DataStructures of Word Processing Tools

Siehe auch Beispiel

unit WPDefs, WPTxtDef

Beachten Sie bitte, daß diese Definitionen sich mit der Weiterentwicklung der Komponenten stark ändern können!

Die Klassen:

TWPRTFText
  TWPRTFTextIO
    TWPRTFTextPaint
      TWPRTFTextInp

besitzen einen Zeiger

FirstPar : PTParagraph

der auf den ersten Absatz zeigt:

```
  PTParagraph = ^TParagraph;
  TParagraph   = record
    indentfirst, indentleft, indentright   : Integer;
    spacebefore, spaceafter, spacebetween : Integer;
    align             :TParAlign;
    prop              :TParProp;
    state             :TParState;
    Border            :TBorder;
    Tabs              :Longint;
    line              :PTLine;
    next,prev         :PTParagraph;
    end;

  PTLine = ^TLine;
  TLine = record
    Height   : Word;
    Base     : Word;
    y_start : LongInt;
    state         : TLinState;
    block         : (No, NoInverted, YesInverted, Yes, UnDef);
    pc            : Pchar;
    pa            : PTAttr;
    pmax,plen     : Word;
    lasttab, spacewid, spacerest : Word;
    next,prev     : PTLine;
    end;
```

Die Struktur sieht damit so aus

```
  TLinState   = Set of (listMustPaint,listAutoNewPage,listHyphenate);
```

```
TborderType= set of (BlEnabled,BlFinish,BlTop,BLBottom,
    BlLeft,BlRight,BLBox,BLDouble,BLDot,BLHair,BLBar,
          BlInnerEdge,BlOuterEdge,BlCenter,BlTabLines);
 PTBorder= ^TBorder;
 TBorder = record
      LineType : TBorderType;
      Thickness : Byte;
      Color : Byte;
      { LO nibble = Color 0..15, HI Nibble = space in mm   }
 end;

 TParProp     = Set of                    (paprNewPage,paprFrame,paprBullett);
TParState   = Set of                (pastMustReformat,pastInitialize);
TParAlign = (paralLeft, paralRight, paralCenter,                paralBlock);
```

Attributes

Structure to show how to access the data in a TWPRichText control

```
WPRichText1.Memo.FirstPar
--> TParagraph.prev = nil
    TParagraph .line
    --> TLine.prev = nil
        TLine.pc -> 'Text'
        TLine.pa -> Attributes (TAttr)
        TLine.next
        --> TLine.prev
            ....
    TParagraph .next
--> TParagraph.prev
    ...
```

Word Processing Tools Version 1.5.5

WordProcessing Tools für Delphi

1. WPRichText
2. DBWPRichText
3. WPRichTextLabel
4. WPToolBar
5. WPRuler
6. WPStatusbar
7. WPSpellCheckDlg
8. WPRtfStorage
9. WPPagePropDlg
10. WPParagraphPropDlg
11. WPParagraphBorderDlg

Lizenzbedingungen

WPTools wurde entwickelt von Julian Ziersch, München.
Stand der Dokumentation: 1.4.1996 (Version 1.51)

email: 100744.2101@compuserve.com

Word Processing Tools

LIZENZBEDINGUNGEN

MIT DER REGISTRIERUNG ERHALTEN SIE DIE LIZENZ DIE PROGRAMME DIE SIE MIT HILFE VON WPTOOLS ERZEUGEN WEITERZUGEBEN. EINE WEITERGABE IST JEDOCH DANN NICHT GESTATTET, WENN ES SICH BEI DEM PROGRAMM UM EIN MODUL (VCL,VBX,DLL) HANDELT, DAS VON DRITTER PARTEI GENUTZT WERDEN KANN.

SIE ERHALTEN DEN KOMPLETTEN QUELLCODE, DEN SIE ÄNDERN DÜRFEN, ABER AN NIEMANDEN WEITERGEBEN, DER NICHT SELBST WPTOOLS REGISTRIERT HAT.

