

1 Disclaimer

This software is delivered as it is. The author assumes no liability for damages, direct or consequential, which may result from it's use.

2 Copyright / Licensing

The software is owned by gig mbh berlin (www.gig-mbh.de).

Two different licenses are available:

1. Free License

Everyone who wants to use the free license has to register with his full name and address via support@gig-mbh.de.

Every software where parts of our free software were used for development has to be free also including source code.

If you derive anything from our software it must be clearly stated that it was derived from it.

Full source code is included.

2. Extended License

Licenses have to be bought by a per developer basis. Site licenses would be available on demand.

Applications built with this software could be deployed without royalty fees. They can be sold and don't need to include source code.

Distribution of a derived version of our software is only allowed with the explicit agreement of the author.

Full source code is included.

3 Support

Support is available via email at support@gig-mbh.de for free but it need not remain so in the future.

4 Introduction

This document describes the function of the components TCustomDBGridEC, TDBGridEC and TIBGridEC.

These components do not represent another completely different grid component implementation written from scratch, it is rather a derivation of the standard VCL's TDBGrid component with a few modifications and enhancements.

The intention for developing these component was in the first place to have grid components working together with our Interbase table component TIBScrollSetEC and our in-memory table component TMemTableEC and offer extended features like sorting and incremental searching without the need for coding.

We have derived a generic class called TCustomDBGridEC from TDBGrid where all modifications and enhancements have been made in a generic dataset independent way. As some of the features need a more specific implementation related to the dataset they are linked to this was done in a derivation of TCustomDBGridEC called TIBGridEC for our Interbase table component TIBScrollSetEC and in TDBGridEC for our in-memory table component TMemTableEC.

This architecture gives you the ability to use TCustomDBGridEC with every TDataSet compatible component or to derive another class from it which includes the more specific implementation which is needed for some features to work with other datasets like the BDE related TTable. I'm sure we will include such component in the near future.

The component is completely written in C++ and was developed under C++Builder 5 Pro but it should be usable on C++ Builder 6 if compiled in it's environment.

Questions, bug reports , enhancement requests, suggestions for improving the docs and comments should be send to support@gig-mbh.de.

5 Modifications and Enhancements

- Focus rect will also be drawn on grids with the option `dgRowSelect` set.
- Focus rect will also be drawn on grids which are empty or where the linked dataset is not active.
- On empty grids the active cell or row will not be shown selected (similar to Windows' listbox components), only the focus rect is shown.
- Redraw of focus rect on horizontal scroll to fix a display issue.
- The options `dbgeoAllwaysShowVBar` and `dbgeoAllwaysShowHBar` make it possible to have permanently shown scroll bars.
- Properties `DefaultMinColWidth`, `DefaultMaxColWidth` and `MinMaxColWidths` to ensure the minimum and maximum column widths for a single column or the whole grid.
- Implementation of an event handler called `OnColumnResizeQuery` to control which columns could be resized.
- Implementation of an event handler called `OnColResized` where adjustments to columns could be made after a sizing operations just before they become active.
- Property `ListMode` which ensures that all columns fit into the grid's client area without showing a horizontal scroll bar. It also controls which columns will be adjusted on resizing operations.
- Indication and changing column sorting in corporation with our Interbase dataset and in-memory table components, and in a generic way in `TDBCustGridEC` to be implemented with other `TDataSet` descendants.
- We have implemented a different scroll behaviour which we think is more pleasant, but you have the choice by setting the `dbgeoClassicScroll` option.
- Fix for generating the `OnMouseDown` event which was not fired on some areas of the grid.
- Property `PopupMenuAreas` to control in which areas a right mouse click will show a popup menu.
- Method `TestMouseArea` for checking on which area of the grid a mouse event occurred

- Methods GetColField, BeginUpdate and EndUpdate are made public.
- Option dbgeoDisableColMoving to enable the resizement of columns but not their movement.
- Possibility to freeze the grid appearance to hide actions performed by the application from the user interface. This is not completely possible with the standard grid components.

