TMS TAdvDBFormPanel - TAdvDBFormBox DEVELOPERS GUIDE

July 2019

Web: https://www.tmssoftware.com
Email: info@tmssoftware.com

Table of contents

Introduction

The TMS TAdvDBFormPanel and TAdvDBFormBox are highly configurable automatic or semi-automatic database form generators.

TMS TAdvDBFormPanel and TAdvDBFormBox can automatically generate a form of DB-aware controls for the fields found in a dataset. The generation of the form is controlled by layout settings.

The generation of the form can be done both at design-time and run-time. Several ways to customize the generation of DB controls as well as customize the generated forms and its controls are available.

The DB forms feature automatic editing and browse modes. In browse mode, the database data is displayed either with labels or not enabled DB controls while in edit mode, DB edit controls are used.

TMS TAdvDBFormPanel and TAdvDBFormBox feature configurable mapping of DB field types to control classes. Default, the mapping for DB field types is on advanced TMS DB-aware controls for an enhanced user-experience. But as the class mapping is configurable, any DB-aware control can be used on the TAdvDBFormPanel or TAdvDBFormBox

Availability

TMS TAdvDBFormPanel and TAdvDBFormBox are available as VCL component for Delphi and C++Builder.

TMS TAdvDBFormPanel and TAdvDBFormBox are available for Delphi XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10, 10.1, 10.2, 10.3 & C++Builder XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10, 10.1, 10.2, 10.3.

TMS TAdvDBFormPanel and TAdvDBFormBox has been designed for and tested with: Vista, Windows 7, Windows 8, Windows 10 in Win32 applications and with Delphi XE2, C++Builder XE2 or newer versions, it can be used in both 32bit and 64bit Windows applications.

List of included components

TAdvDBFormPanel: A panel component that can contain automatic generated DB controls

TDBAdvFormBox: A panel component with a scrollable area that can contain automatic generated DB controls with a caption. From the caption, the form can be put in editing or in browse mode.

Online references

TMS software website:
http://www.tmssoftware.com

TMS TAdvDBFormPanel page:
http://www.tmssoftware.com/site/advdbformpanel.asp

TMS TAdvDBFormBox page:
http://www.tmssoftware.com/site/advdbformbox.asp

TMS manuals:
http://www.tmssoftware.com/site/manuals.asp

TAdvDBFormPanel

TAdvDBFormPanel description

The TMS TAdvDBFormPanel is a highly configurable DB form generator.

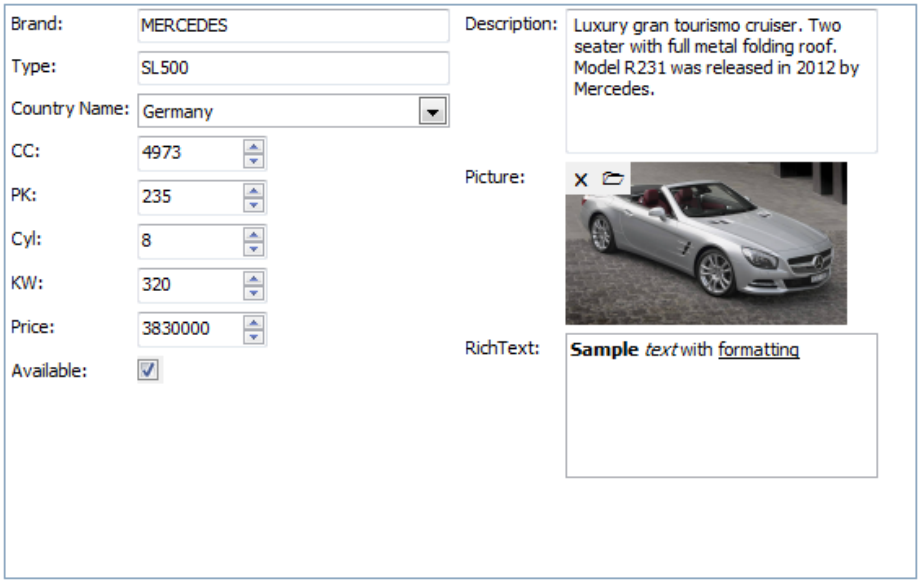
The TMS TAdvDBFormPanel can have DB controls for the connected dataset automatic or semi-automatic generated. DB control generation can be assisted via a collection of items or via an event. The DB control generation can be done at design-time or at run-time.

Layout settings control how the form is generated. This can be in column or row mode and with or without labels for each DB control.

