TMS TAdvKanbanBoard
DEVELOPERS GUIDE

October 2020

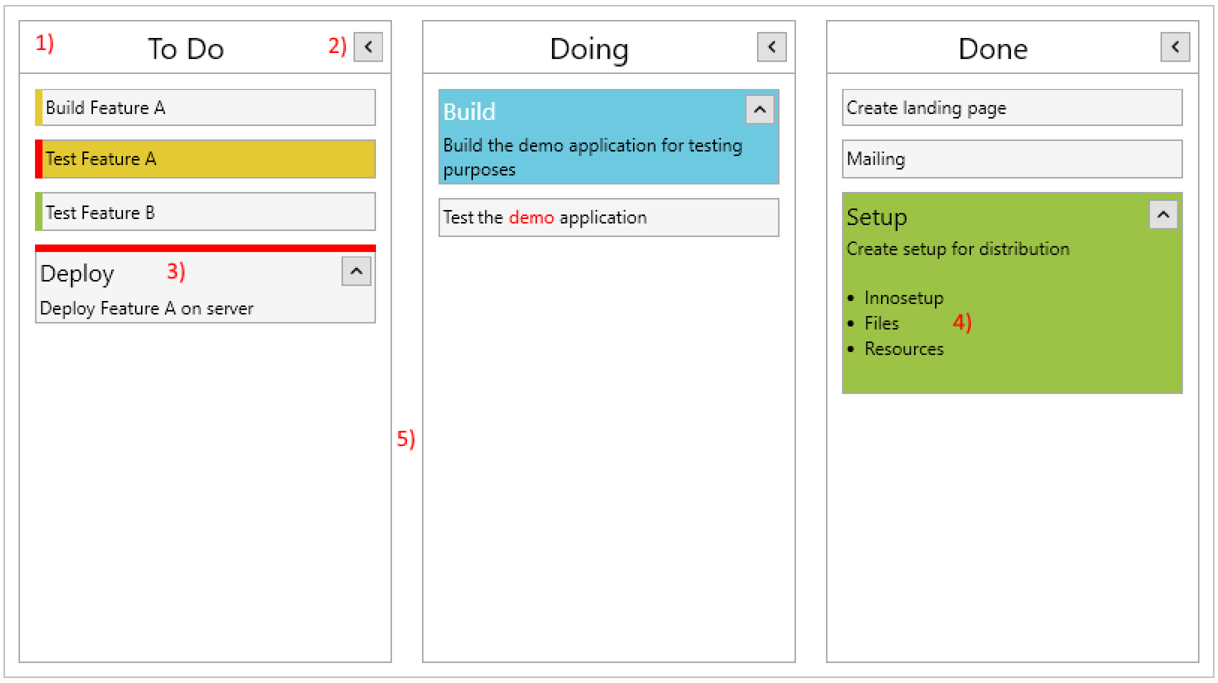
Index

Availability

Supported frameworks and platforms
- VCL Win32/Win64

Supported IDE's
- Delphi XE7 and C++ Builder XE7 or newer releases

Hierarchy



1) Each column in the TMS AdvKanbanBoard has the ability to show a header with optional filtering, sorting and expand/collapse functionality. It can display text with HTML formatting capabilities.

2) Sample of enabling expand/collapse in a column.

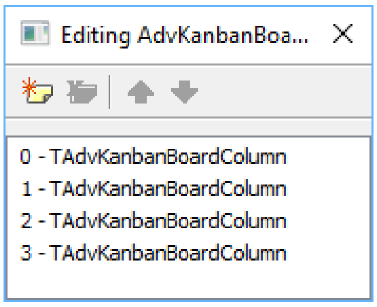
3) A TMS AdvKanbanBoard item, which has a title and a description (text). Both parts of the item support HTML formatted text as well as expand/collapse functionality. Optionally the item can be marked with an area (red line at the top of the item).

4) Sample of HTML formatting capabilities in the form of an unordered list.

5) Margins and spacing options for configuring the appearance of the columns. Each column has a separate width property, and on kanbanboard level columns can additionally be configured via the ColumnsAppearance property.

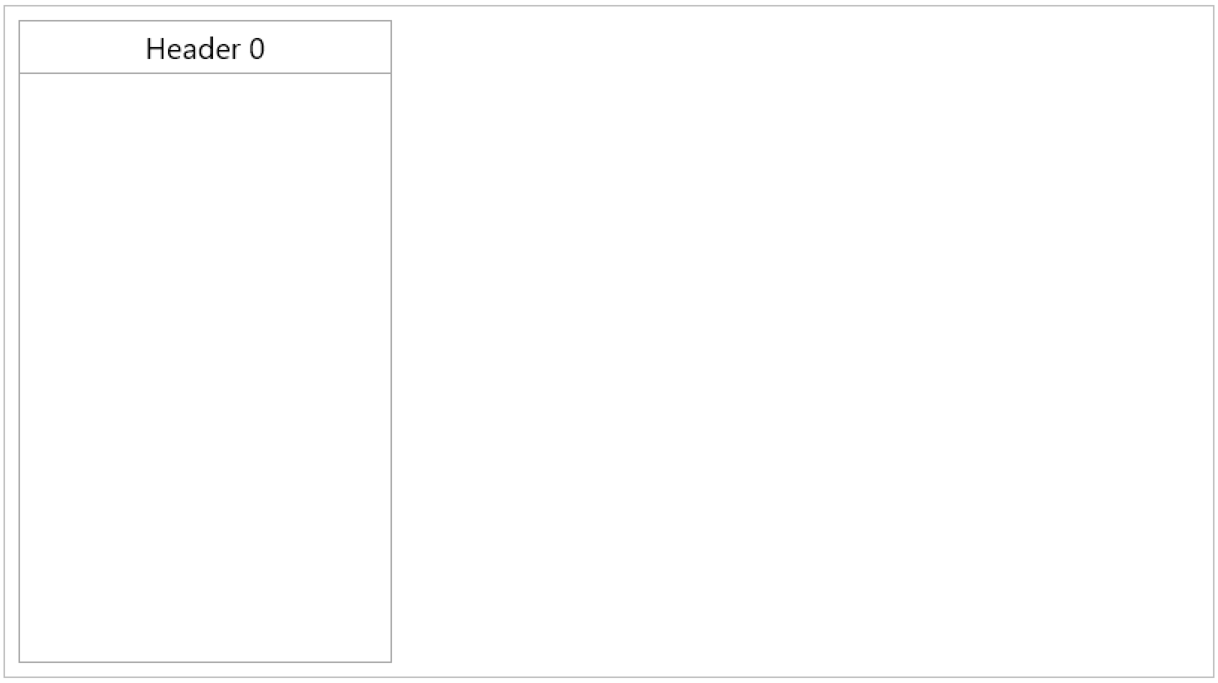
Adding Columns

When starting with the kanbanboard component you'll notice the initialization of 3 columns with a series of items as a preview. This is done to give you, as a developer, an idea on how this component visualizes columns and items, and which features it exposes. The columns and items can be easily removed by clicking on the component, and then clearing the columns in the object inspector, via the columns collection editor.