6 Methods

SetActiveRecord

Description: Moves the active record to the row number specified

Prototype : void __fastcall SetActiveRecordPos(int newpos)

Parameters: **newpos** - specifies the number of the row where the active record will be moved to.

Return values : none

Type: public

TestMouseArea

Description: Returns the area of the grid at the coordinates specified.

Prototype : TDBGridArea __fastcall TestMouseArea(int x, int y)

Parameters: **x** - X-coordinate of the specified area.

y - Y-coordinate of the specified area.

Return value : **dbgIndicatorCol** - Indicator column was specified.
dbgTitle - Title row was specified.
dbgDataArea - Data area was specified.
dbgEmptyArea - Empty area was specified.

Type: public

GetColField

Description: Described in the TCustomDBGrid documentation of the VCL.

BeginUpdate

Description: Described in the TCustomDBGrid documentation of the VCL.

EndUpdate

Description: Described in the TCustomDBGrid documentation of the VCL.

BeginColResize

Description: If you want to avoid internal adjustment operations to be performed on the change of column sizes call this method before and EndColumnResize when you have finished. If you call this method multiple times take care of calling EndColumnResize for every call to BeginColumnResize.

Prototype : void __fastcall BeginColResize(void)

Parameters: none

Return values : none

Type: public

EndColResize

Description: See BeginColResize method.

Prototype : void __fastcall EndColResize(bool repaint = true)

Parameters: **repaint** - Forces the grid to be invalidated

Return values : none

Type: public

SetSortCol

Description: This method is called to set a sort order to a specified column.

Prototype : void __fastcall SetSortCol(TColumn *col, TSortDirection direction)

Parameters: **col** - Pointer to the TColumn object for which new sorting information is being passed.

direction - The new sort order for the specified column. This could be one of the following values: sodiAscending, sodiDescending or sodiNone.

Return values : none

Type: public

GetSortColInfo

Description: This method has to be overwritten in TCustomDBGridEC descendants to enable column sorting. For using TIBGridEC and TDBGridEC you don't have to care about it. These components use this method to get sorting informations on a specific column.

Prototype : virtual void __fastcall GetSortColInfo(TColumn *col,
TSortDirections &Directions,
TSortDirection &CurDir)

Parameters: **col** - Pointer to the TColumn object for which information is queried.

Directions - Here you have to specify the sortorders the specified column supports. This argument may contain a set of the following values: sodiAscending, sodiDescending or sodiNone.

CurDir - Set this parameter to the currently assigned sortorder of the specified column (sodiAscending or sodiDescending). If the specified column is not the current sort column set this parameter to sodiNone.

Return values : none

Type: protected

ChangeSortCol

Description: This method has to be overwritten in TCustomDBGridEC descendants to enable column sorting. For using TIBGridEC and TDBGridEC you don't have to care about it. The component is notified by this method, that the current sort column was changed and to perform all operations necessary so that this change is reflected by the linked TDataSet descendant.

Prototype : virtual void __fastcall ChangeSortCol(TColumn *col,
TSortDirection direction)

Parameters: **col** - Pointer to the TColumn object for which new sorting information is being passed.

direction - The new sort order for the specified column. This could be one of the following values: sodiAscending, sodiDescending or sodiNone.

Return values : none

Type: protected

GetSortCol

Description: Return a pointer to the TColumn object of the current sort column.

Prototype : TColumn * __fastcall GetSortCol(void)

Parameters: none

Return values : Pointer to the current sort column

Type: public

StartIncSearch

Description: Starts incremental searching on the current sort column. An edit control for entering the value to search for is displayed on top of the column title. Searching is also automatically started if you press a key if the grid has the focus and dgEditing is not specified in the grid's options.

Prototype : bool __fastcall StartIncSearch(void)

Parameters: none

Return values : boolean value indicating if sorting could be started or not.

Type: public

EndIncSearch

Description: Terminates the incremental search if currently active. The displayed edit control for entering search values disappears and the grid get back focus. This also happens automatically if you press ESC, ENTER, UP, DOWN or if the edit control loses focus. On pressing ESC the record position before starting incremental search mode is restored if the option dbgeolncSearchRestorePos is set, on all other events the current record position is not changed.