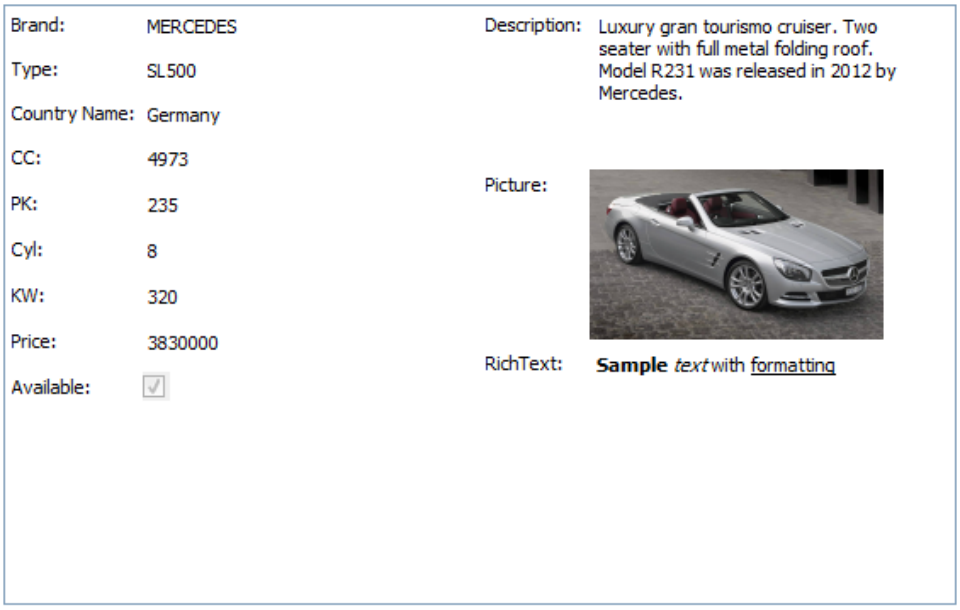
TAdvDBFormPanel features

- Automatic DB form generation for connected dataset
- Can generate the DB form at design-time and/or at run-time
- DB form generation can be fully automatic or semi-automatic, assisted by collection of items that controls each DB control that will be generated
- Layout settings to control generation of the form in column or row mode
- Via layout, it can be controlled whether labels for each DB control are used or not
- Layout can be persisted in JSON format
- Configurable mapping of DB field types to DB class types
- Events are provided to control what DB fields are used on the form
- Events are provided to customize properties of generated DB controls
- Automatic switching between edit mode and browse mode with representation of DB values via label or not enabled DB controls in browse mode

TAdvDBFormPanel architecture



TAdvDBFormPanel is a panel. It is a container for controls. The controls on the panel can be automatically generated according to settings under layout and are either fully automatic generated child controls that are not individually accessible at design-time or via the context menu in the IDE designer, the controls can be generated for design-time access as well.
In addition to automatic generated DB controls, other controls can be added to the panel as well at both design-time and runtime.
With the EditMode property, the form can be automatically switched to browse mode where data is represented by labels or not enabled controls:

