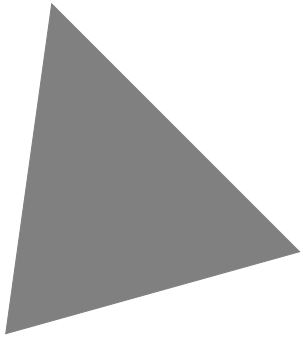
Visual Component Library Reference

Addendum



Borland
C++Builder™
for Windows 95 and Windows NT

Borland International, Inc., 100 Borland Way
P.O. Box 660001, Scotts Valley, CA 95067-0001

Contents

TBookmarkList

Header: dbgrids.hpp

TObject • TBookmarkList

TBookmarkList is a collection of bookmarks that identify particular records in a dataset.

Description Use *TBookmarkList* to manage a set of bookmarks when working with the records in a dataset. Each bookmark identifies a particular record in the dataset, and is identified by a string.

See also *TDBGrid::SelectedRows* property

Properties

The following table lists the properties of *TBookmarkList:*

Property	Visibility	Default	Documented in
Count	public, read-only		TBookmarkList
CurrentRowSelected	public		TBookmarkList
Items	public, read-only		TBookmarkList

Count

Returns the number of bookmarks in the bookmark list.

```
__property int Count;
```

Description Use *Count* to return the number of bookmarks in the bookmark list.

Count determines the number of bookmarks by calling the *Count* method of a private *TStringList* object containing the bookmark names.

See also *TStringList::Count* method

CurrentRowSelected

Specifies whether the grid's current row is selected.

```
__property bool CurrentRowSelected;
```

Description Use *CurrentRowSelected* to specify whether to the grid's current row is selected.

See also *Items* property

Items

Contains the names of the bookmarks in the bookmark list.

```
__property System::AnsiString Items[int Index];
```

Description Use *Items* to return the name of a bookmark in the bookmark list.

Index is the index of the bookmark name to return.

See also *Clear* property, *Refresh* property

Methods

The following table lists the methods of *TBookmarkList:*

Method	Visibility	Documented in
~TBookmarkList	public	TBookmarkList
ClassInfo	public	TObject
ClassName	public	TObject
ClassNameIs	public	TObject
ClassParent	public	TObject
ClassType	public	TObject
CleanupInstance	public	TObject
Clear	public	TBookmarkList
DefaultHandler	public	TObject
Dispatch	public	TObject
FieldAddress	public	TObject
Find	public	TBookmarkList
Free	public	TObject
FreeInstance	public	TObject
IndexOf	public	TBookmarkList
InheritsFrom	public	TObject
InitInstance	public	TObject
InstanceSize	public	TObject
MethodAddress	public	TObject
MethodName	public	TObject
NewInstance	public	TObject
Refresh	public	TBookmarkList
TBookmarkList	public	TBookmarkList

~TBookmarkList

~TBookmarkList frees the memory associated with the *TBookmarkList* object. Do not call *~TBookmarkList* directly. Instead, use the **delete** keyword on the object, which causes *~TBookmarkList* to be invoked automatically.

```
__fastcall virtual ~TBookmarkList(void);
```

Description *TBookmarkList*

- Calls the *Clear* method.
- Frees the private *TStringList* object containing the bookmark names.
- Calls the *TObject::~TObject* method.

See also *Clear* method, *TBookmarkList* method, *TObject::TObject* method

Clear

Clears the bookmark list.

```
void __fastcall Clear(void);
```

Description Use *Clear* to clear the bookmark list. *Clear:*

- Calls the *Clear* method of a private *TStringList* object containing the bookmark names.
- Invalidates the grid.

See also *Items* property, *TStringList::Clear* method, *TWinControl::Invalidate* method

Find

Finds a bookmark in the bookmark list.

```
bool __fastcall Find(const System::AnsiString Item, int &Index);
```

Description Use *Find* to retrieve the index of a bookmark in the bookmark list.

Item is the name of the bookmark to find.

Index is where to store the index of *Item*.

Find returns **true** if *Item* is found in a private *TStringList* object containing the bookmark names. Otherwise *Find* returns **false**.

See also *IndexOf* method

IndexOf

Returns the index of a bookmark in the bookmark list.

```
int __fastcall IndexOf(const System::AnsiString Item);
```

Description Use *IndexOf* to return the index of a bookmark in the bookmark list.

Item is the name of the bookmark.

IndexOf returns -1 if *Item* is not found in the bookmark list.

IndexOf calls the *Find* method to determine whether *Item* exists.

