any SubItem, thus sorting is done via the SubItem number. For instance, if ColumnDefs(2) has 15 as its' SubItemSource, SortKey needs to be set to 15 not 2. for the desired sorting.

The control can automatically sorts the list when the column header is clicked based on the **Sorted**, **SortOrderXXX** and **SortKeyXXX** properties. You may want to change the sort order based on external criteria, and resort the list in a **ClickColumn** event. The list will first be sorted based on the primary **SortKey** property, then the list is sub-sorted based on the secondary key value in this property.

As an example, suppose your list items have as their **Text** property, the last name of a person, the first SubItem contains the person's first name, and the second SubItem has the person's city name. If the **SortKey** property contains **0**, and the **SortOrder** property contains **gtSortAscending**, the list of names would first be sorted in ascending order based on the last name. The first names might come out in any random order. If **SortKey2** is set to the first SubItem (the first name), and **SortOrder2** is set to **gtSortAscending**, the list is sub-sorted based on the first name. Duplicate last names are sub-sorted based on the first name. The city names might come out in any random order as shown below:

SortKey / SortOrder

```
Smith   | Jane  |  Chicago
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
Smith   | Jane  |  San Francisco
```

SortKey2 / SortOrder2

```
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
Smith   | Jane  |  San Francisco
Smith   | Jane  |  Chicago
```

See Also

Sorted, SortKey, SortKey3 properties**, SortOrder, SortOrder2, SortOrder3** properties, **ClickColumn** event

Example

The following code sets the **SortKey2** and **SortOrder** based on the external criteria from a **GTList** control**,** for a **GTCombo** control:

```
Private Sub GTCombo1_ClickColumn (ByVal ColumnHeader as
                                  ColumnHeader)
      If Check1.value = CHECKED Then
            GTCombo1.SortKey2= GTList2.ListIndex
            GTCombo1.SortOrder = GTList2.SortOrder
      End If
End Sub
```

SortKey3 Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a secondary key value based on the sorting results from **SortKey2** that determines how the **ListItem** objects in a **GTList** or **GTCombo** control are sub-sorted.

Usage

object.**SortKey3** [= *number*]

The **SortKey3** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	An integer specifying the secondary sort key as described in Settings.

Settings

The settings for *number* are:

Setting	Description
0	Perform secondary sort after sorting based on **SortKey2** using the ListItem object's Text property.
> 0	Perform secondary sort after sorting based on **SortKey2** using this SubItem number.

Remarks

The **Sorted** property must be set to **True** before the change takes place. The list is only sorted when the control is non-bound and non-virtual. Specifically, the control can not be bound via a DataSource and the Virtual property must be False for sorting to occur.

Note: Sorting by SubItem number is not the same as sorting by column number. Each column can be connected to any SubItem, thus sorting is done via the SubItem number. For instance, if ColumnDefs(2) has 15 as its' SubItemSource, SortKey needs to be set to 15 not 2. for the desired sorting.

The control can automatically sorts the list when the column header is clicked based on the **Sorted**, **SortOrderXXX** and **SortKeyXXX** properties. You may want to change the sort order based on external criteria, and resort the list in a **ClickColumn** event. The list will first be sorted based on the primary **SortKey** property, then the list is sub-sorted based on the **SortKey2** property. Finally, the list is sub-sorted again base on the value in this property.

As an example, suppose your list items have as their **Text** property, the last name of a person, the first SubItem contains the person's first name, and the second SubItem has the person's city name. If the **SortKey** property contains **0**, and the **SortOrder** property contains **gtSortAscending**, the list of names would first be sorted in ascending order based on the last name. The first names might come out in any random order. If **SortKey2** is set to the first SubItem (the first name), and **SortOrder2** is set to **gtSortAscending**, the list is sub-sorted based on the first name. Duplicate last names are sub-sorted based on the first name. The city names might come out in any random order. When the list is sub-sorted again based on **SortKey3** (the city name), and **SortOrder3** is set to **gtSortDescending**, duplicate names, are sorted based on the city and displayed in descending order as shown below:

SortKey / SortOrder

```
Smith   | Jane  |  Chicago
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
```

SortKey2 / SortOrder2

```
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
```

```
Smith   | Jane  | Chicago
```

SortKey3/ SortOrder3

```
Smith   | Ann   | Los Angeles
Smith   | Jane  | Chicago
Smith   | Jane  | Atlanta
```