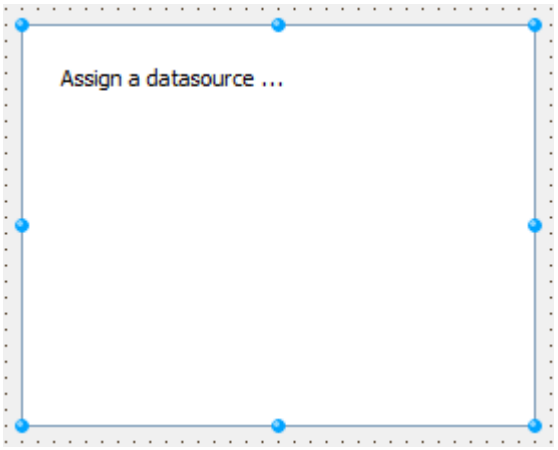


TAdvDBFormPanel use

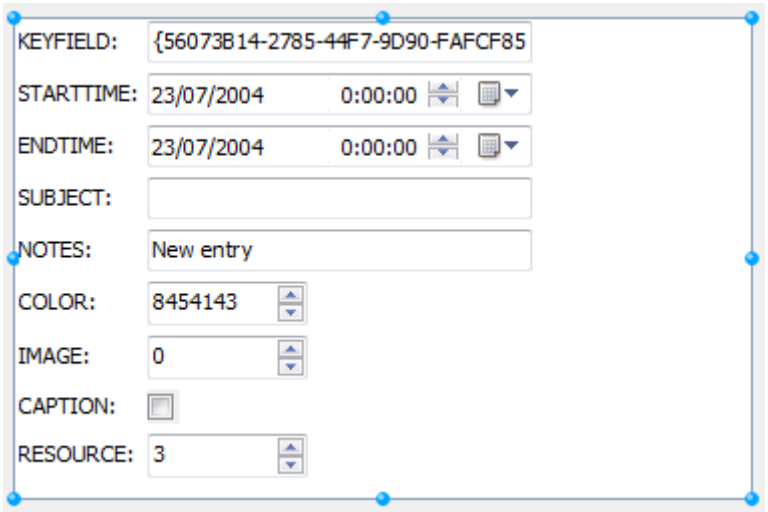
Getting started

From the component palette, select the TAdvDBFormPanel control and drop it on a form.

This shows an empty panel.



Drop a datasource and dataset on the form and assign the datasource to the panel. With default settings, the DB controls will be generated for the dataset:



Under TAdvDBFormPanel .Layout, items will be added for each generated DB control. With this items collection, further customization to the generated form can be done.

TAdvDBFormPanel properties

- **AutoEdit:** Boolean: When set to true, the TAdvDBFormPanel will automatically display the DB values via labels or not enabled DB controls when the dataset is in dsBrowse state and via DB edit controls when the dataset is in dsEdit state.
- **AutoLayout:** Boolean: When set to true, the TAdvDBFormPanel will automatically generate the DB form as soon as an active dataset is connected. When false, this will not happen. The DB form can then be generated either with calling Panel.InitLayout / Panel.ShowLayout or by choosing “Add DB controls” from the context menu at design time.
- **BorderColor:** TColor: sets the color of the panel border.
- **Color:** TColor: sets the start color of the gradient panel background.
- **ColorTo:** TColor: sets the end color of the gradient panel background.
- **EditMode:** TEditMode: sets the panel DB controls either in edit mode (emOn) or in browse mode (emOff). When emOff is selected, DB values are displayed via a label or not-enabled DB control. When emOn is selected, the DB values are in regular DB edit controls.
- **Layout:** Class property of TLayout type (See detailed information in the TLayout paragraph). Controls the layout of the generated form of DB controls.

TLayout class

- **Columns: Integer:** Gets / sets the number of desired columns. The total number of DB fields in the dataset is divided by the desired number of columns and in each column, this number of DB controls is generated. Columns is used when layout mode is lmColumns
- **Controls.Autosize: Boolean:** When true, the width of the DB controls is determined from the DB field's DisplayWidth setting. When false, the width is taken from Controls.Size.
- **Controls.MaximumSize: Integer:** Gets / sets the maximum width that will be used for a generated DB control.
- **Controls.ReadOnlyAsLabel: Boolean:** When true, a read-only DB field will be automatically represented by a label.
- **Controls.Size: Integer:** Gets / sets the default width of DB controls that will be used when Controls.AutoSize = false.
- **Items: TLayoutItems:** Collection of DB control items. See TLayoutItem class.
- **Labels.AutoSize: Boolean:** When true, the width of the DB control label is adapted to the size of the text. When false, the width is taken from Labels.Size.
- **Labels.Font: TFont:** Gets / sets the font for the labels.
- **Labels.Format: string:** Gets / sets the formatting string to be used for the label text. Default value is '%s:'
- **Labels.Position: TLabelPosition;** Allows to select the position of the label relative to the DB control position as lsLeftTop, lsLeftCenter, lsLeftBottom, lsTop, lsBottom.
- **Labels.Size: Integer** When different from zero, sets the fixed size to be used for labels.
- **Margins: TMargins:** Sets the margins to respect from outer panel rectangle to position the generated DB controls.
- **Mode: TLayoutMode:** Selects between column or row layout for generating the DB controls.
- **Spacing:TSpacing:** Sets the vertical and horizontal spacing between controls and labels and controls on the DB form.

The layout can be persisted to JSON. Use the function TLayout.ToJSON: string to get a JSON representation of the TLayout class or use the procedure TLayout.FromJSON(const json: string) to retrieve the TLayout settings from a JSON text.