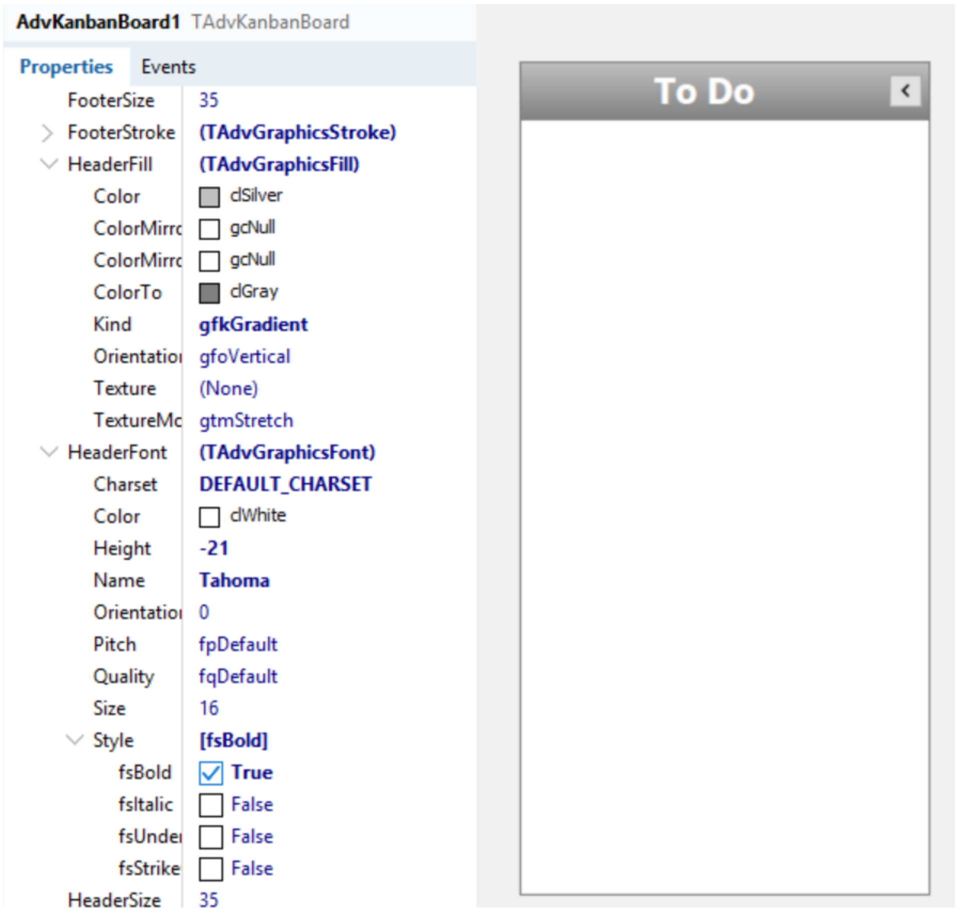


0 - TAdvKanbanBoardColumn
1 - TAdvKanbanBoardColumn
2 - TAdvKanbanBoardColumn
3 - TAdvKanbanBoardColumn

You'll then start with an empty kanbanboard instance. Adding columns is as easy as removing them. Click on the new button to create an instance of TAdvKanbanBoardColumn. You'll notice the kanbanboard is updated with a column, which has a header by default.



The "Header 0" value is ofcourse preset by the component, and this can be changed by setting the HeaderText property. Let's change this to TODO. Additionally, we would also like to change the header appearance, so the header fill and stroke properties can be used for this purpose. After changing these properties, you can expect to see something similar to the screenshot below. (Note that the kanbanboard has a default ColumnAppearance that is applied to all columns. For a specific column appearance, UseDefaultAppearance property needs to be set to False).


AdvKanbanBoard1 TAdvKanbanBoard

Properties Events

FooterSize	35
> FooterStroke	(TAdvGraphicsStroke)
∨ HeaderFill	(TAdvGraphicsFill)
Color	▨ clSilver
ColorMirro	☐ gcNull
ColorMirro	☐ gcNull
ColorTo	▨ clGray
Kind	gfkGradient
Orientatio	gfoVertical
Texture	(None)
TextureMc	gtmStretch
∨ HeaderFont	(TAdvGraphicsFont)
Charset	DEFAULT_CHARSET
Color	☐ clWhite
Height	-21
Name	Tahoma
Orientatio	0
Pitch	fpDefault
Quality	fqDefault
Size	16
∨ Style	[fsBold]
fsBold	✓ True
fsItalic	☐ False
fsUnder	☐ False
fsStrike	☐ False
HeaderSize	35

When the above settings need to be configured programmatically, below is a code snippet that demonstrates this.

```
var
  c: TAdvKanbanBoardColumn;
begin
  AdvKanbanBoard1.BeginUpdate;
  c := AdvKanbanBoard1.Columns.Add;
  c.UseDefaultAppearance := False;
  c.HeaderFill.Color := clSilver;;
  c.HeaderFill.ColorTo := clGray;
  c.HeaderFill.Kind := gfkGradient;
  c.HeaderFont.Color := clWhite;
  c.HeaderFont.Size := 16;
  c.HeaderFont.Style := [TFontStyle.fsBold];
  c.HeaderText := 'To Do';
  AdvKanbanBoard1.EndUpdate;
end;
```

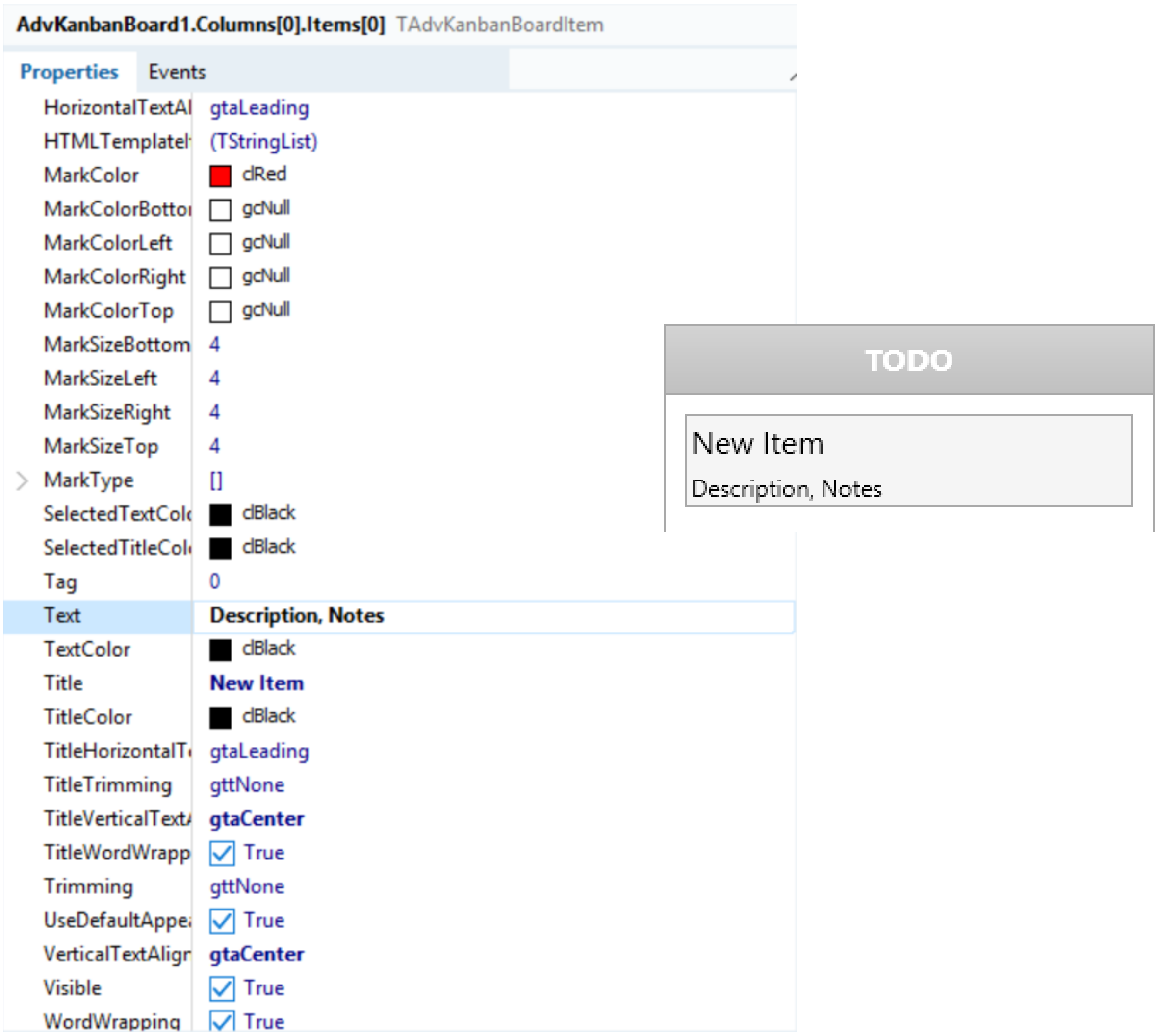
Before we add items to our column, we want to add 2 additional columns. These columns will help us in the next chapters, adding and moving items. Add 2 new colums, “IN PROGRESS” and “DONE”, in the same way as the “TODO” column. After adding these 2 columns, we can proceed to add the items.

Adding Items

Each column has an items collection that can be used to add kanbanboard items. Each item represents a task or todo item depending on the way the kanbanboard is configured and for which purpose it is used.