See Also

Sorted, SortKey, SortKey2 properties**, SortOrder, SortOrder2, SortOrder3** properties, **ClickColumn** event

Example

The following code sets the **SortKey3** and **SortOrder** based on the external criteria from a **GTList** control**,** for a **GTCombo** control:

```
Private Sub GTCombo1_ClickColumn (ByVal ColumnHeader As ColumnHeader)
      If Check1.value = CHECKED Then
            GTCombo1.SortKey3= GTList2.ListIndex
            GTCombo1.SortOrder = GTList2.SortOrder
      End If
End Sub
```

SortOrder Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a value that determines whether **ListItem** objects in a **GTList** or **GTCombo** control are sorted using the primary **SortKey** in ascending or descending order.

Usage

object.**SortOrder** [= *value*]

The **SortOrder** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	An integer specifying the sort order as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
gtSortAscending	0	(Default) Ascending order. Sorts from the beginning of the alphabet (A-Z), the earliest date, or the lowest number.
gtSortDescending	1	Descending order. Sorts from the end of the alphabet (Z-A), the latest date, or the highest number..

Remarks

The **Sorted** property must be set to **True** before the list is reordered. This property applies to the sort order of the primary sort key in the property **SortKey**.

See Also

Sorted property**, SortKey** property

Example

The following code sets the **SortOrder** to a selection from a user's ListBox control:

```
GTList1.SortOrder = Combo1.ListIndex
GTList1.Sorted = True  ' Sort the List.
```

SortOrder2 Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a value that determines the order in which items are displayed after being sorted by the secondary sort key **SortKey2**.

Usage

object.**SortOrder2** [= *value*]

The **SortOrder2** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	An integer specifying the secondary sort order as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
gtSortAscending	0	(Default) Ascending order. Sorts from the beginning of the alphabet (A-Z), the earliest date, or the lowest number.
gtSortDescending	1	Descending order. Sorts from the end of the alphabet (Z-A), the latest date, or the highest number..

Remarks

The **Sorted** property must be set to **True** before the list is reordered. This property applies to the sort order of the secondary sort key in the property **SortKey2**. When the list is sorted on its primary key, duplicates will be displayed in any random order. A secondary sort can be performed using **SortKey2**. The output order of the secondary sort is determined by the value in this property.

See Also

Sorted, SortKey, SortKey2, SortKey3, SortOrder, SortOrder3 properties.

Example

The following code sets the **SortOrder2** to a selection from a user's ListBox control:

```
GTList1.SortOrder2 = Combo1.ListIndex
GTList1.Sorted = True  ' Sort the List.
```

SortOrder3 Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a value that determines the order in which items are displayed after being sub-sorted by the third sorting key **SortKey3**.

Usage

object.**SortOrder3** [= *value*]

The **SortOrder3** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	An integer specifying the sorted output order as described in Settings.

Settings

The settings for *value* are:

Constant	Value	Description
gtSortAscending	0	(Default) Ascending order. Sorts from the beginning of the alphabet (A-Z), the earliest date, or the lowest number.
gtSortDescending	1	Descending order. Sorts from the end of the alphabet (Z-A), the latest date, or the highest number..

Remarks

The **Sorted** property must be set to **True** before the list is reordered. The list is first sorted using the primary key in SortKey, and ordered using SortOrder. Then the list sub-sorts the duplicate records based on SortKey2 using the SortOrder2 to order the output. Finally, this property applies to the sort order of the third sort in the property **SortKey3**. When the list is sorted on its primary key, duplicates will be displayed in any random order. A secondary sort can be performed using **SortKey2**. The output order of the secondary sort is determined by the value in SortOrder2. Any remaining SubItems may be displayed in any random order. When the final third sort is performed using **SortKey3**, this property determines the order of the output displayed.

As an example, suppose your list items have as their **Text** property, the last name of a person, the first SubItem contains the person's first name, and the second SubItem has the person's city name. If the **SortKey** property contains **0**, and the **SortOrder** property contains **gtSortAscending**, the list of names would first be sorted in ascending order based on the last name. The first names might come out in any random order. If **SortKey2** is set to the first SubItem (the first name), and **SortOrder2** is set to **gtSortAscending**, the list is sub-sorted based on the first name. Duplicate last names are sub-sorted based on the first name. The city names might come out in any random order. When the list is sub-sorted again based on **SortKey3** (the city name), and **SortOrder3** is set to **gtSortDescending**, duplicate names, are sorted based on the city and displayed in descending order as shown below:

SortKey / SortOrder

```
Smith   | Jane  |  Chicago
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
```

SortKey2 / SortOrder2

```
Smith   | Ann   |  Los Angeles
Smith   | Jane  |  Atlanta
Smith   | Jane  | Chicago
```

SortKey3/ SortOrder3

```
Smith   | Ann   | Los Angeles
Smith   | Jane  | Chicago
Smith   | Jane  | Atlanta
```

See Also

Sorted, SortKey, SortKey2, SortKey3, SortOrder, SortOrder3 properties.

