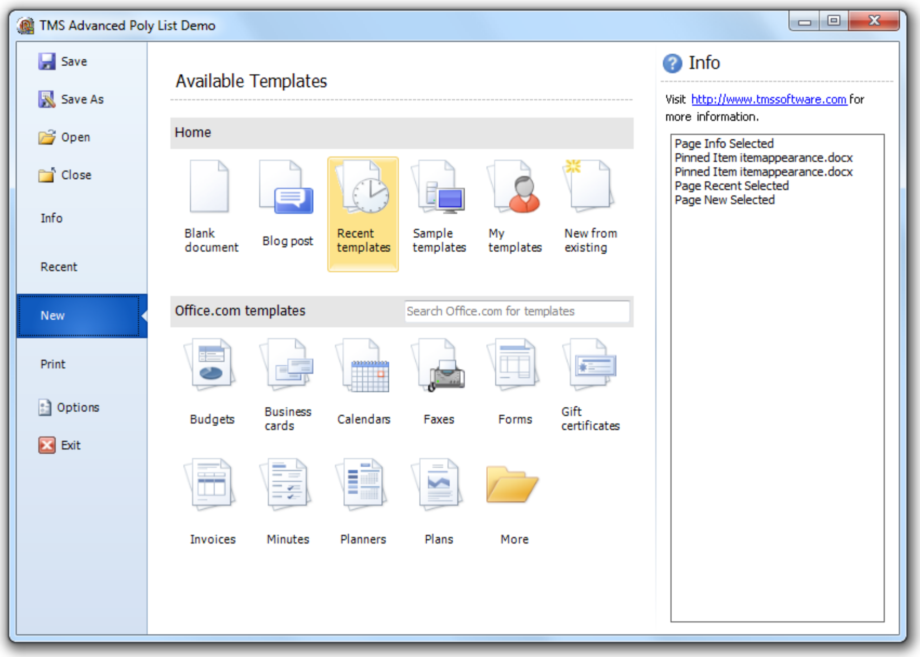
Introduction

Something that immediately draws the attention in Office 2010 is the new application menu. It is a very clearly laid out page of the various actions and options that can chosen & configured from there. On closer inspection, it is really packed with specific & very different user interface elements. To name just a few: small buttoned item, large buttoned item, dropdown list, check list, inplace edit controls, fixed items, formatted text items and so on.

TMS Poly List controls offers the capability to create such modern & fresh user interfaces to Delphi & C++Builder. A lot of thought went particularly into making such user interfaces easy and fast. We wanted to offer a solution that allows you to create a full & finished user interface in less than one hour. It should be a solution where you do not need an additional designer to work for hours to get everything look "right". Also, it is our understanding that such solution can be used in way more scenarios than just for building an application menu for a ribboned application and this is what the sample in this article demonstrates.



Visit http://www.tmssoftware.com for more information.

```
Page Info Selected
Pinned Item itemappearance.docx
Pinned Item itemappearance.docx
Page Recent Selected
Page New Selected
```

Architecture

Although the user interface elements are quite different in the application menu, many behaviors are common, like mouse hovering, mouse selection, focusing, .. as well as many style elements like normal state appearance, selected state appearance, hovered state appearance. Another thing that is common is that most items are treated as lists. Therefore, the basis of the architecture is a polymorph list of items descending from the TCustomItem class. The TCustomItem class knows of the minimum required common mouse, keyboard, selection, painting handling. From TCustomItem, we already created a wide range of ready to use classes:

- TButtonItem: Item with associated button, text
- TCheckItem: Item with checkbox, text
- TRadioItem: Item with radiobutton, text
- TGroupItem: Item with group caption and area that can be used to host another control
- TDropDownItem: Item with text and dropdown button. A new polymorph list of items can be shown from the dropdown
- THTMLItem: Item with capability to show HTML formatted text
- TImageItem: Item with image and text and various positioning methods for image versus text
- TLargeButtonedItem: Item with large button with image
- TImageTextItem: Item with text and image and different positioning for image versus text
- TTextItem: Item with text and optional description line
- TImageTextButtonItem: Item with associated button, text and image
- TWedgeItem: Variation of TTextItem with selection displayed with wedge on any of the 4 sides of the item
- TExpandableImageSectionItem: Item with image and text and button that can collaps or expand different items under the item
- THTMLSectionItem: Section item with HTML formatted text
- TImageSectionItem: Section item with image and text
- TImageTextButtonSectionItem: Section item with image, text and additional button