Continuing from our previous sample, we want to add a new TODO item.
Adding a new item can be done both at designtime or programmatically (runtime). To add a new item at designtime, select the column in columns collection editor via the object inspector. Click on the items property, which will open the items collection editor. Clicking on the new button will add a new instance of TAdvKanbanBoardItem to the list. You'll also notice that an item has visually been added to the column. Configuring this item is similar to configuring the column. You can set properties on item level that will change the title and description (text), and you can also change the appearance of the item.

The item also has a UseDefaultAppearance property that is used to override the default appearance coming from the ItemsAppearance property at kanbanboard level. Below is a sample that demonstrates adding an item both at designtime and at runtime.


AdvKanbanBoard1.Columns[0].Items[0] TAdvKanbanBoardItem

Properties | Events

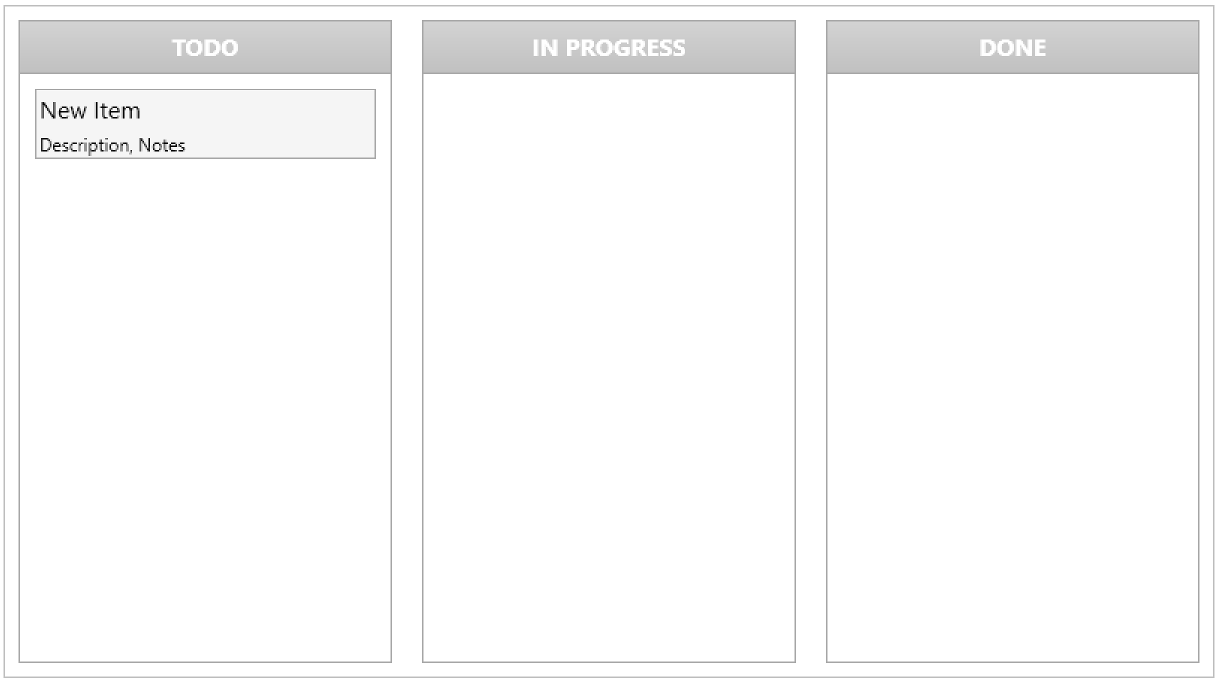
HorizontalTextAl	gtaLeading
HTMLTemplate	(TStringList)
MarkColor	🟥 clRed
MarkColorBotto	☐ gcNull
MarkColorLeft	☐ gcNull
MarkColorRight	☐ gcNull
MarkColorTop	☐ gcNull
MarkSizeBottom	4
MarkSizeLeft	4
MarkSizeRight	4
MarkSizeTop	4
MarkType	[]
SelectedTextCol	⬛ clBlack
SelectedTitleCol	⬛ clBlack
Tag	0
Text	**Description, Notes**
TextColor	⬛ clBlack
Title	**New Item**
TitleColor	⬛ clBlack
TitleHorizontalT	gtaLeading
TitleTrimming	gttNone
TitleVerticalText	**gtaCenter**
TitleWordWrapp	✓ True
Trimming	gttNone
UseDefaultAppe	✓ True
VerticalTextAlig	**gtaCenter**
Visible	✓ True
WordWrapping	✓ True

TODO

New Item
Description, Notes

```
var
  c: TAdvKanbanBoardColumn;
  it: TAdvKanbanBoardItem;
begin
  AdvKanbanBoard1.BeginUpdate;
  c := AdvKanbanBoard1.Columns[0];
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';
  AdvKanbanBoard1.EndUpdate;
end;
```

After adding columns and items, you'll end up with a Kanban board similar to the one shown in the screenshot below.



The full code is

```
var
  c: TAdvKanbanBoardColumn;
  it: TAdvKanbanBoardItem;
begin
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Color := clSilver;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.ColorTo := clGray;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Kind := gfkGradient;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Color := clWhite;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Size := 16;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Style :=
[TFontStyle.fsBold];
  AdvKanbanBoard1.BeginUpdate;
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'IN PROGRESS';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText  := 'DONE';

  AdvKanbanBoard1.EndUpdate;
end;
```

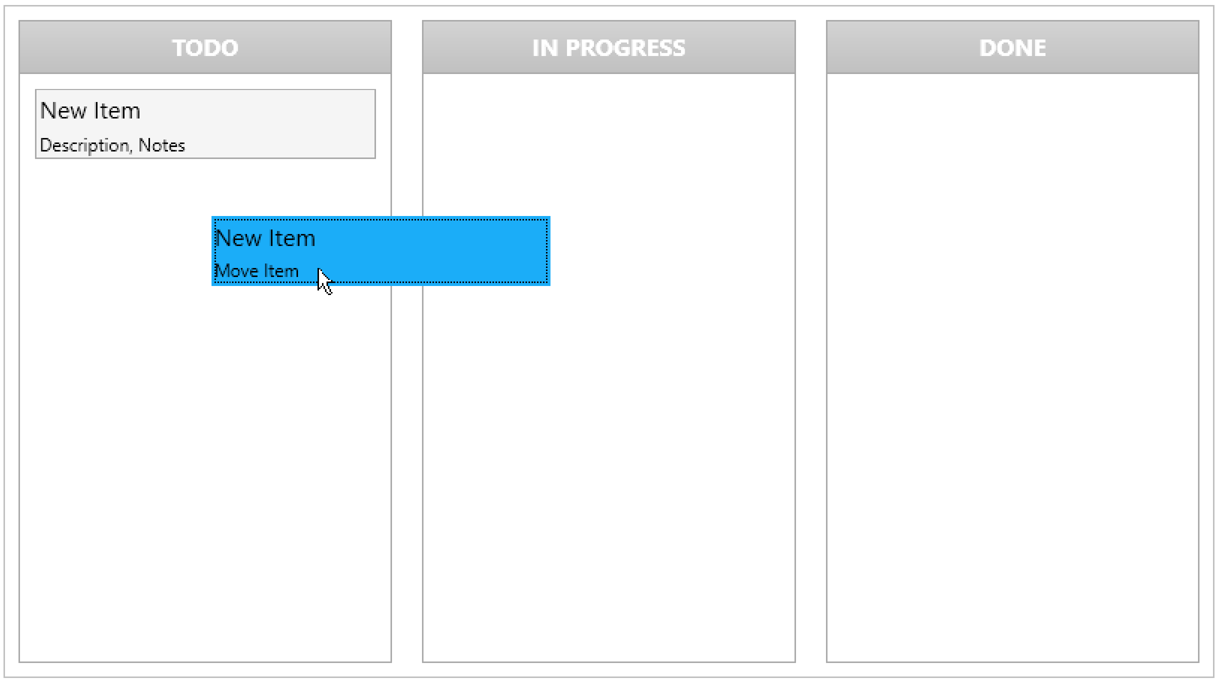
Moving Items

Items can be moved between columns, either programmatically or by interaction. The Kanbanboard Interaction property has a DragDropMode property that can be used to move items between columns. You can select kbdmNone, kbdmMove or kbdmCopy. The kbdmCopy setting will do the same as the kbdmMove, but leave a copy of the selected item in the original column. This way, the item can be duplicated.

To start a moving operation, click or tap on the item that needs to be moved. Click/tap again and hold the mouse/finger down on the item. This way, the item can be moved between columns. Now release the item at the new column of choice. When an item is already added to the list, and the dragged item is released on an already existing item, the item will be inserted at that position. When releasing the item inside the column at the end, the item will be added at the end of the column.