Example

The following code sets the **SortOrder3** to a selection from a user's ListBox control:

```
GTList1.SortOrder3 = Combo1.ListIndex
GTList1.Sorted = True  ' Sort the List.
```

Sorted Property

Applies To

GTList control, **GTCombo** control

Description

Returns or sets a value that determines whether the list items are sorted according to the **SortOrder** and **SortKey** properties.

Usage

object.**Sorted** [= *boolean*]

The **Sorted** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression determining if the items are sorted.

Remarks

Setting this property to **True** sorts the items according to the settings in **SortOrder** and **SortKey**. This property only effects the control when the control is non-bound and is non-virtual. Specifically, the control can not be bound to a database via the **DataSource** property or have virtual items via the **Virtual** property.

See Also

SortKey property**, SortOrder** property

Example

The following code sets the Sorted property to the value of a user's checkbox.

```
List1.Sorted = Check1.Value
```

SubColumn Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the index in the sub-columns the columndef object represents.

Usage

object. **SubColumn** [= *index*]

The **SubColumn** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
index	An index specifying the columndef's subcolumn index.

Remarks

This property specifies what sub-column index the columndef object represents when the columns are stacked one upon another. Typically the sub-column number will be 0 because the column header defines just one column. However, if a column is dragged and dropped into another column, the columns become stacked one upon another. When the columns are stacked, the subcolumn number will change to reflect the index of the column. Stacking columns is useful to provide a more compact view of the data. You may want to change the backcolor of the columndef object for each sub-column, and for the listitem and subitems they represent.

See Also

BackColor (ColumnDef, ListItem, SubItem objects)

Example

The following code gets the ColumnDef object's **SubColumn** property and changes the backcolor to a new value based on the index:

```
GTList1.ColumnDefs(2).BackColor =
      lookupBackColor(GTList1.ColumnDefs(2).SubColumn)
```

SubItemSource Property (ColumnDef object)

Applies To

ColumnDef object

Description

Specifies the index of the SubItem object used in the column.

Usage

object. **SubItemSource** [= *index*]

The **SubItemSource** property has these parts:

Part	Description
object	An object expression that evaluates to a ColumnDef object.
index	An index specifying the SubItem displayed in the column.

Remarks

This property defines which SubItem object's data is displayed in the column when displayed in the list. Usually the ColumnDefs collection indices match the SubItem indices when the data is displayed, however, the **SubItemSource** property can change the default activity.

See Also

SubItems collection

Example

The following code sets the ColumnDef object's **SubItemSource** property to **15** for a **GTList** control

```
GTList1.ColumnDefs(2).SubItemSource = 15
```

SubItems Collection, SubItem Object

See Also Properties Methods Constants

The SubItems collection is a standard collection object that holds a series of SubItem objects. A SubItem object contains an additional text string or picture to be associated with a ListItem object. The collection supports all of the standard methods associated with collections. Select one of the items above to see the Properties, Methods, and Constants associated with the SubItems collection.

The SubItem object defines each SubItem as it appears in the GTCombo or GTList window. Each item has support for:

- Individual BackColor and ForeColor per item.
- Individual font and styles
- Alignment of text and images

Usage

listitemobject.SubItems(*index*)

The syntax lines above refer to the collection and to individual elements in the collection, respectively, according to the standard collection syntax.

Remarks

SubItem objects can contain both text and pictures. However, to use the pictures the parent control, must have an **ImageList** associated with it. If the control's **ImageList** property is set to the string '`Internal`' the control's internal ImageList can be used. The internal ImageList can be filled with images during design time using the control's property page.

Listed below are all of the properties and methods supported by the collection and object. Any property or method that is underlined, is standard and not documented here. See the standard Microsoft documentation for information on those elements. Items marked with the symbol () apply only to the collection. All other properties apply only to the object.