See also *Find* method

Refresh

Refreshes the bookmark list with the current grid data.

```
bool __fastcall Refresh(void);
```

Description Use *Refresh* to refresh the bookmark list with the current grid data.

Refresh returns **true** if successful; otherwise *Refresh* returns **false**.

If successful, *Refresh* invalidates the grid.

See also *TWinControl::Invalidate* method

TBookmarkList

Creates and initializes a new *TBookmarkList* object.

```
__fastcall TBookmarkList(TCustomDBGrid* AGrid);
```

Description Use *TBookmarkList* to create and initialize a new *TBookmarkList* object.

AGrid is the grid to associate with the *TBookmarkList* object.

TBookmarkList

- Calls the *TObject*::*TObject* method.
- Initializes a private pointer to *AGrid*.
- Creates a private *TStringList* object.
- Assigns an internal event handler to the *OnChange* event for the *TStringList* object.

See also *~TBookmarkList* method, *TObject::TObject* method

TCustomHotKey

Header: comctrls.hpp

TObject • TPersistent • TComponent • TControl • TWinControl • TCustomHotKey

TCustomHotKey is the abstract base class type for the hot key object that is a wrapper for the Windows hot key Common Control.

Description

TCustomHotKey is an abstract class that encapsulates behavior of an object that is used to set a shortcut property at runtime. Hot key objects are used to perform an action quickly by enabling users to link the hot key to a combination of keystrokes that will execute that action. TCustomHotKey introduces properties to:

- Associate the hot key with keystrokes to perform an action
- Select a key combination at design-time
- Invalidate specific modifier keys

To change the key combination at run-time hold down the *Alt*, *Ctrl* or *Shift* keys or combinations of them and type a letter or number. The hot key property can then be assigned to the shortcut property of a menu item, for example.

Hot key classes that descend from *TCustomHotKey* inherit the basic functionality of *TCustomHotKey*. For component writers wanting to create a customized hot key objects, use *TCustomHotKey* as a base class for deriving specialized hot key objects.

Properties

The following table lists the properties of *TCustomHotKey*:

Property	Visibility	Default	Documented in
Align	public	alNone	TControl
AutoSize	protected	true	TCustomHotKey
BoundsRect	public		TControl
Brush	public, read-only		TWinControl
ClientHeight	public		TControl
ClientOrigin	public, read-only		TWinControl
ClientRect	public, read-only		TWinControl
ClientWidth	public		TControl
ComponentCount	public, read-only		TComponent
ComponentIndex	public		TComponent
Components	public, read-only		TComponent
ComponentState	public, read-only		TComponent
ComponentStyle	public, read-only		TComponent
ControlCount	public, read-only		TWinControl

Property	Visibility	Default	Documented in
Controls	public, read-only		TWinControl
ControlState	public		TControl
ControlStyle	public		TControl
Cursor	published	crDefault	TControl
DesignInfo	public		TComponent
Enabled	public	true	TControl
Handle	public, read-only		TWinControl
Height	published		TControl
HelpContext	published	0	TWinControl
Hint	published		TControl
HotKey	protected		TCustomHotKey
InvalidKeys	protected		TCustomHotKey
Left	published		TControl
Modifiers	protected		TCustomHotKey
Name	published		TControl
Owner	public, read-only		TComponent
Parent	public		TControl
ShowHint	public		TControl
Showing	public, read-only		TWinControl
TabOrder	public	-1	TWinControl
TabStop	protected	true	TWinControl
Tag	published	0	TComponent
Top	published		TControl
Visible	public	true	TControl
Width	published		TControl

AutoSize

AutoSize determines whether the hot key automatically changes its height to accomodate the font size.

```
__property bool AutoSize;
```

Description Use *AutoSize* to fully display the font on the hot key control. When *AutoSize* is *true*, the height of the hot key changes to display the key combinations typed in it. When *AutoSize* is *false*, the hot key remains the same height, regardless of any font changes. The default value is *true*.

HotKey

HotKey contains the current key combination of the hot key control.

```
__property Menus::TShortCut HotKey;
```

Description Use *HotKey* to set or change the key combination associated with the hot key. *Alt+A* is the default. To modify the value of *HotKey* either edit it directly or set the value of the *Modifiers* property. At run-time *HotKey* is reset when the hot key has focus and the user presses valid key combinations.