For demonstrating the move operation, we add a new item first:

```
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Move Item';
```

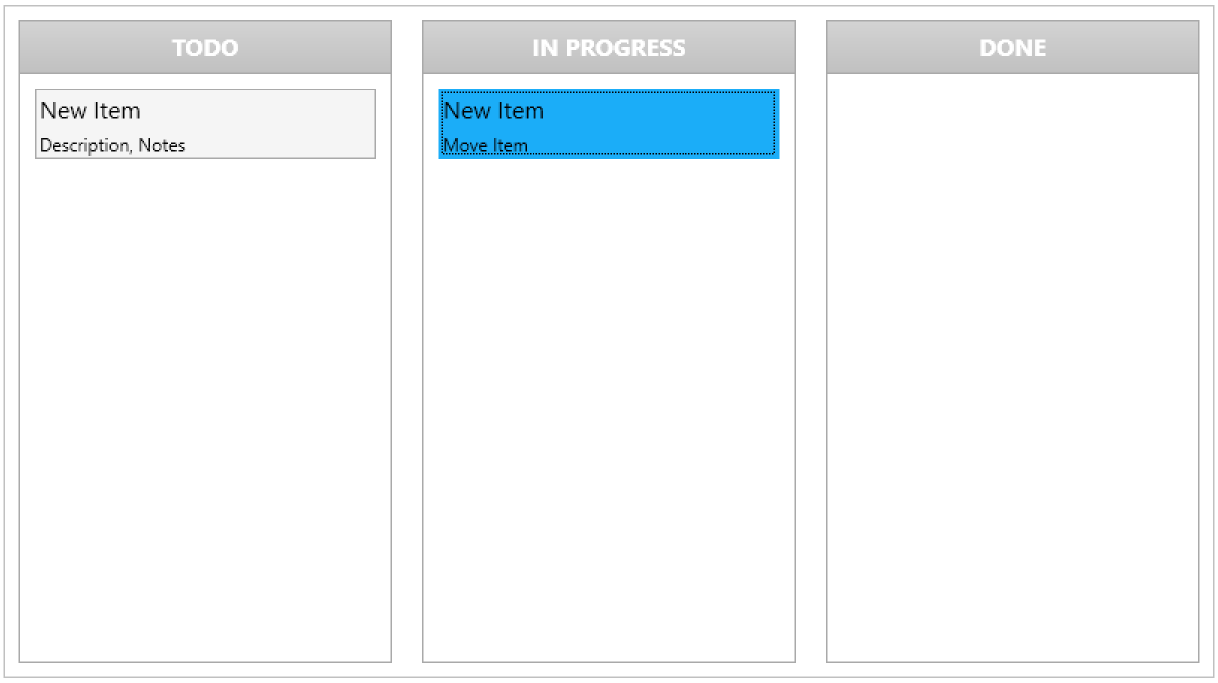


Release the left mouse button / finger in the “IN PROGRESS” column, to move the item to the new column. If we would have selected kbdmCopy instead of kbdmMove, the item will be duplicated, and there would be an item in both the “TODO” and “IN PROGRESS” column.

Of course this can also be done programmatically. To move an item to a specific column, use the MoveItem method at TAdvKanbanBoardItem level.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  it: TAdvKanbanBoardItem;
begin
  it := AdvKanbanBoard1.SelectedItem;
  if Assigned(it) then
    it.MoveItem(AdvKanbanBoard1.Columns[1]).SelectItem;
end;
```

The above code will retrieve the selected item, move it to the new column and the select the new item.

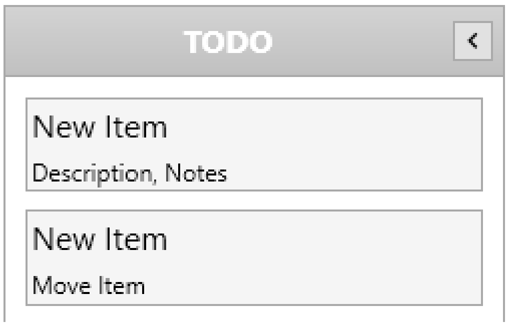


```
procedure TForm1.Button1Click(Sender: TObject);
var
  it: TAdvKanbanBoardItem;
begin
  it := AdvKanbanBoard1.SelectedItem;
  if Assigned(it) then
    it.MoveItem(AdvKanbanBoard1.Columns[1]).SelectItem;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  c: TAdvKanbanBoardColumn;
  it: TAdvKanbanBoardItem;
begin
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Color := clSilver;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.ColorTo := clGray;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Kind := gfkGradient;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Color := clWhite;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Size := 16;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Style :=
[TFontStyle.fsBold];
  AdvKanbanBoard1.BeginUpdate;
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Move Item';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'IN PROGRESS';
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText  := 'DONE';

  AdvKanbanBoard1.EndUpdate;
end;
```

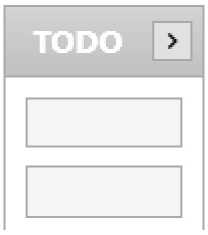
Moving items via interaction can be controlled with events. Sometimes, there is a certain requirement necessary to restrict items from begin moved or copied to a specific column. The OnBeforeDropItem and OnAfterDropItem events can be used to control this requirement.

Expand / Collapse Columns

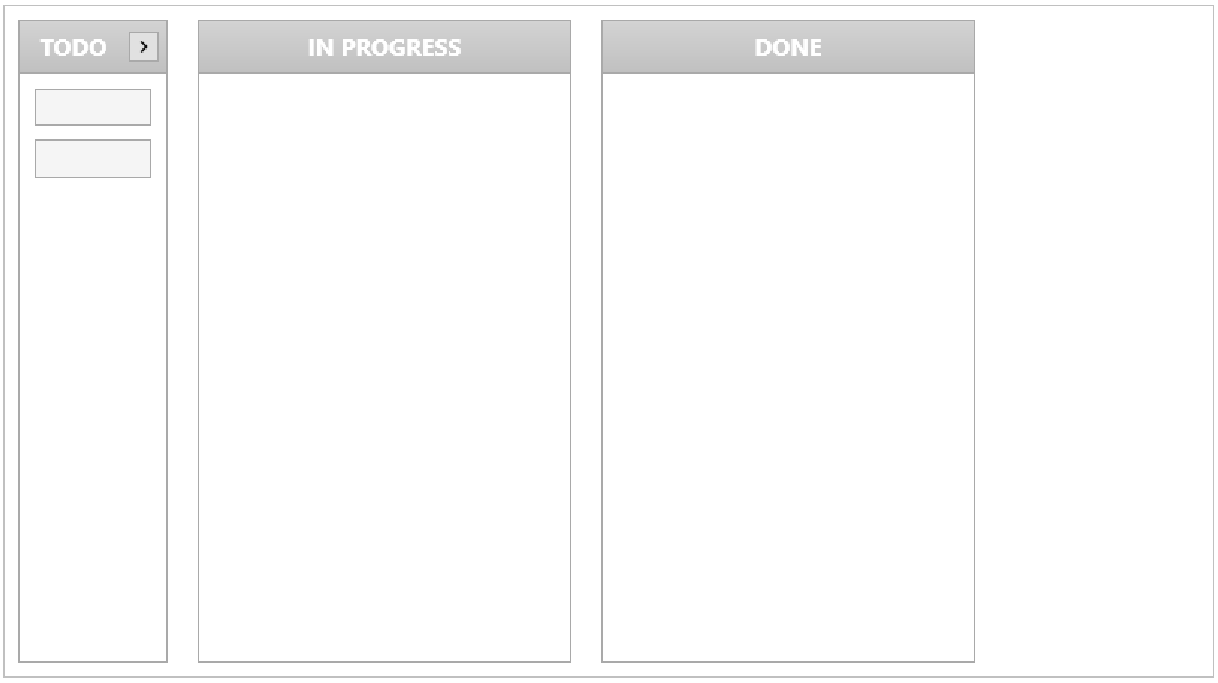
Each column has the ability to expand and collapse. By default the column is expanded. The Width property, that can be configured at TAdvKanbanBoardColumn level, is the expanded width. You can configure a CollapsedWidth as well which is the width that is used when a column is collapsed. To enable column collapsing, set the Expandable property to true. This way, an additional button is shown in the top-right corner of the header.



Clicking on the button will collapse the column, hiding the title and description of the item in a smaller collapsed state. This allows for an easier overview of the other columns, in case the kanbanboard has many columns that exceed the width of the visible area.



The column can also be collapsed / expanded at runtime, with the Expanded property at column level. Below is a sample that initializes the first column as collapsed.

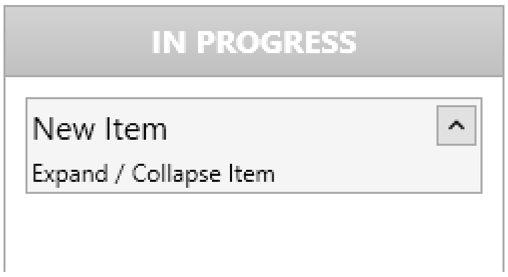