Properties

BackColor	BackStyle	Count
Font	Font3D	ForeColor
Format	Image	Index
Key	MemoryUsed	PictureAlignment
SelectedImage	Text	TextAlignment
Width	WordWrap	

Methods

Add	Clear	Item
Remove	SetDefaults	

SubItems Collection, SubItem Object Methods

Add (Collection only)
Clear (Collection only)
Item (Collection only)
Remove (Collection only)

SetDefaults (Object only)

SubItems Collection, SubItem Object Properties

Underlined items are stock properties

Count (Collection Only)
MemoryUsed (Collection Only)

BackColor
BackStyle
Font
Font3D
ForeColor
Format
Image
Index
Key
PictureAlignment
SelectedImage
Text
TextAlignment
Width
WordWrap

SubItems Property (ListItem object)

Applies To

ListItem object

Description

Returns a reference to a collection of **SubItem** objects.

Usage

object.**SubItems**

The object placeholder represents an object expression that evaluates to one of the objects listed in the Applies To section.

Remarks

You can manipulate **SubItem** objects using standard collection methods (for example, the **Remove** method). Each **SubItem** in the collection can be accessed either by its index or by a unique key, stored in the **Key** property.

See Also

SubItems collection, **SubItem** object, **Key** property.

Example

The following example removes the first SubItem object from the collection:

```
GTCombo1.ListItems.Item(0).SubItems.Remove 1
```

SubItems: See Also

GTCombo control reference
GTList control reference

ColumnDefs collection, ColumnDef object
ListImages collection, ListImage object
ListItems collection ListItem object

SubRows Property

Applies To

GTList control **GTCombo** control

Description

Returns or sets a value that determines how many lines are used per ListItem row to display the ListItem's data.

Usage

object.**SubRows** [= *value*]

The **SubRows** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
value	An integer specifying the number of lines per ListItem row as described in Settings.

Settings

The settings for *value* are:

Value	Description
> 0	An integer expression indicating the number of rows used to display each ListItem's data.

Remarks

Use this property to specify the number of lines used to display ListItem data and SubItem data. This property extends the number of lines in the row. Setting this property to 1 will cause the control to display the data on just the main row.

See Also

SubRowsStatic property.

Example

The following code sets the **SubRows** to a selection from a user's ListBox control:

```
GTList1.SubRows = Combo1.ListIndex + 1
```

SubRows and SubRowsStatic

The **SubRows** and **SubRowsStatic** properties control how many lines per list item are used to display the row of data in the control window. See GTList's Exercise Two (Customizing the look and feel) for an example of how **SubRows** and **SubRowsStatic** can add flexibility to your application.

SubRowsStatic Property

Applies To

GTList control, **GTCombo** control

Description

Determines if new sub-rows can be created by dragging and dropping column headers below the existing sub-rows.

Usage

*object.***SubRowsStatic** [= *boolean*]

The **SubRowsStatic** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression determining if new sub-rows are automatically created by dragging and dropping column headers below an existing sub-row. Default = True (no creation of new sub-rows)

Remarks

Setting this property to **True** only allows column headers to be dragged and dropped to existing sub-rows. Setting this property to **False** creates new sub-rows when a column header is dragged and dropped below an existing sub-row.

See Also

SubRows property, **ColumnDef** object.

Example

The following code enables or disables the **SubRowsStatic** feature based on a user's checkbox value.

```
List1.SubRowsStatic = Check1.value
```

TagVariant Property

Applies To

GTList control **GTCombo** control, **ListItem** object, **ColumnDef** Object

Description

Returns or sets any user-defined value.

Usage

object.**TagVariant** [= *value*]
The **TagVariant** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls or objects in the Applies To list.
value	Any variant data type. The Variant data type is a special data type that can contain any kind of data except fixed-length String data and user-defined types.

Remarks

Use the **TagVariant** property to store specialized information about the control or object

See Also

None

Example

The following code sets the **TagVariant** of a ListItem object:

```
GTList1.ListItems(0).TagVariant = "File or Directory"
```

Technical Specifications

Your system must adhere to the following minimum requirements to fully utilize DataList ActiveX controls:

Microsoft Visual Basic version 4.x or a development tool that supports ActiveX controls. (.OCX files)
At least 5 megabytes of available hard disk space (full installation).
For the 32-bit version of DataList ActiveX, you must be running either Windows 95 or Windows NT 3.51 or later.
For the 16-bit version of DataList ActiveX, you must be running either Windows version 3.1 or later.