See also *Modifiers* property

InvalidKeys

InvalidKeys allows the user to specify that one or more modifier keys should be considered invalid.

```
enum THKInvalidKey { hcNone, hcShift, hcCtrl, hcAlt, hcShiftCtrl, hcShiftAlt, hcCtrlAlt,
hcShiftCtrlAlt };

typedef Set<THKInvalidKey, hcNone, hcShiftCtrlAlt>  THKInvalidKeys;

__property THKInvalidKeys InvalidKeys;
```

Description Use *InvalidKeys* to select which modifier keys are not allowed. If this property is set, a default modifier should also be specified in the *Modifiers* property. When the invalid key is selected, the default modifier is displayed in its place.

THKInvalidKey is the type of the *InvalidKeys* property. These are the possible values:

Value	Meaning
hcNone	Unmodified keys are invalid.
hcShift	The *Shift* key is invalid as a modifier.
hcCtrl	The *Ctrl* key is invalid as a modifier.
hcAlt	The *Alt* key is invalid as a modifier.
hcShiftCtrl	The *Shift+Ctrl* key combination is invalid as a modifier.
hcShiftAlt	The *Shift+Alt* key combinations invalid as a modifier.
hcCtrlAlt	The *Ctrl+Alt* key combination is invalid as a modifier.
hcShiftCtrlAlt	The *Shift+Ctrl+Alt* key combination is invalid as a modifier.

The default values are *hcNone* and *hcShift*.

See also *Modifiers* property

Modifiers

Modifiers specifies the key or keys that are used in combination with a non-modifier key.

```
enum THKModifier { hkShift, hkCtrl, hkAlt, hkExt };

typedef Set<THKModifier, hkShift, hkExt>  THKModifiers;

__property THKModifiers Modifiers;
```

Description Use *Modifiers* to select a modifier key such as *Ctrl*, *Alt*, or *Shift* for the hot key. These are the possible values:

Value	Meaning
hkShift	The *Shift* key is used as a modifier.
hkCtrl	The *Ctrl* key is used as a modifier.
hkAlt	The *Alt* key is used as a modifier.
hkExt	The Extra key is used as a modifier.

The default value is *hkAlt*.

Modifier keys are used in combination with another non-modifier key such as character and function keys, arrow keys, and so on. More than one modifier key can be used.

Methods

The following table lists the methods of *TCustomHotKey*:

Method	Visibility	Documented in
~TCustomHotKey	public	TCustomHotKey
Assign	public	TPersistent
BeginDrag	public	TControl
BringToFront	public	TControl
Broadcast	public	TWinControl
CanFocus	public	TWinControl
ClassInfo	public	TObject
ClassName	public	TObject
ClassNameIs	public	TObject
ClassParent	public	TObject
ClassType	public	TObject
CleanupInstance	public	TObject
ClientToScreen	public	TControl
ContainsControl	public	TWinControl
ControlAtPos	public	TWinControl
DefaultHandler	public	TObject
DestroyComponents	public	TComponent
Destroying	public	TComponent
DisableAlign	public	TWinControl
Dispatch	public	TObject
DragDrop	public	TControl
Dragging	public	TControl
EnableAlign	public	TWinControl
EndDrag	public	TControl

Method	Visibility	Documented in
FieldAddress	public	TObject
FindComponent	public	TComponent
Focused	public	TWinControl
Free	public	TObject
FreeInstance	public	TObject
FreeNotification	public	TComponent
GetParentComponent	public	TComponent
GetTabOrderList	public	TWinControl
GetTextBuf	public	TControl
GetTextLen	public	TControl
HandleAllocated	public	TWinControl
HandleNeeded	public	TWinControl
HasParent	public	TComponent
Hide	public	TControl
InheritsFrom	public	TObject
InitInstance	public	TObject
InsertComponent	public	TComponent
InsertControl	public	TWinControl
InstanceSize	public	TObject
Invalidate	public	TWinControl
MethodAddress	public	TObject
MethodName	public	TObject
NewInstance	public	TObject
PaintTo	public	TWinControl
Perform	public	TControl
Realign	public	TWinControl
Refresh	public	TControl
RemoveComponent	public	TComponent
RemoveControl	public	TWinControl
Repaint	public	TWinControl
ScaleBy	public	TWinControl
ScreenToClient	public	TControl
ScrollBy	public	TWinControl
SendToBack	public	TControl
SetBounds	public	TWinControl
SetFocus	public	TWinControl
SetTextBuf	public	TControl
Show	public	TControl
Update	public	TWinControl
TCustomHotKey	public	TCustomHotKey
UpdateControlState	public	TWinControl

~TCustomHotKey

~TCustomHotKey frees the memory associated with the *TCustomHotKey* object. Do not call *~TCustomHotKey* directly. Instead, use the **delete** keyword on the object, which causes *~TCustomHotKey* to be invoked automatically.

```
__fastcall virtual ~TCustomHotKey(void);
```

See also *TCustomHotKey::TCustomHotKey* method

TCustomHotKey

TCustomHotKey instantiates a hot key control.

```
__fastcall virtual TCustomHotKey(Classes::TComponent* AOwner);
```

Description Call *TCustomHotKey* to instantiate a hot key control at runtime. For hot key controls created at design time, *TCustomHotKey* is called automatically.