```
var
  c: TAdvKanbanBoardColumn;
  it: TAdvKanbanBoardItem;
begin
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Color := clSilver;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.ColorTo := clGray;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Kind := gfkGradient;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Color := clWhite;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Size := 16;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Style :=
[TFontStyle.fsBold];
  AdvKanbanBoard1.BeginUpdate;
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  c.Expandable := True;
  c.Expanded := False;

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Move Item';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'IN PROGRESS';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText  := 'DONE';

  AdvKanbanBoard1.EndUpdate;
end;
```

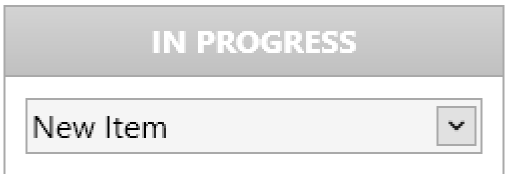
Expand / Collapse Items

In the same way as a column is collapsible / expandable, an item can be configured to support this kind of interaction as well. When accessing an item, either at designtime or at runtime. The Expandable property and Expanded property works the same way as the properties at column. The only difference is that the item is collapsed in height instead of in width and that the minimum size if fixed.

```
it := c.Items.Add;
it.Title := 'New Item';
it.Text := 'Expand / Collapse Item';
it.Expandable := True;
```



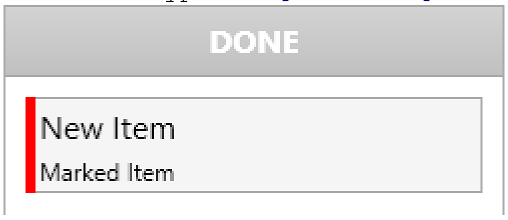
When clicking the button, the item will shrink in height and only show the title area.



Marking an item

An item can be marked, which, by default, shows a colored line at the mark position that can be configured via the MarkType and MarkColor property. The MarkType is a set, so that means that multiple values can be added. This way, you can add multiple colored mark lines. Each line can be colored separately. Additionally, a MarkSize property is available to set the size of the lines.

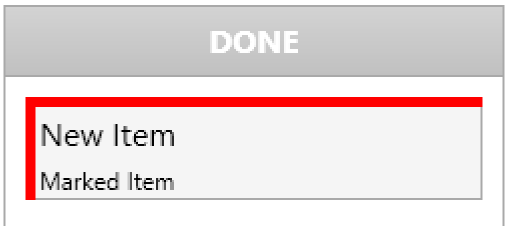
```
it := c.Items.Add;
it.Title := 'New Item';
it.Text := 'Marked Item';
it.MarkType := [kbmtLeft];
```



Changing the MarkType property to add more mark areas for drawing is shown in the following sample:

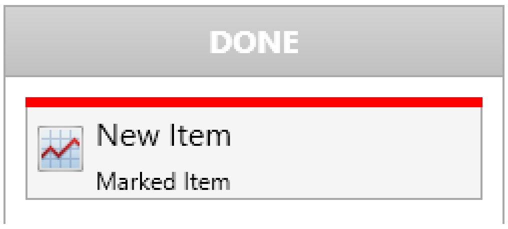
```
it := c.Items.Add;
it.Title := 'New Item';
```

```
  it.Text := 'Marked Item';
  it.MarkType := [kbmtLeft, kbmtTop];
```



Additionally, a custom mark can be applied. To do this, the OnItemCustomDrawMark event overrides the default drawing for each mark type. Below is a sample that increase the MarkSize to provide more space for drawing, and adds a bitmapcontainer for drawing a bitmap inside the left mark area. This way, multiple mark areas can be combined drawing either the default appearance or drawing custom graphics.

```
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Marked Item';
  it.MarkType := [kbmtLeft, kbmtTop];
  it.MarkSizeLeft := 32;

procedure TForm1.AdvKanbanBoard1ItemCustomDrawMark(Sender: TObject;
  AGraphics: TAdvGraphics; ARect: TRectF; AMarkType:
TAdvKanbanBoardMarkType;
  AColumn: TAdvKanbanBoardColumn; AItem: TAdvKanbanBoardItem);begin
  case AMarkType of
    kbmtLeft:
    begin
      AGraphics.PictureContainer := PictureContainer1;
      AGraphics.DrawBitmapWithName(ConvertToRect(ARect),
PictureContainer1.Items[0]);
    end;
    else
      AGraphics.DrawRectangle(ARect, gcrmNone);
  end;
end;
```



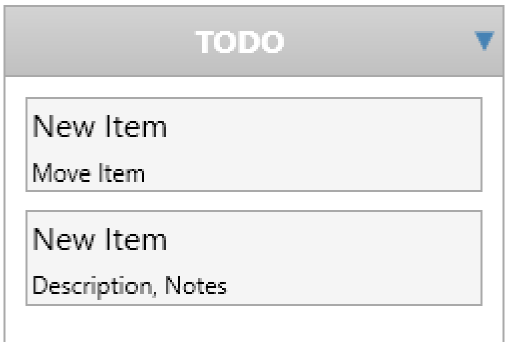
Sorting

Sorting is supported for each column in the kanbanboard. By default sorting is not enabled, but this can easily be enabled by using the Sorting property at column level. Setting the sorting property to gsmNormal or gsmNormalCaseSensitive. Let’s enable this on our first column:

```
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  c.Sorting := kbsNormal;
```

Clicking on the header of the column will start a sorting operation. By default the first sorting operation is descending. Programmatically this can be done with the following code:

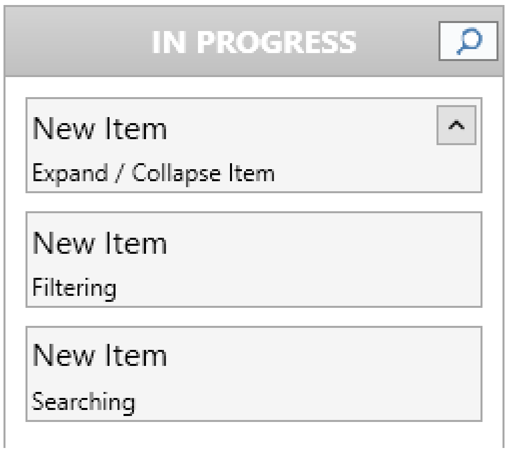
```
  c.Items.Sort(False, kbismAscending);
```



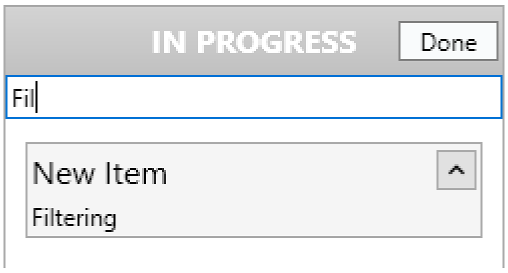
Filtering

Filtering can be applied to each column separately. To enable filtering via interaction, you can set the ShowFilterButton property to true. Let's apply this on our second column and add another item:

```
  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'IN PROGRESS';
  c.ShowFilterButton := True;
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Filtering';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Searching';
```



Clicking or tapping on this button will show a filter edit box that can be used to filter items in the list. The filter is applied only to the Text property, the Title property is ignored.



Filtering can also be applied programmatically, without user interaction.

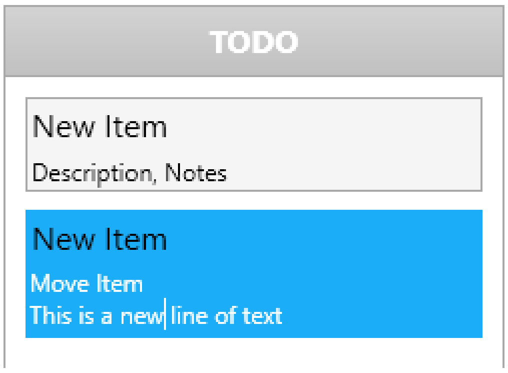
```
  f := c.Filter.Add;
  f.Condition := 'Fil*';
  c.ApplyFilter;
```

The above filter operation will programmatically hide the items that don't match the filter condition. The filter also has options to filter with or without case sensitivity.

Editing

An item description / text area can be edited. This is done via a memo inplace editor. To enable editing set the property Editing to true at kanbanboard interaction level.