TLayoutItem class

- **ColumnBreak: Boolean:** When true, the next DB control generated will be in the next column (when Layout.Mode = lmColumns)
- **DataControl: TDataControl:** Sets the control type to use for the DB field:

  dcEdit: regular DB edit control
  dcSpinEdit: DB spin editor
  dcCheckBox; DB checkbox
  dcComboBox: DB combobox (values of the combobox are set with LayoutItem.Values)
  dcRadioGroup: DB radiogroup (values of the radiogroup are set with LayoutItem.Values)
  dcImage: DB image control
  dcMemo: DB memo
  dcRichEdit: DB rich text editor
  dcLookupComboBox: DB lookup combobox
  dcLabel: DB label
  dcNone: no control
  dcDateTime: DB datetimepicker for entry of both date & time
  dcMaskEdit: DB mask edit control
  dcDate: DB datetimepicker for entry of date
  dcTime: DB datetimepicker for entry of time

- **DataField: string :** Sets the dataset DB field to connect to the data control.
- **EditMask: string:** Sets the mask to use in case the datacontrol type is dcMaskEdit.
- **IsStatic: Boolean:** When true, the item does not switch to a label when the dataset is in dsBrowse state. It is just disabled when TAdvDBFormPanel.AutoEdit = true and the dataset is in dsBrowse state.
- **LabelCaption: string :** Sets the caption for the label accompanying the DB control. By default, the label caption is set to the DB field name.
- **LineBreak: Boolean:** When true, the next DB control generated will be in the next row (when Layout.Mode = lmRows)
- **ShowLabel: Boolean:** When true, a label is displayed together with the DB control for the DB field.
- **Values: TStringList :** Holds the values to use for either a dcComboBox or dcRadioGroup DB control type.

TAdvDBFormPanel methods

- **procedure InitClassMap(ControlSet: TDBControlSet);**

  Initializes the mapping table of DB field type to DB control classes. By default, the mapping is done via TMS DB edit controls (ControlSet = dbsTMS). A mapping can also be done to the standard VCL DB controls (ControlSet = dbsVCL).

- **procedure InitFields;**

  Automatically connects DB fields to an existing collection of layout items in Layout.Items.

- **procedure InitLayout;**

  Automatically creates the layout items collection from the connected dataset. The layout items collection will be filled automatically with the data control types derived from the class map.

- **procedure ShowLayout;**

  Render an existing set of layout items with DB controls in the panel.

- **procedure RemoveControls;**

  Removes the generated DB controls from the panel without affecting the layout items collection.

- **procedure RemoveLayout;**

  Remove the layout items and also the DB controls that were generated for it.

- **procedure SetEditMode(OnOff: TEditMode);**

  Set the panel in edit mode or in browse mode. When the edit mode is selected, DB fields are displayed with DB edit controls. When the browse is selected, the DB fields are displayed as labels or disabled DB controls.

TAdvDBFormPanel events

- **OnControlCreated(Sender: TObject; AControl: TControl; AField: TField);**

  Event triggered when a DB control is created for a DB field. This can be used for further customization of the DB control that was automatically created.

- **OnFieldMap(Sender: TObject; AField: TField; var Allow: Boolean);**

  Event triggered before the TAdvDBFormPanel will generate a layout item for a DB field found in the dataset. By setting Allow to false, this prevents that a layout item will be generated for a certain DB field.

  OnFieldMapped(Sender: TObject; AField: TField; ALayoutItem: TLayoutItem);

  Event triggered after the TAdvDBFormPanel has generated a layout item for a DB field found in the dataset. This allows dynamically altering the automatic generated layout item.

- OnLayoutCreated(Sender: TObject; Bounds: TRect);