TCustomHotKey allocates memory for a hot key control and calls the constructor of its base class. Then it sets default values for the *Width* and *Height* properties, and initializes the following properties:

- *TabStop* to true
- *ParentColor* to false
- *AutoSize* to true
- *InvalidKeys* to hcNone and hcShift
- *Modifiers* to [hkAlt]
- *HotKey* to 'Alt+A'

See also *TControl::ParentColor* method, *TCustomHotKey::~TCustomHotKey* method, *TWinControl::TWinControl* method

TEdit

Header: stdctrls.hpp

TObject • TPersistent • TComponent • TControl • TWinControl • TCustomEdit • TEdit

TEdit is a wrapper for a Windows single-line edit control.

Description Use a *TEdit* object to put a standard Windows edit control on a form. Edit controls are used to retrieve text that users type. Edit controls can also display text to the user.

When only displaying text to the user, choose an edit control to allow users to select text and copy it to the Clipboard. Choose a label object if the selection capabilities of an edit control are not needed.

TEdit implements the generic behavior introduced in *TCustomEdit*. *TEdit* publishes many of the properties inherited from *TCustomEdit*, but does not introduce any new behavior. For specialized edit controls, use other descendant classes of *TCustomEdit* or derive from it.

See also *TDBEdit* object, *TDBMemo* object, *TInplaceEdit* object, *TLabel* object, *TMaskEdit* object, *TMemo* object, *TRichEdit* object

Properties

The following table lists the properties of *TEdit*:

Property	Visibility	Default	Documented in
Align	public	alNone	TControl
AutoSelect	published	true	TCustomEdit
AutoSize	published	true	TCustomEdit
BorderStyle	published	bsSingle	TCustomEdit
BoundsRect	public		TControl
Brush	public, read-only		TWinControl
CharCase	published	ecNormal	TCustomEdit
ClientHeight	public		TControl
ClientOrigin	public, read-only		TWinControl
ClientRect	public, read-only		TWinControl
ClientWidth	public		TControl
Color	published	clWindow	TControl
ComponentCount	public, read-only		TComponent
ComponentIndex	public		TComponent
Components	public, read-only		TComponent
ComponentState	public, read-only		TComponent
ComponentStyle	public, read-only		TComponent
ControlCount	public, read-only		TWinControl

Property	Visibility	Default	Documented in
Controls	public, read-only		TWinControl
ControlState	public		TControl
ControlStyle	public		TControl
Ctl3D	published		TWinControl
Cursor	published	crDefault	TControl
DesignInfo	public		TComponent
DragCursor	published	crDrag	TControl
DragMode	published	dmManual	TControl
Enabled	published	true	TControl
Font	published		TControl
Handle	public, read-only		TWinControl
Height	published		TControl
HelpContext	published	0	TWinControl
HideSelection	published	true	TCustomEdit
Hint	published		TControl
Left	published		TControl
MaxLength	published	0	TCustomEdit
Modified	public		TCustomEdit
Name	published		TControl
OEMConvert	published	false	TCustomEdit
Owner	public, read-only		TComponent
Parent	public		TControl
ParentColor	published	false	TControl
ParentCtl3D	published	true	TWinControl
ParentFont	published	true	TControl
ParentShowHint	published	true	TControl
PasswordChar	published	#0	TCustomEdit
PopupMenu	published		TControl
ReadOnly	published, read-only		TCustomEdit
SelLength	public		TCustomEdit
SelStart	public		TCustomEdit
SelText	public		TCustomEdit
ShowHint	published		TControl
Showing	public, read-only		TWinControl
TabOrder	published	–1	TWinControl
TabStop	published	true	TWinControl
Tag	published	0	TComponent
Text	published		TControl
Top	published		TControl
Visible	published	true	TControl
Width	published		TControl

Events

The following table lists the events of *TEdit*:

Event	Visibility	Documented in
OnChange	published	TCustomEdit
OnClick	published	TControl
OnDblClick	published	TControl
OnDragDrop	published	TControl
OnDragOver	published	TControl
OnEndDrag	published	TControl
OnEnter	published	TWinControl
OnExit	published	TWinControl
OnKeyDown	published	TWinControl
OnKeyPress	published	TWinControl
OnKeyUp	published	TWinControl
OnMouseDown	published	TControl
OnMouseMove	published	TControl
OnMouseUp	published	TControl
OnStartDrag	published	TControl

OnChange

Occurs when the content of the edit control changes.

```
__property OnChange;
```

Methods

The following table lists the methods of *TEdit*:

Method	Visibility	Documented in
~TEdit	public	TEdit
Assign	public	TPersistent
BeginDrag	public	TControl
BringToFront	public	TControl
Broadcast	public	TWinControl
CanFocus	public	TWinControl
ClassInfo	public	TObject
ClassName	public	TObject
ClassNameIs	public	TObject
ClassParent	public	TObject
ClassType	public	TObject

Method	Visibility	Documented in
CleanupInstance	public	TObject
Clear	public	TCustomEdit
ClearSelection	public	TCustomEdit
ClientToScreen	public	TControl
ContainsControl	public	TWinControl
ControlAtPos	public	TWinControl
CopyToClipboard	public	TCustomEdit
CutToClipboard	public	TCustomEdit
DefaultHandler	public	TObject
DestroyComponents	public	TComponent
Destroying	public	TComponent
DisableAlign	public	TWinControl
Dispatch	public	TObject
DragDrop	public	TControl
Dragging	public	TControl
EnableAlign	public	TWinControl
EndDrag	public	TControl
FieldAddress	public	TObject
FindComponent	public	TComponent
Focused	public	TWinControl
Free	public	TObject
FreeInstance	public	TObject
FreeNotification	public	TComponent
GetParentComponent	public	TComponent
GetSelTextBuf	public	TCustomEdit
GetTabOrderList	public	TWinControl
GetTextBuf	public	TControl
GetTextLen	public	TControl
HandleAllocated	public	TWinControl
HandleNeeded	public	TWinControl
HasParent	public	TComponent
Hide	public	TControl
InheritsFrom	public	TObject
InitInstance	public	TObject
InsertComponent	public	TComponent
InsertControl	public	TWinControl
InstanceSize	public	TObject
Invalidate	public	TWinControl
MethodAddress	public	TObject
MethodName	public	TObject
NewInstance	public	TObject
PaintTo	public	TWinControl

Method	Visibility	Documented in
PasteFromClipboard	public	TCustomEdit
Perform	public	TControl
Realign	public	TWinControl
Refresh	public	TControl
RemoveComponent	public	TComponent
RemoveControl	public	TWinControl
Repaint	public	TWinControl
ScaleBy	public	TWinControl
ScreenToClient	public	TControl
ScrollBy	public	TWinControl
SelectAll	public	TCustomEdit
SendToBack	public	TControl
SetBounds	public	TWinControl
SetFocus	public	TWinControl
SetSelTextBuf	public	TCustomEdit
SetTextBuf	public	TControl
Show	public	TControl
TEdit	public	TEdit
Update	public	TWinControl
UpdateControlState	public	TWinControl

~TEdit

~TEdit frees the memory associated with the *TEdit* object. Do not call *~TEdit* directly. Instead, use the **delete** keyword on the object, which causes *~TEdit* to be invoked automatically.

```
__fastcall virtual ~TEdit(void);
```

TEdit

TEdit creates a new *TEdit* object.

```
__fastcall virtual TEdit(Classes::TComponent* AOwner);
```

TEnumPropDesc

Header: olectrls.hpp

TObject • TEnumPropDesc

TEnumPropDesc provides the mapping between strings used to name the values of a property in an OLE control object and the values themselves.

Description Use *TEnumPropDesc* to interpret the strings used by an OLE control object to represent the values of one of its properties. *TEnumPropDesc* can provide a list of possible value strings to a callback function, or map between the string names of property values and the values themselves. *TEnumPropDesc* objects are made available through the *GetEnumPropDesc* method of an OLE control object.