Technical Support

Internet Email

If you have access to Internet email, feel free to submit your questions and feedback to our technical support staff via electronic mail at **support@green-tree.com**. You can also visit our World Wide Web site at **www.green-tree.com** for information on future releases of DataList, Sample Programs, and Frequently Asked Questions (FAQs).

Fax

To fax questions or feedback regarding any GreenTree product, fax to **(516) 271-8067.**

Telephone Support

For free technical support for DataList ActiveX or any other GreenTree product, contact GreenTree Technologies, Inc. at **(516) 271-6995**. GreenTree's support hours are **9AM to 5PM (EST)**, Monday through Friday.

TextAlignment Property (ListItem, SubItem object)

Applies To

ListItem object, **SubItem** object

Description

Specifies the text alignment used for the object's text.

Usage

object. **TextAlignment** [= *number*]

The **TextAlignment** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To list.
number	A number or value specifying the alignment for the object's text as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtTextAlignmentLeftTop	0	Left Top. Caption text is left/top aligned.
gtTextAlignmentLeftMiddle	1	Left Middle. Caption text is left/middle aligned.
gtTextAlignmentLeftBottom	2	Left Bottom. Caption text is left/bottom aligned.
gtTextAlignmentRightTop	3	Right Top. Caption text is right/top aligned.
gtTextAlignmentRightMiddle	4	Right Middle. Caption text is right/middle aligned.
gtTextAlignmentRightBottom	5	Right Bottom. Caption text is right/bottom aligned.
gtTextAlignmentCenterTop	6	Center Top. Caption text is center/top aligned.
gtTextAlignmentCenterMiddle	7	Center Middle. Caption text is center/middle aligned.
gtTextAlignmentCenterBottom	8	Center Bottom. Caption text is center/bottom aligned.

Remarks

Use this property to get or set the text alignment for the object.

See Also

ListItems collection, **SubItems** collection, **DefTextAlignment** property

Example

The following code sets the alignment to the CenterTop for a **ListItem** object contained in a **GTCombo** control:

```
Dim xListItem As ListItem
xListItem = GTCombo1.ListItems.Item(0)
xListItem. TextAlignment = gtTextAlignmentCenterTop
```

TopIndex Property

Applies To

GTList control**, GTCombo** control

Description

Returns or sets the index of the first visible item in the control's window.

Usage

object.**TopIndex** [= *number*]

The **TopIndex** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
number	A long integer specifying the zero based index of the item to be displayed as the top item in the control's window.

Remarks

You can use this property to determine or set which item in the list is the first visible list item in the control's window. You might use it in conjunction with the NewIndex property to assure all newly added items to the list are displayed as the top item in the control's window.

The **TopIndex** property returns -1 if there are no items in the list.

See Also

NewIndex property, **ListItems** collection

Example

This example gets the text for the first visible list item, and assigns the text to a label's caption property.

```
Label1.Caption = GTList1.ListItems(GTList1.TopIndex).Text
```

VertScroll Event

Applies To

GTList control, **GTCombo** control

Description

Occurs when the list portion of the control is scrolled in the vertical direction.

Usage

Private Sub *object*_**VertScroll()**

The *object* expression evaluates to one of the controls in the Applies To list.

Remarks

This event is fired each time thelist is scrolled in the vertical direction. You may want to check the **LeftColumn** property to determine which ListItem is at the top of the list inside this event handler.

See Also

VertScrollBar, **LeftColumn** propterties, **HorzScroll** event

Example

This example displays the left column number when the list is scrolled for a **GTList** control:

```
Private Sub GTList1_VertScroll ()
      Label1.Caption = "Column: " + STR$(GTList1.LeftColumn)
End Sub
```

VertScrollBar Property

Applies To

GTList control **GTCombo** control

Description

Gets or Sets the vertical scrolling mode and the appearance of scroll bars for the control's window.

Usage

object.**VertScrollBar** [= *mode*]

The **VertScrollBar** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
mode	An integer specifying the vertical scrolling bar setting for the control.

Settings

The settings for *number* are:

Setting	Value	Description
gtSBNever	0	Never. No vertical scroll bar is displayed in the control's window.
gtSBAlways	1	Always. A vertical scroll bar is always present on the bottom of the control's window.
gtSBNeeded	2	(Default) Needed. A vertical scroll bar will be added to bottom of the control's window automatically when the width of the ListItems and its associated SubItems exceeds the width that can be displayed in the window or the width of the ColumnDefs exceeds the width of the control's window.