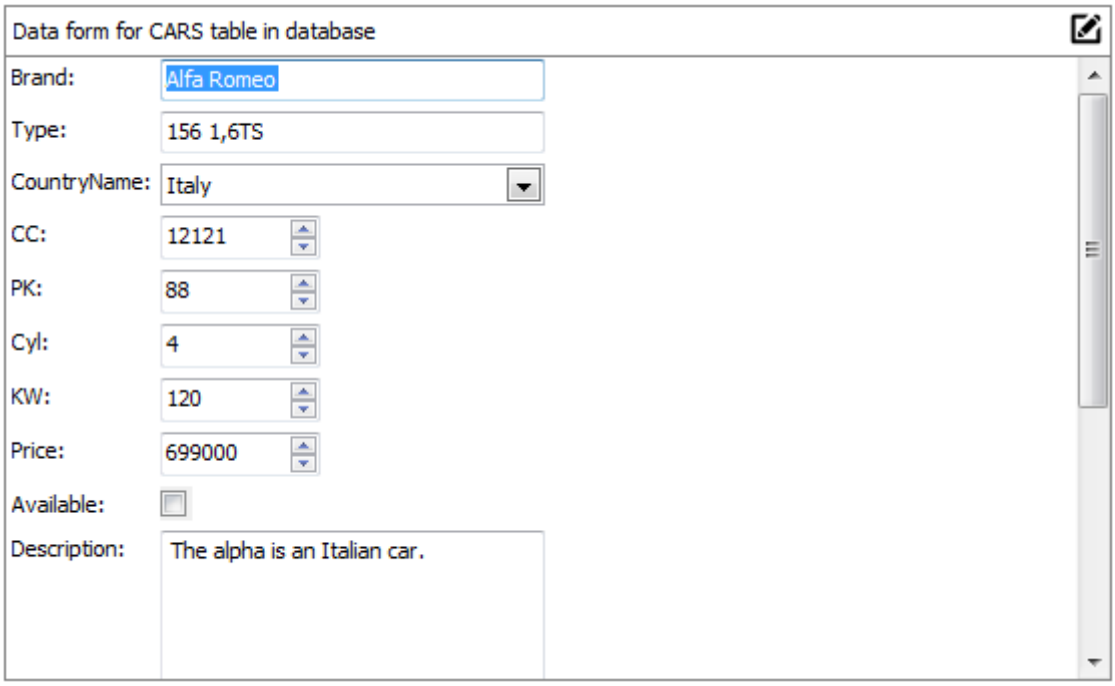
  Event triggered when the layout has been created, i.e. all DB controls for the layout were created. This returns the bounding rectangle that was needed for all DB controls and allows for adaption of the panel size to needed space.

TAdvDBFormBox architecture

TAdvDBFormBox is very similar to a TAdvDBFormPanel. The major difference is that it has an optional caption and a scrollable area.

Via the caption, it is possible to put the form in edit mode or browse mode via buttons in the top right corner.

The scrollable area ensures that regardless of the number of DB controls generated in the panel, the controls will always be reachable via the scrollbars.



TAdvDBFormBox properties

Compared to TAdvDBFormPanel, TAdvDBFormBox introduces two extra properties:

Caption

- Caption.Color: TColor : sets the start color of the gradient background of the caption.
- Caption.ColorTo: TColor : sets the end color of the gradient background of the caption.
- Caption.Font: TFont : sets the caption font.
- Caption.Height: integer : sets the caption height.
- Caption.HintCancel: string : sets the hint for the cancel button in the caption.
- Caption.HintEdit: string : sets the hint for the edit button in the caption.
- Caption.HintOK : string : sets the hint for the confirm button in the caption.
- Caption.ImageIndex: integer : sets the index of the image to optionally use in the caption.
- Caption.Text: string : sets the caption text.
- Caption.Visible: Boolean : sets whether the caption is visible or not.

ScrollArea

Sets the size of the scrollable area in the TAdvDBFormBox

- ScrollArea.Width: integer : width of the scroll area in pixels.
- ScrollArea.Height: integer : height of the scroll area in pixels.

TAdvDBFormPanel / TAdvFormBox custom class mapping

By default, the TAdvDBFormPanel and TAdvDBFormBox map the various DB field types that exist to mostly TMS DB-aware controls when these exist and offer more features than the standard VCL DB controls.

The default class mapping as such is from datacontrol type to class:

dcEdit: TDBAdvEdit

dcSpinEdit: TDBAdvSpinEdit

dcCheckBox: TDBCheckBox

dcDateTime: TAdvDBDateTimePicker

dcDate: TAdvDBDateTimePicker

dcTime: TAdvDBDateTimePicker

dcMemo: TDBMemo

dcComboBox: TDBComboBox

dcRadioGroup; TDBRadioGroup

dcLabel: TDBAdvLabel

dcImage: TDBAdvGDIPPicture

dcRichEdit: TDBRichEdit

dcMaskEdit: TDBAdvMaskEdit

dcLookupComboBox: TDBLookupComboBox

This mapping can be customized though.

It is accessible via methods AdvDBFormPanel.Layouter.ClearClassMap / AdvDBFormPanel.Layouter.AddClassMap and for the TAdvDBFormBox via AdvDBFormBox.DBForm.Layouter.ClearClassMap / AdvDBFormBox.DBForm.Layouter.AddClassMap.

Example:
To choose the TMS TAdvDBLookupComboBox as lookup editor, call:

```
// Remove any existing mappings for dcLookupComboBox
AdvDBFormPanel.Layouter.RemoveClassMap(dcLookupComboBox);
```

```
// Add new mapping
AdvDBFormPanel.Layouter.AddClassMap(dcLookupComboBox, TAdvDBLookupComboBox);
```