See also *TOleControl::GetEnumPropDesc* method

Methods

The following table lists the methods of *TEnumPropDesc*:

Method	Visibility	Documented in
~TEnumPropDesc	public	TEnumPropDesc
ClassInfo	public	TObject
ClassName	public	TObject
ClassNameIs	public	TObject
ClassParent	public	TObject
ClassType	public	TObject
CleanupInstance	public	TObject
DefaultHandler	public	TObject
Dispatch	public	TObject
FieldAddress	public	TObject
Free	public	TObject
FreeInstance	public	TObject
GetStrings	public	TEnumPropDesc
InheritsFrom	public	TObject
InitInstance	public	TObject
InstanceSize	public	TObject
MethodAddress	public	TObject
MethodName	public	TObject
NewInstance	public	TObject
StringToValue	public	TEnumPropDesc
TEnumPropDesc	public	TEnumPropDesc
ValueToString	public	TEnumPropDesc

~TEnumPropDesc

~TEnumPropDesc frees the memory associated with the *TEnumPropDesc* object. Do not call *~TEnumPropDesc* directly. Instead, use the **delete** keyword on the object, which causes *~TEnumPropDesc* to be invoked automatically.

```
__fastcall virtual ~TEnumPropDesc(void);
```

Description *~TEnumPropDesc* frees the internal list used to map value strings to property values.

See also *TOleControl* object

GetStrings

GetStrings calls the callback *Proc* for each property value string.

```
void __fastcall GetStrings(Classes::TGetStrProc Proc);
```

Description Use *GetStrings* to execute code for every property value string represented in the *TEnumPropDesc* object. The callback *Proc* is called for every value, with the *S* parameter set to a string that gives the integer value followed by the string representation of that value.

For example, consider a *TEnumPropDesc* object that represents a color property, with values equal to *TColor* values. The callback *Proc* would be called for every built-in color name with strings of the form '255 – clRed'.

See also *StringToValue* method, *ValueToString* method

StringToValue

StringToValue maps a string representation of the property value to the integer value.

```
int __fastcall StringToValue(const System::AnsiString S);
```

Description Use *StringToValue* to map the string representation of a property value used by an OLE control object to an integer value that is a more accurate representation of the property value. If the property value is not an integer, the integer associated with the string can be a pointer to the value.

See also *ValueToString* method

TEnumPropDesc

TEnumPropDesc creates an instance of *TEnumPropDesc*.

```
__fastcall TEnumPropDesc(int DispID, int ValueCount, Ole2::ITypeInfo* TypeInfo);
```

Description *TEnumPropDesc* is called by an OLE control object to represent the values of one of its properties. Applications should not need to create instances of

TEnumPropDesc. When working with OLE properties, get the *TEnumPropDesc* objects from the OLE control object.

DispID identifies the property of the OLE object, *ValueCount* is the number of distinct values the property can take, and *TypeInfo* is an *ITypeInfo* object that the constructor can use to capture the values and their strings.

See also *TOleControl::GetEnumPropDesc* method

ValueToString

ValueToString maps an integer that indicates the value of the property to the string representation of that value used by the OLE control object.

```
System::AnsiString __fastcall ValueToString(int V);
```

Description Use *ValueToString* to map the integer value of a property value to the string representation used by an OLE control object. If the property value is not an integer, the integer associated with the string can be a pointer to the value.

See also *StringToValue* method

TStringGrid

Header: grids.hpp Palette: Additional

TObject • TPersistent • TComponent • TControl • TWinControl • TCustomControl • TCustomGrid • TDrawGrid • TStringGrid

TStringGrid is a grid control designed to simplify the handling of strings and associated objects.

Description Add a *TStringGrid* object to a form to present textual data in a tabular format. *TStringGrid* provides many properties to control the appearance of the grid, as well as events and methods that take advantage of the tabular organization of the grid in responding to user actions.

TStringGrid introduces the ability to associate an object with each string in the grid. These objects can encapsulate any information or behavior represented by the strings that are presented to the user.

If the strings to be presented in a grid represent field values from the records in a dataset, use *TDBGrid* instead.

See also *TDBGrid* object

Properties

The following table lists the properties of *TStringGrid*:

Property	Visibility	Default	Documented in
Align	published	alNone	TControl
BorderStyle	published	bsSingle	TCustomGrid
BoundsRect	public		TControl
Brush	public, read-only		TWinControl
Canvas	public, read-only		TCustomControl
Cells	public		TStringGrid
ClientHeight	public		TControl
ClientOrigin	public, read-only		TWinControl
ClientRect	public, read-only		TWinControl
ClientWidth	public		TControl
Col	public		TCustomGrid
ColCount	published	5	TCustomGrid
Color	published	clWindow	TControl
Cols	public		TStringGrid
ColWidths	public		TCustomGrid
ComponentCount	public, read-only		TComponent
ComponentIndex	public		TComponent
Components	public, read-only		TComponent
ComponentState	public, read-only		TComponent