Prototype : void __fastcall EndIncSearch(bool focusgrid = true)

Parameters: **focusgrid** - Indicates whether the grid gets the focus after the search control has been closed

Return values : none

Type: public

CanStartIncSearch

Description: This method can be overwritten in descendant classes if you want to restrict the availability of incremental search to other than the available sort columns. In TIBGridEC and TDBGridEC this has been done to control availability of incremental search by the SortColLinks property.

Prototype : virtual bool __fastcall CanStartIncSearch(TColumn *col)

Parameters: **col** - Pointer to the column to check for incremental search

Return values : true if searching should be available otherwise false

Type: protected

DoIncSearch

Description: This method must be overwritten in descendant classes. It is called whenever the entered search string changes to locate to a new record position.

Prototype : virtual void __fastcall DoIncSearch(const AnsiString text)

Parameters: **text** - Search string to locate

Return values : none

Type: protected

7 Properties

ExOptions

Description: Here you can set different options to influence grid behaviour.

dbgeoAllwaysShowVBar - A vertical scrollbar is permanently shown and not only when necessary for scrolling.

dbgeoAllwaysShowHBar - A horizontal scrollbar is permanently shown and not only when necessary for scrolling.

dbgeoAllwaysShowFocusRect - On ownerdrawn grids (DefaultDrawing == false) the focus rect is not drawn by default. By setting this option you could force the focus rect allways to be drawn.

dbgeoAllwaysShowSorting - On ownerdrawn grids (DefaultDrawing == false) the sorting indicator symbols are not drawn by default. By setting this option you could force the symbols allways to be drawn.

dbgeoShowSortableCols - All sortable columns are marked in the header, otherwise only the currently sorted column is indicated.

dbgeoDisableColMoving - If you add dgColumnResize to the Options property of the grid resizing and moving of columns will be enabled. By setting this option you could disable moving of columns resulting in a grid where only column resizing is possible.

dbgeoDisableAppendRecord - Appending records by moving the cursor behind the last row is no longer possible.

dbgeoDisableInsertRecord - Inserting records by pressing <INS> is no longer possible.

dbgeoDisableDeleteRecord - Deleting records by pressing is no longer possible.

dbgeoThumbTracking - When this option is set scrolling operations are preformed immediately after the thumb is moved and not only after it has been released.

dbgeoCanDisableSorting - Here you could specify if the underlying dataset supports an undefined (natural) sort order. If this option is no set it is not possible to remove a column's sort indication by a mouse click. If set to yes clicking on a column in descending order will remove the sort indication.

dbgeoClassicScroll - Switches between the normal and a modified scroll behaviour of the grid. Simply check it out.

Type: published

Description:	This property specifies the minimum width every column on the grid must have at least. It will be ensured on all sizing operations.
---------------------	---

Type: published

Description: This property specifies the maximum width every column on the grid can have. It will be ensured on all sizing operations.

Type: published

PopupMenuAreas

Description: This set of values specify the areas where a right mouse button click will show a popup menu. Valid values are any combination of: dbgaIndicatorCol, dbgaTitle, dbgaDataArea and dbgaEmptyArea

Definition : `__property TDBGridAreas PopupMenuAreas =
{read=FPopupMenuAreas,
write=FPopupMenuAreas,
default=15}`

Type: published

ListMode

Description: With this property you can enforce all columns to be shown in the visible area of the grid without using a horizontal scrollbar. In this mode it also controls how the resizing of a column or the whole grid affects the size of other columns. Possible values are:

limoNone - Normal grid mode without any adjustment.

limoLastCol - The last column is always adjusted to fill the grid's area

limoNextCol - The column on the right of the resized one is always adjusted to fill the grid's area.

limoRightCols - All columns on the right of the resized one are adjusted to fill the grid's area.

limoAllCols - All columns are adjusted to fill the grid's area.

Definition : `__property TListModeType ListMode = {read=FListMode,
write=SetListMode,
default=limoNone}`

Type: published

MinMaxColWidths

Description: With this property you can specify the min and max column width for every single column. Every single line in the string list contains the following values for one column: <name>;<min>;<max>. Name specifies the DisplayName of the column (This is the name also shown in the object inspector), min and max the minimum and maximum pixel width of the column. For min and max you can also specify -1 which results in the appropriate default value to take effect.

