Как создать **FontListBox**?

unit FontListBox; interface uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls; type TFontListBox = class(TCustomListbox) Private { Private declarations } fFontSample : Boolean; fShowTrueType : Boolean; fCanvas : TControlCanvas; Procedure SetFontSample(B : Boolean); Procedure SetShowTrueType(B : Boolean); Protected { Protected declarations } Procedure CreateWnd; override; Public { Public declarations } Constructor Create(AOwner : TComponent); override; Destructor Destroy; override; Procedure DrawItem(Index : Integer; R : TRect; State : TOwnerDrawState); override; Published { Published declarations } { Properties } Property Fontsample : Boolean Read fFontSample Write SetFontSample; Property Align; Property Anchors; Property BiDiMode; Property BorderStyle; Property Color; Property Columns; Property Constraints; Property Cursor; Property DragCursor; Property DragKind; Property DragMode; Property Enabled; //Poperty ExtendedSelection; Does not exist in base class Property Font; Property Height; Property HelpContext; Property Hint; Property ImeMode; Property ImeName; Property IntegralHeight; Property Itemheight; Property Items; Property Left; Property MultiSelect; Property Name; Property ParentBiDiMode; Property ParentColor; Property ParentFont; Property ParentShowHint; Property PopupMenu; Property ShowTrueType : Boolean Read fShowTrueType Write SetShowTrueType; Property ShowHint; Property Sorted; Property Style; Property TabOrder; Property TabStop; Property TabWidth; Property Tag; Property Top; Property Visible; Property Width; { Events } Property OnClick; Property OnContextPopup; Property OnDblClick; Property OnDragDrop; Property OnDragOver; Property OnDrawItem; Property OnEndDock; Property OnEnter; Property OnExit; Property OnKeyDown; Property OnKeyPress; Property OnKeyUp; Property OnMeasureItem; Property OnMouseDown; Property OnMouseMove; Property OnMouseUp; Property OnStartDock; Property OnStartDrag; End; Procedure Register; Implementation {------------------------------------------------} Procedure Register; // Hello Begin RegisterComponents('Samples', [TFontListBox]); End; {------------------------------------------------} Procedure TFontListBox.SetShowTrueType(B : Boolean); Begin If B fShowTrueType then Begin fShowTrueType := B; Invalidate; // Force an update during design time End; End; {------------------------------------------------} Procedure TFontListBox.SetFontSample(B : Boolean); Begin If fFontSample B then Begin fFontSample := B; Invalidate; // Force an update during design time End; End; {------------------------------------------------} Destructor TFontListBox.Destroy; Begin fCanvas.Free; // Free the canvas Inherited Destroy; End; {------------------------------------------------} Constructor TFontListBox.Create(AOwner : TComponent); Begin Inherited Create(AOwner); // Initialize properties ParentFont := True; Font.Size := 8; Font.Style := []; Sorted := True; fFontSample := False; Style := lbOwnerDrawFixed; fCanvas := TControlCanvas.Create; fCanvas.Control := Self; ItemHeight := 16; fShowTrueType := False; End; {------------------------------------------------} procedure TFontListBox.CreateWnd; Begin inherited CreateWnd; Items := Screen.Fonts; //

```
Copy all the fonts to the ListBox.Items   ItemIndex := 0;        // Select first item   End;
{------------------------------------------------------}   procedure TFontListBox.DrawItem(Index : Integer; R
: TRect;     State : TOwnerDrawState);     Var     Metrics          : TTextMetric;     LogFnt
: TLogFont;     oldFont,newFont   : HFont;     IsTrueTypeFont   : Boolean;     fFontStyle       :
TFontStyles;     fFontName        : TFontName;     fFontColor       : TColor;     Begin
LogFnt.lfHeight := 10;   LogFnt.lfWidth := 10;   LogFnt.lfEscapement := 0;   LogFnt.lfWeight :=
FW_REGULAR;   LogFnt.lfItalic := 0;   LogFnt.lfUnderline := 0;   LogFnt.lfStrikeOut := 0;
LogFnt.lfCharSet := DEFAULT_CHARSET;   LogFnt.lfOutPrecision :=
OUT_DEFAULT_PRECIS;   LogFnt.lfClipPrecision := CLIP_DEFAULT_PRECIS;
LogFnt.lfQuality := DEFAULT_QUALITY;   LogFnt.lfPitchAndFamily := DEFAULT_PITCH or
FF_DONTCARE;   StrPCopy(LogFnt.lfFaceName,Items[Index]);   newFont :=
CreateFontIndirect(LogFnt);   oldFont := SelectObject(fCanvas.Handle,newFont);
GetTextMetrics(fCanvas.Handle,Metrics);   // Now we can check for TrueType   IsTrueTypeFont
:= True;   If (Metrics.tmPitchAndFamily and TMPF_TRUETYPE) = 0 then     IsTrueTypeFont :=
False;     Canvas.FillRect(R);   If fShowTrueType and IsTrueTypeFont then     Begin     // Save
font settings     fFontName := Canvas.Font.Name;     fFontStyle := Canvas.Font.Style;
fFontColor := Canvas.Font.Color;     // Set new font settings     Canvas.Font.Name := 'Times
new roman';     Canvas.Font.Style := [fsBold];     //Canvas.Font.Color := clBlack;
Canvas.TextOut(R.Left + 2,R.Top,'T');     If fFontColor  clHighLightText then
Canvas.Font.Color := clGray;     Canvas.TextOut(R.Left + 7,R.Top + 3,'T');     //Restore font
settings     Canvas.Font.Style := fFontStyle;     Canvas.Font.Color := fFontColor;
Canvas.Font.Name := fFontName;     End;     If fFontSample then     // The font will be drawn
the actual font     Canvas.Font.Name :=  Items[Index]   else     // The font will be drawn in
property "Font"     Canvas.Font.Name :=  Font.Name;     If fShowTrueType then
Canvas.TextOut(R.Left + 20,R.Top,Items[Index]) // Show TrueType   else
Canvas.TextOut(R.Left,R.Top,Items[Index]); // Don't show TrueType
SelectObject(fCanvas.Handle,oldFont);   DeleteObject(newFont);   End;
{------------------------------------------------------}   End.
```

По материалам http://delphi.3000.com