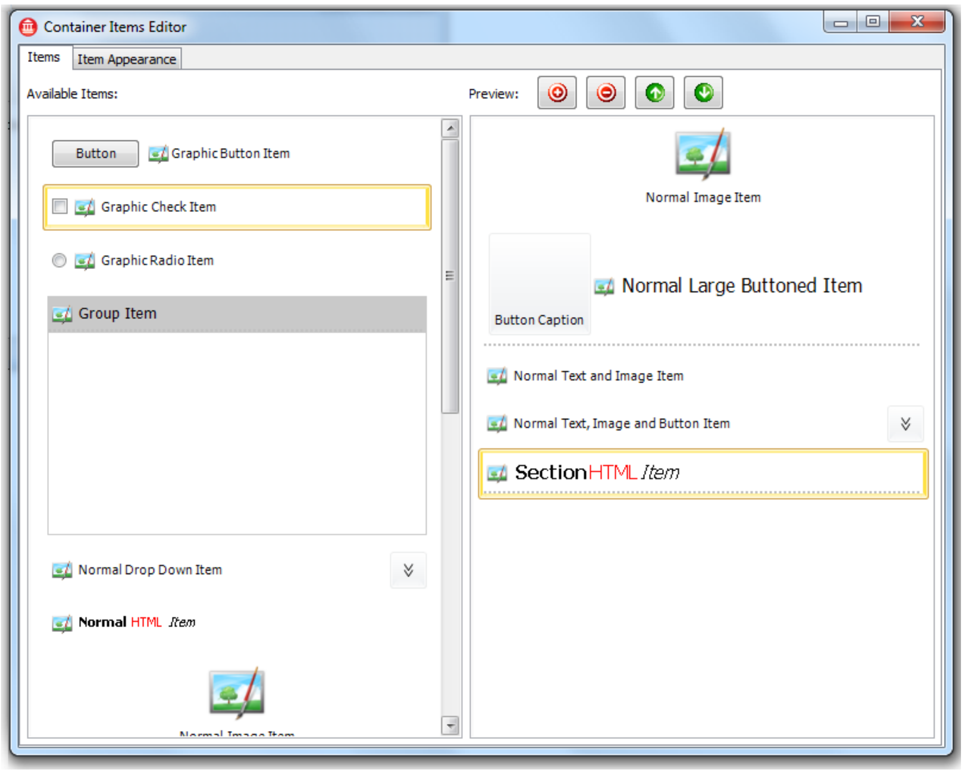
The design becomes really interesting knowing that with each item a control can be associated. Thus, a polymorph list control can in turn host inside an item or multiple items other polymorph lists or the TDropdownItem class allows to show a new polymorph list upon clicking its dropdown button. In addition, new classes descending from TCustomItem or any of the existing item classes can be created and registered. We have implemented each descending class in a separate unit. One of the advantages of this choice is that only the classes effectively used will be linked with your application. In the future we or you can create very specific custom item classes but these will not affect at all an application that is not using them.

Various list control types

Managing the polymorph list of items in memory is one thing, for actually displaying the items, we provide several controls. The base class that manages the list is TCustomItemsContainer. This holds & manages the list of items. We have provided 5 controls descending from this:

- TAdvVerticalPolyList: A scrollable list of items vertically (ie. under each other) organised
- TAdvHorizontalPolyList: A scrollable list of items horizontally (ie. next to each other) organised
- TAdvPolyList: A grid structure of items with configurable number of columns or rows
- TAdvPolyBox: A control in which items can be absolutely positioned
- TAdvPolyPager: A page control where a vertical list of items can be used to select a page

Using any of these controls is no more difficult than dropping it on the form, double click to start the designer, pick from a visual list of available item types and drag in the list preview. Click the item in the list preview and use the Object Inspector to set its properties.



This sample code snippet shows how an item can be added in code:

```
uses
   GDIPHTMLItem;
var
  hi: THTMLItem;
begin
  hi := THTMLItem(AdvVerticalPolyList1.AddItem(THTMLItem));
  hi.Caption := 'Hello <b>world</b>';
end;
```

The TMS PictureContainer to reuse images

In modern user interfaces, PNG images have almost become the norm. The PictureContainer component offers one central repository of images used in the application. It can be placed on a datamodule for example and reused in several forms to avoid inflating the DFM file with images that are used multiple times. The PictureContainer can host images of several sizes mixed contrary to a TImageList. Each image in the PictureContainer has an identifier name and with this name, the image can be chosen for an item. The TMS Poly List controls have all built-in support to work with the TMS PictureContainer component.