Definition : `__property TMinMaxColWidths *MinMaxColWidths =
{read=FMinMaxColWidths,
write=SetMinMaxColWidths}`

Type: published

SortColSymbols

Description: If you want to have your own symbols for indication of sort columns you could assign a TImageList to this property. It must contain three bitmaps for the appropriate symbols. The first image will be used for ascending indication, the second one for descending and the third for marking all sortable columns. If nothing is specified for this property default symbols will be used.

Definition : `__property TImageList *SortColSymbols = {read=FSortColSymbols,
write=SetSortColSymbols}`

Type: published

SortColLinks

Description: This property establishes a link between the columns in a grid which you want to be sortable by clicking on their title and the items of the OrderItems property of our TIBScrollSetEC or TMemTableEC dataset which contains the ordering specifications for the sort orders which are available. Every single line in the string list contains the following values for one column:
<name>;<order_item_id>;<asc>;<desc>;<inc search>.
Name specifies the DisplayName of the column (This is the name also shown in the object inspector), order_item_id specifies the id of the OrderItem property of the dataset you want this column to be linked to. Asc, desc and inc search contains „true” or „false” whether you want ascending or descending sortorder to be supported and if you want incremental search to be available for this column.

Definition : `__property TSortColLinks *SortColLinks = {read=FSortColLinks, write=SetSortColLinks}`

Type: published (only available in TIBGridEC and TDBGridEC)

IncSearchActive

Description: This property indicates whether the grid is currently in incremental search mode. It is also possible to start or end incremental search mode by setting this property.

Definition : `__property bool IncSearchActive = {read=FIncSearchActive, write=SetIncSearchActive}`

Type: public

IncSearchDelay

Description: You can set a delay value in milliseconds. An incremental search action is performed only if no additional key was pressed for the amount of time specified. This property is useful if you use the grid in applications operating over a slow WAN link to reduce the number of queries performed to locate the new record.

Definition : `__property int IncSearchDelay = {read=FIncSearchDelay,
write=FIncSearchDelay,
default=0};`

Type: published

FreezeGrid

Description: Can be used to „freeze” the current display state of the grid. After that any modification to the grid are covered. When you are done with the modifications you could set the property back to false and the changes become visible to the user.

Definition : `__property bool FreezeGrid = {read=FFreezeGrid,
write=SetFreezeGrid};`

Type: public

8 Events

OnColResizeQuery

Description: This event is fired for every resized column. Set the CanResize parameter of the event handler to false if you want to avoid the resizement of the column.

Handler : void __fastcall (__closure *TOnColSizeQueryEvent)(TColumn* Col, bool &CanResize)

Type: published

OnColResized

Description: This event is fired after all columns have been resized. From within the event handler it is possible to adjust column sizes just before they take effect.

Handler : void __fastcall (__closure *TNotifyEvent)(TObject *Sender)

Type: published

OnChangeSortCol

Description: This event is fired whenever the current sort column is changed.

Handler : void __fastcall (__closure *TonChangeSortColEvent)(TColumn *col, TSortDirection direction)

Type: published

OnCanStartIncSearch

Description: Before incremental search starts this event is fired. If you do not want to start Incremental search in this situation then set the CanStart parameter to false.

Handler : void __fastcall (__closure *TonCanStartIncSearchEvent)
(TColumn* col, bool &CanSearch);

Type: published

OnCreateIncSearchCtrl

Description: After the edit control for incremental search was created this event is fired and gives you the ability to change the control's properties.

Handler : void __fastcall (*TONCreateIncSearchCtrl)(TColumn *col,
TEditEC *edtctrl);

Type: published

OnIncSearch

Description: This event is fired whenever the text of the search edit cotrol changes and locating to a new record is required. Use this event handler to impement our own search routine. If the parameter Done is set to true the default search routine will not be called.

Handler : void __fastcall (__closure *TonIncSearchEvent)
(AnsiString &Text, bool &Done)

Type: published