```
  AdvKanbanBoard1.Interaction.Editing := True;
```



To start editing, click on an item and press either the F2 or Enter key or click/tap on the selected item. This immediately starts inplace editing. Pressing Escape will cancel editing and to accept changes, press F2 or Enter key again, or tap outside the item. After a succesful editing operation, the OnUpdateItemText event is called.

Sample code

The sample code accompanying the above chapters is shown below.

```
unit Unit10;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs;
```

```
type
  TForm1 = class(TForm)
    AdvKanbanBoard1: TAdvKanbanBoard;
    Button1: TButton;
    PictureContainer1: TPictureContainer;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure AdvKanbanBoard1ItemCustomDrawMark(Sender: TObject;
      AGraphics: TAdvGraphics; ARect: TRectF;
      AMarkType: TAdvKanbanBoardMarkType; AColumn: TAdvKanbanBoardColumn;
      AItem: TAdvKanbanBoardItem);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form10: TForm1;

implementation

{$R *.fmx}

procedure TForm1.Button1Click(Sender: TObject);
var
  it: TAdvKanbanBoardItem;
begin
  it := AdvKanbanBoard1.SelectedItem;
  if Assigned(it) then
    it.MoveItem(AdvKanbanBoard1.Columns[1]).SelectItem;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  c: TAdvKanbanBoardColumn;
  it: TAdvKanbanBoardItem;
begin
  AdvKanbanBoard1.BeginUpdate;
  AdvKanbanBoard1.Columns.Clear;

  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Color := clSilver;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.ColorTo := clGray;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFill.Kind := gfkGradient;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Color := clWhite;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Size := 16;
  AdvKanbanBoard1.ColumnsAppearance.HeaderFont.Style :=
[TFontStyle.fsBold];

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'TODO';
  c.Sorting := kbsNormal;
  c.Items.Sort(False, kbismAscending);

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Description, Notes';
  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Move Item';

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText := 'IN PROGRESS';
  c.ShowFilterButton := True;

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Expand / Collapse Item';
  it.Expandable := True;

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Filtering';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Searching';

  f := c.Filter.Add;
  f.Condition := 'F*';

  c.ApplyFilter;

  c := AdvKanbanBoard1.Columns.Add;
  c.HeaderText  := 'DONE';

  it := c.Items.Add;
  it.Title := 'New Item';
  it.Text := 'Marked Item';
  it.MarkType := [kbmtLeft, kbmtTop];
  it.MarkSizeLeft := 32;

  AdvKanbanBoard1.Interaction.Editing := True;

  AdvKanbanBoard1.EndUpdate;
end;

procedure TForm1.AdvKanbanBoard1ItemCustomDrawMark(Sender: TObject;
  AGraphics: TAdvGraphics; ARect: TRectF; AMarkType:
TAdvKanbanBoardMarkType;
  AColumn: TAdvKanbanBoardColumn; AItem: TAdvKanbanBoardItem);
begin
  case AMarkType of
    kbmtLeft:
    begin
      AGraphics.PictureContainer := PictureContainer1;
      AGraphics.DrawBitmapWithName(ConvertToRect(ARect),
PictureContainer1.Items[0]);
    end;
    else
      AGraphics.DrawRectangle(ARect, gcrmNone);
  end;
end;
```

```
end.
```

Database Adapter

The kanbanboard supports loading and manipulating items from a dataset. The TAdvKanbanBoardDatabaseAdapter component can be used for this purpose. Data-enabling is easy and only requires the following 3 steps:

1) Prepare your dataset to match fields / data according to the requirements of the kanbanboard database adapter. The field requirements are:

   Column: Integer
   DBKey: string
   Text: string
   Title: String

2) Attach the DataSource to the kanbanboard database adapter.
3) Attach the kanbanboard database adapter to the Adapter property of the kanbanboard.
4) Set Active to True.

Additionally, the kanbanboard database adapter supports mapping other fields to the item as well. This is done via the OnFieldsToItem and OnItemToFields. These 2 events also have the column equivalent in case specific settings need to be done at column level.

Below is a sample with a TClientDataSet that has been filled programmatically with a few items, hooked up to a kanbanboard database adapter and connected to a kanbanboard.

```
unit UDemo;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Data.DB, Datasnap.DBClient,
  AdvCustomControl, AdvCustomScrollControl, AdvKanbanBoard,
AdvCustomComponent, AdvKanbanBoardDatabaseAdapter;
type
  TForm1 = class(TForm)
    AdvKanbanBoard1: TAdvKanbanBoard;
    AdvKanbanBoardDatabaseAdapter1: TAdvKanbanBoardDatabaseAdapter;
    ClientDataSet1: TClientDataSet;
    DataSource1: TDataSource;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure AdvKanbanBoardDatabaseAdapter1(Sender: TObject;
      AFields: TFields; AItem: TAdvKanbanBoardItem);
    procedure Button1Click(Sender: TObject);
    procedure AdvKanbanBoardDatabaseAdapter1(Sender: TObject;
      AColumn: Integer; AFields: TFields; AColumnField: TField);
    procedure AdvKanbanBoardDatabaseAdapter1(Sender: TObject;
      AFields: TFields; AColumnField: TField; var AColumn: Integer;
      var AAccept: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form34: TForm1;

implementation

{$R *.dfm}

type
  TAdvKanbanBoardOpen = class(TAdvKanbanBoard);

procedure TForm1.Button1Click(Sender: TObject);
begin
  AdvKanbanBoardDatabaseAdapter1.Active := not
AdvKanbanBoardDatabaseAdapter1.Active;
end;

procedure TForm1.FormCreate(Sender: TObject);
var
  I: Integer;
begin
  DataSource1.DataSet := ClientDataSet1;

  ClientDataSet1.FieldDefs.Add('Id', ftString, 255);
  ClientDataSet1.FieldDefs.Add('Column', ftString, 10);
  ClientDataSet1.FieldDefs.Add('Title', ftString, 10);
  ClientDataSet1.FieldDefs.Add('Text', ftString, 255);
  ClientDataSet1.FieldDefs.Add('Color', ftLongWord);
  ClientDataSet1.CreateDataSet;

  AdvKanbanBoard1.Adapter := AdvKanbanBoardDatabaseAdapter1;
  AdvKanbanBoardDatabaseAdapter1.Item.AutoIncrementDBKey := False;
  AdvKanbanBoardDatabaseAdapter1.Item.DataSource := DataSource1;
  AdvKanbanBoardDatabaseAdapter1.Item.DBKey := 'Id';
  AdvKanbanBoardDatabaseAdapter1.Item.Title := 'Title';
  AdvKanbanBoardDatabaseAdapter1.Item.Text := 'Text';
  AdvKanbanBoardDatabaseAdapter1.Item.Column := 'Column';