Property	Visibility	Default	Documented in
ComponentStyle	public, read-only		TComponent
ControlCount	public, read-only		TWinControl
Controls	public, read-only		TWinControl
ControlState	public		TControl
ControlStyle	public		TControl
Ctl3D	published		TWinControl
Cursor	published	crDefault	TControl
DefaultColWidth	published	64	TCustomGrid
DefaultDrawing	published	true	TCustomGrid
DefaultRowHeight	published	24	TCustomGrid
DesignInfo	public		TComponent
DragCursor	published	crDrag	TControl
DragMode	published	dmManual	TControl
EditorMode	public		TCustomGrid
Enabled	published	true	TControl
FixedColor	published	clBtnFace	TCustomGrid
FixedCols	published	1	TCustomGrid
FixedRows	published	1	TCustomGrid
Font	published		TControl
GridHeight	public, read-only		TCustomGrid
GridLineWidth	published	1	TCustomGrid
GridWidth	public, read-only		TCustomGrid
Handle	public, read-only		TWinControl
Height	published		TControl
HelpContext	published	0	TWinControl
Hint	published		TControl
Left	published		TControl
LeftCol	public		TCustomGrid
Name	published		TControl
Objects	public		TStringGrid
Options	published	[goFixedVertLine, goFixedHorzLine, goVertLine, goHorzLine, goRangeSelect]	TCustomGrid
Owner	public, read-only		TComponent
Parent	public		TControl
ParentColor	published	false	TControl
ParentCtl3D	published	true	TWinControl
ParentFont	published	true	TControl
ParentShowHint	published	true	TControl
ParentWindow	public		TWinControl
PopupMenu	published		TControl

Property	Visibility	Default	Documented in
Row	public		TCustomGrid
RowCount	published	5	TCustomGrid
RowHeights	public		TCustomGrid
Rows	public		TStringGrid
ScrollBars	published	ssBoth	TCustomGrid
Selection	public		TCustomGrid
ShowHint	published		TControl
Showing	public, read-only		TWinControl
TabOrder	published	-1	TWinControl
TabStop	published	true	TWinControl
TabStops	public		TCustomGrid
Tag	published	0	TComponent
Top	published		TControl
TopRow	public		TCustomGrid
Visible	published	true	TControl
VisibleColCount	published, read-only		TCustomGrid
VisibleRowCount	published, read-only		TCustomGrid
Width	published		TControl
WindowProc	public		TControl

Cells

Cells is an array of strings, one string for each cell in the grid.

```
__property System::AnsiString Cells[int ACol][int ARow];
```

Description Use *Cells* to access the string within a particular cell. *ACol* is the column coordinate of the cell, and *ARow* is the row coordinate of the cell. The first row is row zero, and the first column is column zero.

The *ColCount* and *RowCount* property values define the size of the array of strings.

To access the objects associated with the strings in the *Cells* array, use the *Objects* property.

See also *ColCount* property, *Objects* property, *RowCount* property

Cols

Cols is an array of the lists strings with their associated objects for each column.

```
__property Classes::TStrings* Cols[int Index];
```

Description Use *Cols* to access all the strings for a single column, along with their associated objects. The number of strings and associated objects is always

equal to the value of the *ColCount* property, the number of columns in the grid. The *Index* parameter is the number of the column, where the value of the first column in the grid is zero.

To get all the strings and objects for a row, rather than for a column, use the *Rows* property.

See also *ColCount* property, *Objects* property, *Rows* property

Objects

Objects is an array of objects, one for each cell in the grid.

```
__property System::TObject* Objects[int ACol][int ARow];
```

Description Use the *Objects* property to access the object associated with a particular cell. The *ColCount* and *RowCount* values define the size of the array of objects. *ACol* is the column coordinate of the cell, and *ARow* is the row coordinate of the cell, where the first column is column 0, and the first row is row 0.

Note The string grid does not own the objects in the *Objects* array. Objects added to the *Objects* array still exist even if the string grid is destroyed. They must be explicitly destroyed by the application.

See also *Cells* property, *ColCount* property, *RowCount* property

Rows

Rows is an array of the lists of strings and their associated objects for each row.

```
__property Classes::TStrings* Rows[int Index];
```

Description Use *Rows* to access all the strings for a single row, along with their associated objects. The number of strings and associated objects is always equal to the value of the *RowCount* property, the number of rows in the grid. The *Index* parameter is the number of the row, where the value of the first row in the grid is zero.

