WPForm - the native Delphi Form and Label Tool
Copyright (C) 2001 by Julian Ziersch Software, www.wptools.com
Info: support@wptools.de

Welcome to the first beta release of WPForm Version 2

WPForm is a component set to provide a tool to create labels and forms with ease. It has the ability to create lists as well. Its user interface is intuitive and cusomizable - you dont have to provide the eduser a tool which provides too many features to explain - in WPForm you can customize everything!

A few words about the new version:

WPForm Version 1.x already was very powerful but unfortunately much of its power was difficult to use. We tried to change that in Version 2 by splitting up objects and merging others. The internal object design as completely redone, there are fewer objects since it was not any longer necessary to limit the size of one unit.

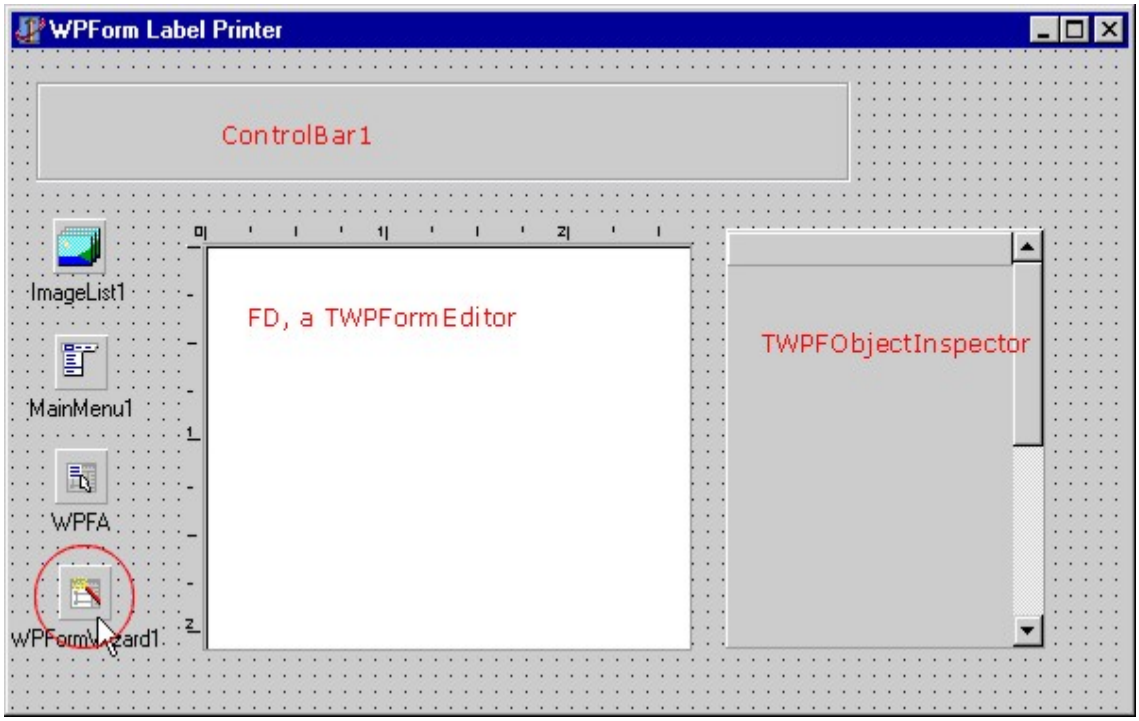
We have tried to make V2 as compatible to V1 as possible but still, for exiting projects there a many changes necessary. First of all the "report" property of a TWPFormEditor has been removed. Instead you have to place the new TWPReportEngine component on your form and attach it to the editor. This ReportEngine now also serves for Label printing which has the big advantage that you can preview the label printing output!
Version 2 is fully action based - it works with Delphi 3 but the power introduced through actions is not available there. The old TWPFormControl and buttons are still there (in the palette marked with a red x) but for your new projects better not use them. The actions work smoother.

The fileformat is fully backwards compatible, this means your old forms can be loaded without problems. We want to provide alternative fileformats in future, but the current 'WPF' format has been proven to be very fast and reliable.