Putting it all together: creating a pager with sections

TAdvPolyPager is the control we can put at work to create a pagecontrol where the tabs to select a page are poly list items. This has the advantage that we can organize the pages in sections and display rich information on sections and tabs. To make all steps in setting up the control very clear for this article, everything is done in code. First of all, the TGDIPictureContainer is created and filled with images:

```
var
  pc:

  procedure LoadPicture(imagefile, imagename: string);
  begin
    with pc.Items.Add do
    begin
      Picture.LoadFromFile(imagefile);
      Name := imagename;
    end;
  end;

  pc := TGDIPPictureContainer.Create(self);
  // assign created PictureContainer to the PolyPager
  AdvPolyPager1.ListPictureContainer := pc;

  //this command loads one image from file
  //in the PictureContainer and sets the name
  LoadPicture('.\group_24.png','group');
```

Next, the TAdvPolyPager is filled with an expandable section item and two normal image items below this section item. Note that for the section item, a status indicator is set and that images from the PictureContainer are assigned via Item.ImageName:

```
// create and add item of class TExpandableImageSectionItem
eisi := TExpandableImageSectionItem(
        AdvPolyPager1.AddItem(TExpandableImageSectionItem));
eisi.Caption := 'Email items';
// set the status indicator text for the item
eisi.Status.Caption := '2';
eisi.Status.Visible := true;
eisi.ImageName := 'folder';  //set image name from PictureContainer

// create and add item of class TImageTextItem
iti := TImageTextItem(AdvPolyPager1.AddItem(TImageTextItem));
iti.Caption := 'Business account';
iti.ImageName := 'group';

iti := TImageTextItem(AdvPolyPager1.AddItem(TImageTextItem));
iti.Caption := 'Personal account';
iti.ImageName := 'admin';
```

Finally, the pages are created that are linked with the TImageTextItem instances in the list. The sections can be clicked to expand/collaps the TImageTextItem instances but it will not affect the active page. Clicking on the TImageTextItem instances will change the selected page. Therefore, we need to link an item to a page. This is done with the method AddAdvPolyPage that creates a page and links it to the item passed via the 2nd parameter:

```
AdvPolyPager1.AddAdvPolyPage('One',AdvPolyPager1.Items[1]);
```

Here a page is created and linked to the 2nd item in the list, ie. the first TImageTextItem. Access to the created pages is provided via:

```
AdvPolyPager1.AdvPolyPages[index]
```

Each page can have its own complex fill. To set a gradient background for each page, we can use:

```
AdvPolyPager1.AdvPolyPages[index].PageAppearance.Color := clWhite;
AdvPolyPager1.AdvPolyPages[index].PageAppearance.ColorTo := clYellow;
```

For this demo, we simply create a label for each page added:

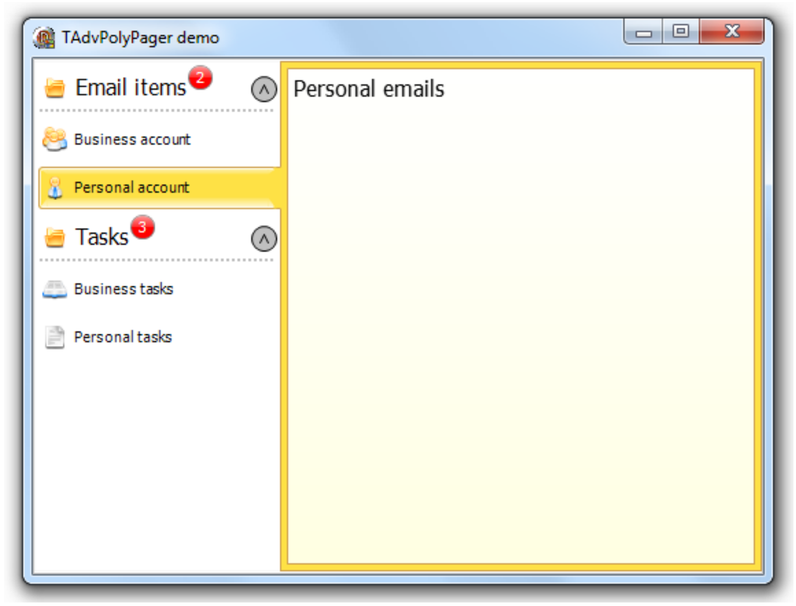
```
procedure CreateLabel(pp: TWinControl; Caption: string);
 begin
   lbl := TLabel.Create(pp);
   lbl.Parent := pp;
   lbl.Caption := Caption;
   lbl.Font.Size := 12;
   lbl.Top := 10;
   lbl.Left := 10;
 end;

 CreateLabel(AdvPolyPager1.AdvPolyPages[1], 'Personal emails');
```

Finally, the active page is initialized in code via:

```
 AdvPolyPager1.ActivePageIndex := 1;
```

The result obtained is:



Download a trial version of TMS Advanced Poly List at
http://www.tmssoftware.com/site/advpolylist.asp

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