To get all the strings and objects for a column, rather than for a row, use the *Cols* property.

See also *ColCount* property, *Cols* property, *Objects* property

Events

The following table lists the events of *TStringGrid*:

Event	Visibility	Documented in
OnClick	published	TControl
OnColumnMoved	published	TDrawGrid

Event	Visibility	Documented in
OnDblClick	published	TControl
OnDragDrop	published	TControl
OnDragOver	published	TControl
OnDrawCell	published	TDrawGrid
OnEndDrag	published	TControl
OnEnter	published	TWinControl
OnExit	published	TWinControl
OnGetEditMask	published	TDrawGrid
OnGetEditText	published	TDrawGrid
OnKeyDown	published	TWinControl
OnKeyPress	published	TWinControl
OnKeyUp	published	TWinControl
OnMouseDown	published	TControl
OnMouseMove	published	TControl
OnMouseUp	published	TControl
OnRowMoved	published	TDrawGrid
OnSelectCell	published	TDrawGrid
OnSetEditText	published	TDrawGrid
OnStartDrag	published	TControl
OnTopLeftChanged	published	TDrawGrid

Methods

The following table lists the methods of *TStringGrid*:

Method	Visibility	Documented in
~TStringGrid	public	TStringGrid
Assign	public	TPersistent
BeginDrag	public	TControl
BringToFront	public	TControl
Broadcast	public	TWinControl
CanFocus	public	TWinControl
CellRect	public	TDrawGrid
ClassInfo	public	TObject
ClassName	public	TObject
ClassNameIs	public	TObject
ClassParent	public	TObject
ClassType	public	TObject
CleanupInstance	public	TObject
ClientToScreen	public	TControl
ContainsControl	public	TWinControl

Method	Visibility	Documented in
ControlAtPos	public	TWinControl
DefaultHandler	public	TObject
DestroyComponents	public	TComponent
Destroying	public	TComponent
DisableAlign	public	TWinControl
Dispatch	public	TObject
DragDrop	public	TControl
Dragging	public	TControl
EnableAlign	public	TWinControl
EndDrag	public	TControl
FieldAddress	public	TObject
FindComponent	public	TComponent
Focused	public	TWinControl
Free	public	TObject
FreeInstance	public	TObject
FreeNotification	public	TComponent
GetParentComponent	public	TComponent
GetTabOrderList	public	TWinControl
GetTextBuf	public	TControl
GetTextLen	public	TControl
HandleAllocated	public	TWinControl
HandleNeeded	public	TWinControl
HasParent	public	TComponent
Hide	public	TControl
InheritsFrom	public	TObject
InitInstance	public	TObject
InsertComponent	public	TComponent
InsertControl	public	TWinControl
InstanceSize	public	TObject
Invalidate	public	TWinControl
MethodAddress	public	TObject
MethodName	public	TObject
MouseCoord	public	TCustomGrid
MouseToCell	public	TDrawGrid
NewInstance	public	TObject
PaintTo	public	TWinControl
Perform	public	TControl
Realign	public	TWinControl
Refresh	public	TControl
RemoveComponent	public	TComponent
RemoveControl	public	TWinControl
Repaint	public	TWinControl

Method	Visibility	Documented in
ScaleBy	public	TWinControl
ScreenToClient	public	TControl
ScrollBy	public	TWinControl
SendToBack	public	TControl
SetBounds	public	TWinControl
SetFocus	public	TWinControl
SetTextBuf	public	TControl
Show	public	TControl
TStringGrid	public	TStringGrid
Update	public	TWinControl
UpdateControlState	public	TWinControl

TStringGrid

TStringGrid creates an instance of *TStringGrid*.

```
__fastcall virtual TStringGrid(Classes::TComponent* AOwner);
```

Description Call *TStringGrid* to create an instance of *TStringGrid* at runtime. For string grids placed on forms at design time, *TStringGrid* is called automatically.

After calling the constructor of its base class, *TStringGrid* creates the helper objects used to manage the array of strings and their associated objects.

See also *TCustomGrid::TCustomGrid* method

~TStringGrid

~TStringGrid frees the memory associated with the *TStringGrid* object. Do not call *~TStringGrid* directly. Instead, use the **delete** keyword on the object, which causes *~TStringGrid* to be invoked automatically.

```
__fastcall virtual ~TStringGrid(void);
```

Description *TStringGrid* frees the helper objects used to manage the array of strings and their associated objects.