A first project, a label printer

With other tools (no names here :) you have to use their ready-to-use designer application, you cannot adapt it to the actual needs, you cannot hide unnecessary features and words of all - you cannot fully integrate it into your application.
This is different with WPForm, we provide you with all controls to build a user interface and ready to use property (popup) dialogs. But there are also properties if you don't want to use the predefined controls.
The new - version 2 - action management (Delphi 4 or higher) lets you create the prototype of an application within a few minutes, toolbuttons and menues included!
How is this possible? The anser is the new TWPFormWizzard, a pseudo component which creates 89 actions, toolbuttons and menus when you click on it in the Delphi (4 or higher:) designer.

1) Creating the form for a new WPForm application



- this form can be found in Demos\1\LabelPrinter.DPR -

Here you see which controls we have created on the for. The following properties have to be set:

FD: This is the TWPFormEditor, we have renamed it to FD and modified its property "Page".

ControlBar1: RowHeight = 32

ImageList1: Button width and height=22
We loaded then loaded the file Resource\WPFButtons.BMP

MainMenu1: No changes

WPFA: This is a TWPFActionList but we have renamed it to WPFA.
property FormEdit has to point to the TWPFormEditor
property ImageList to ImageList1

WPFObjectInspector1: In its property 'Bands' we have created a 2 new band and set
in band 1: Caption=Graphics, Style=wpobiObjects
in band 2: Caption=Page, STyle=wobiPage
The property FD.PropertyInspector should point to WPFObjectInspector1

Now we create the TWPFormWizard control. We have set the properties
ActionList=WPFA
ButtonPanel=ControlBar1
MainMenu=MainMenu1

Please save the form!

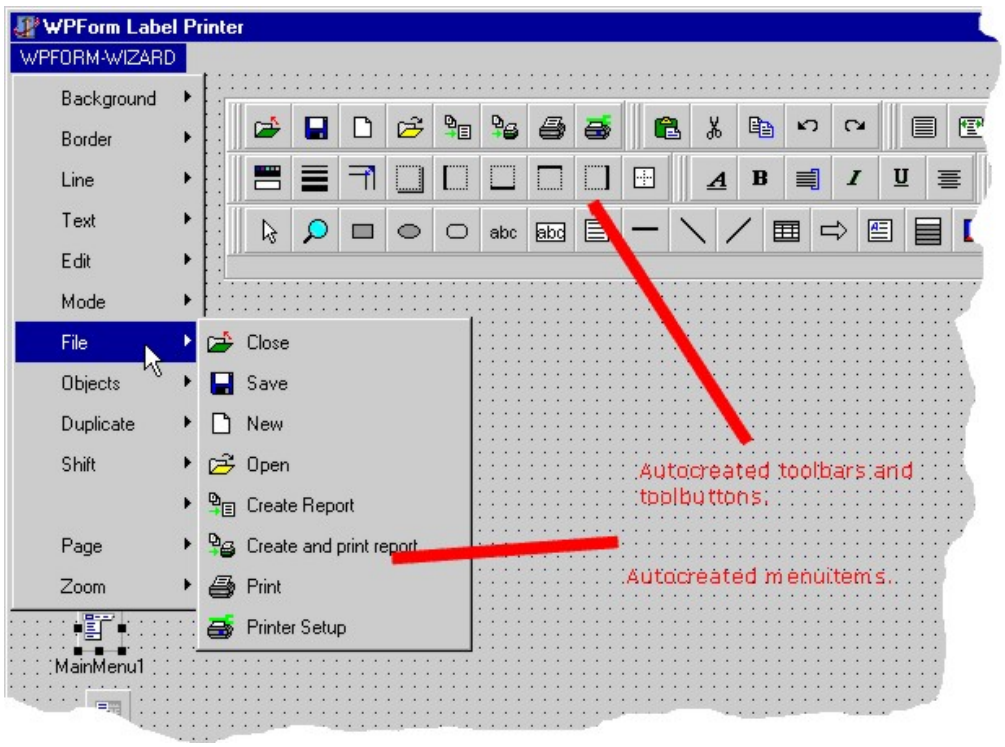
Now you can double click on the WPFormWizard and watch it doing its work .....

.....