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -4;
  ClientDataSet1.FieldByName('Title').AsString := 'Miami';
  ClientDataSet1.FieldByName('Text').AsString := 'Dialy shoot at the
beach';
  ClientDataSet1.FieldByName('Color').AsLongWord := gcOrange;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'New York';
  ClientDataSet1.FieldByName('Text').AsString := 'Shoe model';
  ClientDataSet1.FieldByName('Color').AsLongWord := clGray;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'New York';
  ClientDataSet1.FieldByName('Text').AsString := 'Shoe model';
  ClientDataSet1.FieldByName('Color').AsLongWord := clGray;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'Barcelona';
  ClientDataSet1.FieldByName('Text').AsString := 'Audition for photoshoot';
  ClientDataSet1.FieldByName('Color').AsLongWord := clGray;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -4;
  ClientDataSet1.FieldByName('Title').AsString := 'TV Ad';
  ClientDataSet1.FieldByName('Text').AsString := 'Advertisement for
toothpaste';
  ClientDataSet1.FieldByName('Color').AsLongWord := clWhite;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'Barcelona';
  ClientDataSet1.FieldByName('Text').AsString := 'Meet with Daniel Harris
for audition';
  ClientDataSet1.FieldByName('Color').AsLongWord := clWhite;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'Clothes';
  ClientDataSet1.FieldByName('Text').AsString := 'New clothes line
presentation in Milan';
  ClientDataSet1.FieldByName('Color').AsLongWord := clOlive;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'Photoshoot';
  ClientDataSet1.FieldByName('Text').AsString := 'Photoshoot for bikini
magazine';
  ClientDataSet1.FieldByName('Color').AsLongWord := clSkyBlue;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -3;
  ClientDataSet1.FieldByName('Title').AsString := 'Catwalk';
  ClientDataSet1.FieldByName('Text').AsString := 'Catwalk in Paris';
  ClientDataSet1.FieldByName('Color').AsLongWord := clFuchsia;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -2;
  ClientDataSet1.FieldByName('Title').AsString := 'TV Ad';
  ClientDataSet1.FieldByName('Text').AsString := 'Dinner with friends at
the seafood restaurant while shooting a new advertisement';
  ClientDataSet1.FieldByName('Color').AsLongWord := clFuchsia;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -2;
  ClientDataSet1.FieldByName('Title').AsString := 'Catwalk';
  ClientDataSet1.FieldByName('Text').AsString := 'Catwalk in Barcelona';
  ClientDataSet1.FieldByName('Color').AsLongWord := clFuchsia;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -2;
  ClientDataSet1.FieldByName('Title').AsString := 'Test shoot';
  ClientDataSet1.FieldByName('Text').AsString := 'Test shoot at the market
in Phuket';
  ClientDataSet1.FieldByName('Color').AsLongWord := clOlive;

  ClientDataSet1.Insert;
  ClientDataSet1.FieldByName('Id').AsString := TGuid.NewGuid.ToString;
  ClientDataSet1.FieldByName('Column').AsInteger := -2;
  ClientDataSet1.FieldByName('Title').AsString := 'Test shoot 2';
  ClientDataSet1.FieldByName('Text').AsString := 'Second Test shoot at the
market in Phuket';
  ClientDataSet1.FieldByName('Color').AsLongWord := clOlive;

  ClientDataSet1.Post;

  AdvKanbanBoard1.Interaction.Editing := True;
end;

procedure TForm1.AdvKanbanBoardDatabaseAdapter1ColumnToFields(
  Sender: TObject; AColumn: Integer; AFields: TFields; AColumnField:
TField);
begin
  AColumnField.AsInteger := AColumn - 4;
end;

procedure TForm1.AdvKanbanBoardDatabaseAdapter1FieldsToColumn(
  Sender: TObject; AFields: TFields; AColumnField: TField; var AColumn:
Integer;
  var AAccept: Boolean);
begin
  AColumn := AColumn + 4;
end;

procedure TForm1.AdvKanbanBoardDatabaseAdapter1FieldsToItem(Sender:
TObject; AFields: TFields; AItem: TAdvKanbanBoardItem);
var
  c: TColor;
begin
  c := AFields.FieldByName('Color').AsLongWord;
  if c <> gcNull then
  begin
    AItem.Color := c;
    AItem.UseDefaultAppearance := False;
  end
  else
    AItem.UseDefaultAppearance := True;
end;