Remarks

This property controls the appearance of the control's vertical scroll bars, and the attributes of the scroll bar's thumb.

See Also

HorzScrollBar property

Example

The following code sets the **VertScrollBar** to a value set by a user's ListBox control:

```
GTList1.VertScrollBar = ListBox1.ListIndex
```

Virtual Property

Applies To

GTList control, **GTCombo** control

Description

Determines if the control operates in Virtual or non-Virtual mode.

Usage

object.**Virtual** [= *boolean*]

The **Virtual** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
boolean	A boolean expression determining if the control operates in Virtual or non-virtual mode as described below.

Remarks

If the control is in Virtual mode, the application is expected to supply data to the control when the ListItemDataRequest event occurs. The application first set the VirtualItemCount property to determine the number of virtual items in the list, then fills in the list item data when the ListItemDataRequest event is fired.

Note: Virtual mode is automatically disabled when the control is bound to an external database, or listitems are added to the control using the **AddItem** method.

See Also

VirtualItemCount, DataSourceList, DataSource properties**, AddItem** method.

Example

The following code gets the value of the current virtual mode and displays it in a user's checkbox:

```
Check1.Value = List1.Virtual
```

Virtual and VirtualItemCount

Virtual and **VirtualItemCount** are powerful properties that allow the application to control the data displayed by the control. Instead of gathering the data from a database as is the case with DataSourceList, the control notifies the application when data is needed, and the application supplies the data to the control. **Virtual** indicates to the control that the application will be supplying the data to the control when the control requests the data. To use the virtual list feature **Virtual** must be **True**, and the control can not be bound to a data source. The application controls how many items can be in the virtual list by specifying the count in the **VirtualItemCount** property. When the control needs data a **ListItemDataRequest** event is fired. Inside the event handler, the application fills in the data requested by the control. That's all there is to it. GreenTree makes virtual data lists simple.

Of course, you can always use the **GTCombo** control without binding it to a data source, or supplying virtual list items. By calling the control's **AddItem** method, a new list item can be added to the combo box list. Removing an item is just as easy, and clearing the entire contents of the list is a simple one line method call.

VirtualItemCount Property

Applies To

GTList control **GTCombo** control

Description

Gets or Sets the number of items in the virtual list when the control is operating in Virtual mode.

Usage

object.**VirtualItemCount** [= *count*]

The **VirtualItemCount** property has these parts:

Part	Description
object	An object expression that evaluates to one of the controls in the Applies To list.
count	An integer specifying the number of virtual items in the control's list.

Settings

The settings for *number* are:

Setting	Description
>= 0	Sets the number of virtual items in the control's list.

Remarks

Use this property to specify how many items are in the virtual list. This property is only valid when the Virtual property is True. The value in this property controls the minimum and maximum values for the control's vertical scroll bar, and the delta for each of the scrollbar's thumb positions. An index is provided in the ListItemDataRequest event that indicates which list item needs to be loaded with application supplied data. The indices supplied to the application will be in the range of 0 to the value of this property minus 1. Setting this property to a value of 10,000 would yield indices of 0 to 9999.

See Also

Virtual property, **ListItemDataRequest** event.

Example

The following code sets the **VirtualItemCount** to 10,000 items.

```
GTList1.VirtualItemCount = 10000
```

What is DataList ActiveX?

See Also

DataList ActiveX is a set of ActiveX custom controls that provide extended ComboBox and extended ListBox functionality to your application. DataList consists of two controls, GTList and GTCombo.

GTList ActiveX Control

The **GTList** Control is a greatly enhanced version of the standard ListBox control. The **GTList** control has all of the functionality commonly associated with a ListBox control plus support for many powerful features that add incremental capabilities to your application. While **GTList** is an extremely powerful tool, it is also very simple to incorporate and use, maximizing your productivity. **GTList** takes care of all of the details associated with the list, and managing data in the list, while you can concentrate on the specifics of your application. Of course, as you might expect from GreenTree Technologies, **GTList** is a data bound control, so access to your data is quick and easy.