After the TWPFormWizard has completed its job you will see the message

Further more the menu will be populated with a new item which contains all the newly created menu items. The ControlBar will be filled with new TToolBar elements which include many many new TToolButton elements. The Action property of all the new menu and button controls are already pointing to the correct Action in the TWPFActionList.

We have arranged the TToolBars a bit and the screen then looked like



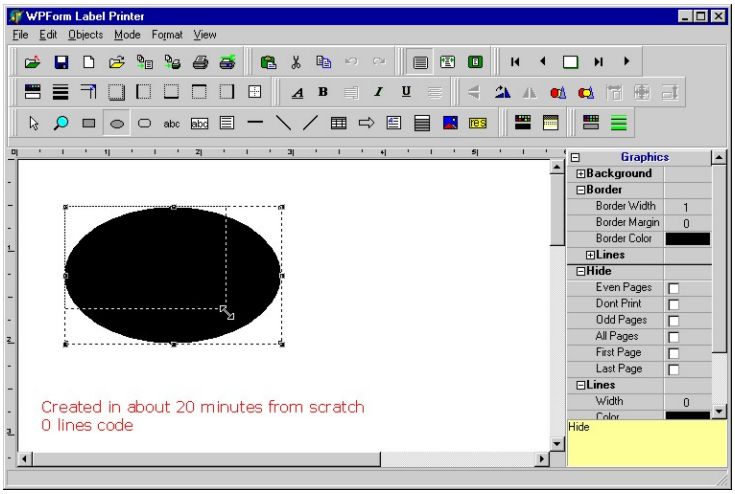
You can now use the Delphi menu editor to move the menus and submenus around and can further arrange the TToolBar controls. Of course you can delete menu items and tool buttons!

We have also modified the "Align" property of the controls and the deleted the TWPFormWizard.

Now you can compile and run the new project and try out the report designer.

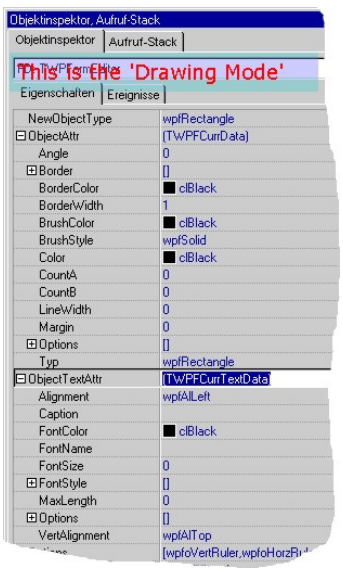
Note: It is possible to use the TWPFormWizard later - it will not create actions which already have been created. It will only create toolbars if 'ButtonPanel' has been set and no menus if property 'MainMenu' is not used. For security reasons it does not store the settings for its properties so you have to assign them before you can start it.

2) The new application in action:



It took only a few minutes to create this application, including its 80+ actions, menu items and buttons!

Now you can draw a form. To draw you have to select a tool (for example an ellipse or a rectangle) and then draw a frame on the paper. The new object will be created using the current 'DrawingMode'.
This DrawingMode is a feature not seen in the Delphi IDE or other reporting tools: if you change the line width or background style while no object is selected the 'DrawingMode' will me modified.
Of course, the drawing mode can be modified by code (property ObjectAttr and ObjectTextAttr) and also in the Delphi IDE. When you change it there you change the way the editor starts to work.



The editor has also a WorkMode (Select, Zoom or New Object) and an NewObjectType - which is the object which will be created when the user draws the frame.

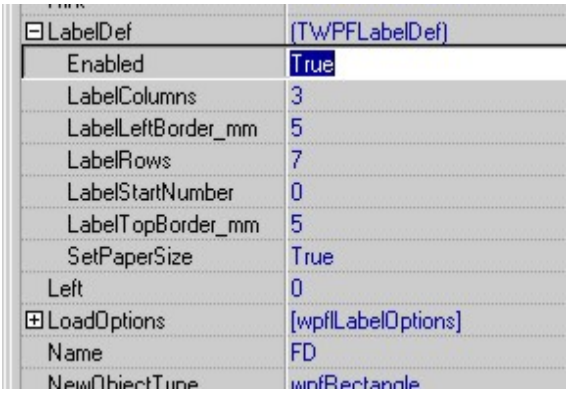
Since you can use the OnAfterAction event you can set the properties when the user switches the working mode.