end.
```

Important methods, properties and events

Properties

Columns	The collection of columns shown in the kanbanboard.
Columns[Index]→CollapsedWidth	The width of the column in collapsed state.
Columns[Index]→Expandable	Allows changing the column expanded or collapsed state via interaction.
Columns[Index]→Expanded	A property to toggle between expanded and collapsed state of the column.
Columns[Index]→Fill	The background fill of the column.
Columns[Index]→Filter	The filter of the column, that can be used to apply a filter operation to show or hide specific items.
Columns[Index]→FooterFill	The appearance of the footer area.
Columns[Index]→FooterFont	The font of the footer text.
Columns[Index]→FooterHorizontalTextAlign	The horizontal alignment of the footer text.
Columns[Index]→FooterStroke	The border of the footer area.
Columns[Index]→FooterText	The text of the footer.
Columns[Index]→FooterTrimming	The trimming of the footer text.
Columns[Index]→FooterVerticalTextAlign	The vertical alignment of the footer text.
Columns[Index]→FooterVisible	Shows or hides the footer.
Columns[Index]→FooterWordWrapping	Enables or disables word-wrapped text inide the footer.
Columns[Index]→HeaderFill	The appearance of the header area.
Columns[Index]→HeaderFont	The font of the header text.
Columns[Index]→HeaderHorizontalTextAlign	The horizontal alignment of the header text.
Columns[Index]→HeaderStroke	The border of the header area.
Columns[Index]→HeaderTrimming	The trimming of the header text.
Columns[Index]→HeaderVerticalTextAlign	The vertical alignment of the header text.
Columns[Index]→HeaderVisible	Shows or hides the header.
Columns[Index]→HeaderWordWrapping	Enables or disables word wrapped text inside the header.
Columns[Index]→Items	The collection of items inside the column.
Columns[Index]→Items[Index]→Bitmap	The bitmap loaded from a file, or assigned from a stream shown at the left side of the item.
Columns[Index]→Items[Index]→BitmapName	The bitmap loaded from a bitmap container shown at the left side of the item.

Columns[Index]→Items[Index]→Color	The color of the item in case UseDefaultAppearance is set to False
Columns[Index]→Items[Index]→DisabledTextColor	The text color of the item in disabled state in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→DisabledTitleColor	The title color of the item in disabled state in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→Enabled	Enables/disables the item.
Columns[Index]→Items[Index]→Expandable	Allows expanding/collapsing the item.
Columns[Index]→Items[Index]→Expanded	Property to trigger programmatic expanding/collapsing of an item.
Columns[Index]→Items[Index]→Height	The height of an item. By default the height is -1 which is used to apply auto-sizing.
Columns[Index]→Items[Index]→HorizontalTextAlign	The horizontal alignment of the text in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→HTMLTemplateItems	A list of name and value pairs to be used in combination with the HTMLTemplate property at ItemsAppearance level.
Columns[Index]→Items[Index]→MarkColor	The default color of the mark areas of an item.
Columns[Index]→Items[Index]→MarkColorBottom	The color of the bottom mark area.
Columns[Index]→Items[Index]→MarkColorLeft	The color of the left mark area.
Columns[Index]→Items[Index]→MarkColorRight	The color of the right mark area.
Columns[Index]→Items[Index]→MarkColorTop	The color of the top mark area.
Columns[Index]→Items[Index]→MarkSizeBottom	The size of the bottom mark area.
Columns[Index]→Items[Index]→MarkSizeLeft	The size of the left mark area.
Columns[Index]→Items[Index]→MarkSizeRight	The size of the right mark area.
Columns[Index]→Items[Index]→MarkSizeTop	The size of the top mark area.
Columns[Index]→Items[Index]→MarkType	The type of mark areas that will be shown in the item.
Columns[Index]→Items[Index]→SelectedTextColor	The color of the text in selected state in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→SelectedTitleColor	The color of the title in selected state in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→Text	The description/text of the item.
Columns[Index]→Items[Index]→TextColor	The color of the text of the item in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→Title	The title of the item.
Columns[Index]→Items[Index]→TitleColor	The color of the title of the item in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→TitleHorizontalTextAlign	The horizontal alignment of the title in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→TitleTrimming	The trimming of the title in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→TitleVerticalTextAlign	The vertical aligment of the title in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→TitleWordWrapping	The wordwrapping of the title in case UseDefaultAppearance is set To False.

Columns[Index]→Items[Index]→Trimming	The trimming of the text in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→UseDefaultAppearance	Overrides the default appearance configured via the ItemsAppearance property at kanbanboard level.
Columns[Index]→Items[Index]→VerticalTextAlign	The vertical alignment of the text in case UseDefaultAppearance is set to False.
Columns[Index]→Items[Index]→Visible	The item visibility.
Columns[Index]→Items[Index]→WordWrapping	The wordwrapping of the text in case UseDefaultAppearance is set to False.
Columns[Index]→ShowFilterButton	Shows or hides the filter button inside the column header.
Columns[Index]→Sorting	Enables or disables sorting mode.
Columns[Index]→Stroke	The border of the column.
Columns[Index]→UseDefaultAppearance	Overrides the default appearance configured via the ColumnsAppearance property at kanbanboard level.
Columns[Index]→Visible	The column visibility.
Columns[Index]→Width	The width of the column in expanded state.
ColumnsAppearance	The overall columns appearance, applied when UseDefaultAppearance of the column collection item is set to True.
ColumnsAppearance→Fill	The background fill of the column.
ColumnsAppearance→FooterFill	The footer fill of the column.
ColumnsAppearance→FooterFont	The font of the footer text of the column.
ColumnsAppearance→FooterSize	The size of the footer area of the column.
ColumnsAppearance→FooterStroke	The border of the footer area of the column.
ColumnsAppearance→HeaderFill	The header fill of the column.
ColumnsAppearance→HeaderFont	The font of the header text of the column.
ColumnsAppearance→HeaderSize	The size of the header area of the column.
ColumnsAppearance→HeaderStroke	The border of the header area of the column.
ColumnsAppearance→Margins	The margins used around the column.
ColumnsAppearance→Spacing	The spacing used between columns.
ColumnsAppearance→Stroke	The border of the column.
Interaction	Various interaction properties for controlling the kanbanboard behaviour.
Interaction→AutoOpenURL	Automatically opens hyperlinks.
Interaction→DragDropMode	Enables moving or copying items to another column.
Interaction→Editing	Enables editing of the description/text area of the item.
Interaction→KeyboardEdit	Enables or disabled starting inplace editing via keyboard.
Interaction→MouseEditMode	Configures the way editing is started via the mouse or finger.
Interaction→MultiSelect	Enables multi-selection of items.
Interaction→SwipeBounceGesture	Enables or disables a bounce gesture

	when the scrolling list reaches the beginning or the end after swiping inside a column.
Interaction→TouchScrolling	Enables / disables scrolling via touch.
ItemsAppearance	The appearance of the items in applied to all items with UseDefaultAppearance set to True.
ItemsAppearance→DisabledFill	The fill of the item in disabled state.
ItemsAppearance→DisabledStroke	The border of the item in disabled state.
ItemsAppearance→Fill	The fill of the item.
ItemsAppearance→FixedHeight	The height of the item in case the HeightMode is set to fixed.
ItemsAppearance→Font	The font of the description/text of the item.
ItemsAppearance→Height	The height of the item.
ItemsAppearance→HeightMode	The height mode of the item.
ItemsAppearance→HTMLTemplate	The template used for rendering HTML formatted text inside an item. Needs to be combined with HTMLTemplateItems at item level.
ItemsAppearance→Margins	The margins used around an item.
ItemsAppearance→SelectedFill	The fill of the items in selected state.
ItemsAppearance→SelectedStroke	The border of the items in selected state.
ItemsAppearance→ShowFocus	Shows or hides a focus border round the selected item.
ItemsAppearance→Spacing	The spacing between items.
ItemsAppearance→Stroke	The border of the item.
ItemsAppearance→TitleFont	The font of the title of the item.

Methods

Columns[Index]→AddItem(AText: String = ‘’);	Adds a new item to the column.
Columns[Index]→ApplyFilter;	Applies the filter of the column set with the Filter property.
Columns[Index]→ClearSelection;	Clears the selection of a column.
Columns[Index]→CopyItem(AItem: TAdvKanbanBoardItem; AToColumn: TAdvKanbanBoardColumn; AIndex: Integer = -1): TAdvKanbanBoardItem;	Copies an item to a specific column at a specific index. When the index is -1, the item will be placed at the end of the list.
Columns[Index]→DisableInteraction;	Disables interaction with the column.
Columns[Index]→EnableInteraction;	Enables interaction with the column.
Columns[Index]→GetSelectedItems: TAdvKanbanBoardItemArray;	Gets an array of selected items in case multi-select is true.
Columns[Index]→Items[Index]→CopyItem(AToColumn: TAdvKanbanBoardColumn; AIndex: Integer = -1): TAdvKanbanBoardItem;	Copies an item to a specific column at a specific index. When the index is -1, the item will be placed at the end of the list.
Columns[Index]→Items[Index]→IsSelected: Boolean;	Returns a Boolean if an item is selected or not.
Columns[Index]→Items[Index]→LoadFromString(AString: String);	Loads an item from a string.
Columns[Index]→Items[Index]→MoveItem(AToColumn: TAdvKanbanBoardColumn; AIndex: Integer = -1):	Moves an item to a specific column at a specific index. When the index is -1, the

TAdvKanbanBoardItem;	item will be placed at the end of the list.
Columns[Index]→Items[Index]→SaveToString(ATextOnly: Boolean = True);	Saves an item to a string.
Columns[Index]→Items[Index]→SelectItem;	Selects the item.
Columns[Index]→LookupItem(ALookupString: String; ACaseSensitive: Boolean; AAutoSelect: Boolean = False): TAdvKanbanBoardItem;	Looks up an item with a specific string. Optionally configurable are case sensitivity and auto-select when the item is found.
Columns[Index]→MoveItem(AItem: TAdvKanbanBoardItem; AToColumn: TAdvKanbanBoardColumn; AIndex: Integer = -1): TAdvKanbanBoardItem;	Moves an item to a specific column at a specific index. When the index is -1, the item will be placed at the end of the list.
Columns[Index]→RemoveFilter;	Removes an active filter from the column, but does not clear the filter data.
Columns[Index]→RemoveFilters;	Removes an active filter from the column and clears all filter data.
Columns[Index]→RemoveItem(AItem: TAdvKanbanBoardItem);	Removes an item from the column.
Columns[Index]→ScrollToItem(AItemIndex: Integer);	Scrolls to a specific item index.
Columns[Index]→SelectedItemCount: Integer;	Return the amount of selected items.
Columns[Index]→SelectItem(AItemIndex: Integer);	Selects an item with a specific item index.
Columns[Index]→SelectItems(AItemIndexes: TAdvKanbanBoardIntegerArray);	Selects an array of item indexes.
Columns[Index]→StartEditMode;	Starts inplace editing for the selected item.
Columns[Index]→StartFiltering;	Starts filtering.
Columns[Index]→StopEditMode;	Stops inplace editing for the selected item.
Columns[Index]→StopFiltering;	Stops filtering.
Columns[Index]→ToggleEditMode;	Toggles editing mode for the selected item.
Columns[Index]→XYToItem(X, Y: Single): TAdvKanbanBoardItem;	Returns the item at a specific X and Y coordinate.
Columns[Index]→XYToItemIndex(X, Y: Single): TAdvKanbanBoardItem;	Returns the index of an item at a specific X and Y coordinate.
FindItemWithDBKey(ADBKey: String): TAdvKanbanBoardItem;	Returns an item with a specific DBKey. Used in combination with an active database adapter connection.
SelectedItem: TAdvKanbanBoardItem;	Return the selected item.
SelectItem(AItem: TAdvKanbanBoardItem);	Selects an item.
XYToColumn(X, Y: Single): TAdvKanbanBoardColumn;	Returns the column at a specific X and Y coordinate.

Events

OnAfterApplyFilter	Event triggered after the filter is applied via interaction.
OnAfterDrawItem	Event triggered after an item is drawn.
OnAfterDrawItemText	Event triggered after the text / description of an item is drawn.
OnAfterDrawItemTitle	Event triggered after the title of an item is

	drawn.
OnAfterDropItem	Event triggered after an item is moved, reordered or copied.
OnBeforeApplyFilter	Event triggered before a filter is applied via interaction.
OnBeforeDrawItem	Event triggered before an item is drawn.
OnBeforeDrawItemText	Event triggered before the text / description of an item is drawn.
OnBeforeDrawItemTitle	Event triggered before the title of an item is drawn.
OnBeforeDropItem	Event triggered before an item is moved, reordered or copied.
OnColumnCollapse	Event triggered when a column is collapsed.
OnColumnExpand	Event triggered when a column is expanded.
OnCustomizeColumn	Event triggered when a column is added to the kanban board. This event allows further customization to the internal tableview representing a column.
OnDoneButtonClicked	Event triggered when the done button is clicked.
OnItemCollapse	Event triggered when an item is collapsed.
OnItemCompare	Event triggered when a column is sorted to allow custom sorting.
OnItemCustomDrawMark	Even triggered when an item mark is drawn. This event allows customization of the mark area.
OnItemExpand	Event triggered when an item is expanded.
OnSelectItem	Event triggered when an item is selected.
OnUpdateItemText	Event triggered when an item text / description is updated.