Listed below are just some of the features of **GTList:**

- Automatic column creation when bound to a data source.
- Alternating row colors for even and odd items.
- Images associated with each list item.
- SubItems associated with each item.
- Images attached to each SubItem
- Data binding on the main control list items
- Column headers associated with the list item and it's SubItems
- Virtual lists where the application supplies items at runtime.
- Images attached to each column header
- Dynamic loading of images or text for SubItems
- ScrollTips associated with the controls scroll bars.
- ExtendTips that appear over each list item and its SubItems.
- Event notification when items or column headers are clicked.
- Event notification when ExtendTips and ScrollTips begin and end.
- Notification when users begin and end moving columns.
- Grid lines to visually separate rows and columns.
- Multiple lines of data per list row.
- Elastic column sizing that maintains the overall width of the control.
- Word wrapping on column headers, list items, and SubItem text.
- Ability to re-size and move columns.

GTCombo ActiveX Control

The **GTCombo** Control is a greatly enhanced version of the standard ComboBox control. The **GTCombo** control has all of the functionality commonly associated with a ComboBox control plus support for many powerful features that add incremental capabilities to your application. While **GTCombo** is an extremely powerful tool, it is also very simple to incorporate and use, maximizing your productivity. **GTCombo** takes care of all of the details associated with the combo box list, and managing data in the list, while you can concentrate on the specifics of your application. Of course, as you might expect from *GreenTree Technologies*, **GTCombo** is a data bound control, so access to all your data is right at your fingertips.

Listed below are just some of the features of **GTCombo:**

- Images associated with each list item.
- SubItems associated with each item.
- Images attached to each SubItem
- Separate data binding on the main control edit box and on the control's list items

- Column headers associated with the list item and it's SubItems
- Images attached to each column header
- ScrollTip items associated with the controls scroll bars.
- ExtendTips can appear over items in the combo box.
- Event notification when items or column headers are clicked.
- Ability to re-size, move, and create columns on the fly.
- Virtual lists of items can be loaded into the control at runtime by the application.
- Automatic column creation when bound to a data source.
- Alternating row colors for even and odd items.
- ScrollTips associated with the controls scroll bars.
- Event notification when ExtendTips and ScrollTips begin and end.
- Notification when users begin and end moving columns.
- Grid lines to visually separate rows and columns.
- Multiple lines of data per list row.
- Elastic column sizing that maintains the overall width of the control.
- Word wrapping on column headers, list items, and SubItem text.

WordWrap Property (ColumnDef object, ListItem object, SubItem object)

Applies To

ColumnDef object, **ListItem** object, **SubItem** object

Description

Specifies if the object's data will word wrap based on the width of the column header.

Usage

object. **WordWrap** [= *number*]

The **WordWrap** property has these parts:

Part	Description
object	An object expression that evaluates to one of the objects in the Applies To List.
number	A number or value specifying the word wrap setting as specified in Settings.

Settings

The settings for *number* are:

Setting	Value	Description
gtWordWrapOff	0	Off. No word wrapping take place.
gtWordWrapOn	1	On. Word wrapping is enabled.
gtWordWrapUseDefault	2	Use Default. Use the parent control's **DefWordWrap** property value.

Remarks

Use this property to get or set the word wrap setting for the object's data text. Setting this property to **gtWordWrapUseDefault** uses the value in the parent control's **DefWordWrap** property. Turning Word Wrap mode on causes all of the SubItems displayed for the column to use word wrapping.

See Also

DefWordWrap property

Example

The following code sets the list item object's **WordWrap** property to **True** for a **GTList** control

```
GTList1.ListItems(0).WordWrap = gtWordWrapOn
```

boolean expression

An Expression that evaluates to either **True** or **False**.

constant

A named item that retains a constant value throughout the execution of a program, as opposed to a variable whose value can change during execution. Each application can define its own set of constants. Additional constants may be defined by the user using the **Const** statement. Constants can be used anywhere in your code in place of actual values. A constant may be a string, literal number, or other constant, or any combination that contains arithmetic or logical operators except **Is**.and exponentiation.

container
An object that holds child forms or controls.

named arguments
An argument that has a name that is predefined in the object library. Instead of providing values for arguments in the order expected by the syntax, you can use named arguments to assign values in any order.

numeric expression
Any expression that can be evaluated as a number. Elements of the expression can include any combination of keywords, variables, constants, and operators that result in a number.

object expression
An expression that specifies a particular object. This expression can include any of the object's containers.

string expression
Any expression that results in a sequence of contiguous characters. Elements of an expression can include a function that returns a string, a string literal, a string constant, a string variable, a string **Variant,** or a function that returns a string **Variant.**