3) Now we want to actually print labels !

Please create 2 new components on your form and modify the property LabelDef

Properties:

WPFReportEngine.FormEditor = FD
FD.PreviewDlg=WPFPreviewDlg1



The **TWPFReportEngine** is resonsible for the creation of a report file. Labels are also first created as a report file which makes it possible to preview the lables as they will be printed. A TWPFReportEngine is also used to load a report file (LoadFromFile) and
If you don't want to create this report file you can still print labels. But you have to do some coding for this and use the procedure PrintPageOnCanvas. If you label is just the size of a paper (you are in the lucky position that you have a special label printer!) you can simply execute Print to print the current form.

The TWPFPreviewDlg is actually a form with some toolbuttons and a regular TWPFReportEngine (to load a report file) and a TWPFormEditor to display it. It has the ability to start non modal and automatically delete the viewed files once the user closes the dialog.

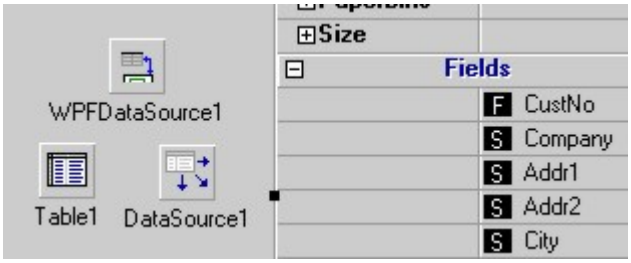
If you have set the property FD.PreviewDlg the buttons on the toolbar will already work.

Now you can also review the properties of the ObjectInspector. In the PageOptions property you can change which page properties are displayed, if you want to show or hide label obtions.

WPForm has the ability to store page information for each page in a multipage form, a feature which should not be used in a label printer. Using the property LabelStartNumber you can start in the middle of the page - this is very useful if a label sheet was already partly used!

4) Adding data access to the application:

We created the following 3 components:



In the TWPFObjectInspector we have created a new band and set is property CanDrag to true and its Style property to wpobiFields.

The WPFDataSource1 has its property FormEditor set to FD and the properties AutoLoadFieldList and AutoLoadData are both true.

If you are using several (wpf)datasources you can use the property TableAlias (both, in the ObjectInspector Band and in the TWPFDataSource) to differenciate between them. The field list in the object inspector will be only updated if the TableAlias matches. Its value will be displayedin the first column (which is empty in the example above).

Since we wan to drag fields from the object inspector to the form editor we have to create two event handlers for this control:

```
procedure TForm1.FDDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  if Source = WPFObjectInspector1.Nodes  then Accept := TRUE;
end;

procedure TForm1.FDDragDrop(Sender, Source: TObject; X, Y: Integer);
  var
  s: string;
begin
  if Source = WPFObjectInspector1.Nodes  then
  begin
    s := WPFObjectInspector1.Nodes.Selected.Text;
    FD.InsertField(s,  x, y);
    FD.RefreshData(FALSE);
  end;
end;
```

Now you can set the Table1 to active and run the application. It should be possible to drag fields from the obejct inspector to the TWPFormEdit and the value will be displayed there.
We have added a TDBNavigator to the toolbar and this makes it possible to scroll through the database.

But if you print the label only the current will be printed. The reason is that we havent set one property: TWPFReportEngine.MasteDataSource. If it points to the TWPFDataSource the reporter knows that all the data of this datasource has to be printed.
If you want to modify this operation check out the event TWPFReportEngine.OnReportState.

5) Modify the project to prints lists instead of labels

You have to switch off the properties in "LabelDef" - simply set Enabled to FALSE.

When you see the page create a 'ReportArea'. This is a control object which duplicates all the graphic controls found in its first row (you can change the size of this row!).

The TWPFReportEngine will by default (only if MasterDataSurce is set!) check if a page contains a report engine and if it does will not any longer advance to the next data record when the form has been completely processed but when one row of the area was completely created. The FieldName of the ReportArea has to be the same as the TableAlias used the in TWPFDataSource.

Using the OnReportState of the TWPFReportEngine you can change this behavior and use multiple and nested datasets in the form.