OBWOHL ICH DIE VCL LANGE UND AUSGIEBIG DEBUGT HABE, MUß ICH AUF FOLGENDE HAFTUNGSBESCHRÄNKUNG BESTEHEN:

INDEM SIE DIE KOMPONENTE IN BENUTZUNG NEHMEN ERKLÄREN SIE, DASS SIE DEN AUTOR VON JEDWEDER HAFTUNG FÜR SCHÄDEN FREIZEICHNEN, DIE NICHT DURCH VORSATZ ODER GROBER FAHRLÄSSIGKEIT VERURSACHT WURDEN. AUSGESCHLOSSEN IST INSBESONDERE DIE HAFTUNG FÜR SCHÄDEN, DIE DURCH EIN FEHLVERHALTEN DES PROGRAMMS ODER AUFGRUND UNRICHTIGER DOKUMENTATION ENTSTANDEN SIND (MANGEL - UND MANGELFOLGE SCHÄDEN).

record TAttr

Siehe auch

unit WPDefs;

Achtung! Definitionen wird sich mit der Weiterentwicklung der Komponenten (abwärts kompatibel) ändern. Fußnoten und Textobjecte sind zur Zeit (28.3.1996) nur _geplant_!

```
WrtStyle=Set of (
            afsBold,
            afsItalic,
            afsUnderline,
            afsStrikeOut,
            afsProtected,
            afsHidden,
            afsHyperLink,
            afsIsMarked,
            afsIsObject,
            afsIsFootnote
            );

TDataType = (dtChar, dtObject, dtFootnote);

PTTextObj = ^TTextObj;
TTextObj = class
     Name          : string;
     tag           : Longint;
     prev, next : PTTextObj;
     DataType     : (dtGraphic, dtComponent, dtCalculation);
     data          : TObject;
  end;

PTFootnote = ^TFootnote;
TFootnote   = class
     Typ   : (fnChar, fnNumber);
     c      : Char;
     num   : Integer;
     text : String;
     prev,next : PTFootnote;
  end;



TAttr = record
     Width          : Word;
     Height         : Word;
     Style          : WrtStyle;
     case TDataType of
       dtChar      : (Base     : Word;
                      Font     : Byte;
                      Color    : Byte;
                      Size     : Byte;);
       dtObject   : (Data     : TTextObj);
```

```
   dtFootnote: (Footnote : TFootnote);
end;
```

DataStructures

TDBWPRichText

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TDBWPRichText = class(TWPCustomRichText)

TDBWPRichText erlaubt es, formatierten Text - beinahe beliebiger Länge - zu editieren. Damit haben Sie endlich die Möglichkeit riesige Textdatein in einer Datenbank zu speichern. Die Textdaten dürfen in einem Memo oder einem BLOB Feld vorliegen. TDBWPRichText lädt RTF oder normales ANSI Format, die Umschaltung kann sogar automatisch erfolgen.

TWPRichText

HyperLinkCursor
NoBlockMarking
OneClickHyperlink
MemoryFormat
CaretDisabled
FitToWindowHorz
SaveFormat
LoadFormat
Zooming
Resizing
BackgroundColor
Runtime only:
SelText,SelLength,SelStart
LastError
Lines
Modified
CPLineNr
CPLine
CPPosition
IsLastLine
SetModeControl

FontSelect
SaveAs Save
Load Insert
PrinterSetup Print!mPrint Print_xywh
ReplaceDialog FindDialog
DeleteSelection
PasteFromClipboard + Copy/CutToClipboard
LoadFromStream LoadFromStreamWithReader
SaveToStream SaveSelectionToStream
SaveToStreamWithWriter SaveSelectionToStreamWithWriter
SaveToFile LoadFromFile
Clear
LoadTemplate
WriteStatus
Find
FindNext
Spell_...
GetFontNr
GetFontName
Set_PageSize
Set_ParSpacing
Set_ParMargin
Change_ParSpacing
Set_ParBorderFinish
Set_ParBorder
PutParText,InsertParText,GetParText
InputText
GetTextLen GetTextBuf
SetSelTextBuf SetTextBuf
SetPosition

HyperLinkEvent
PrintFooter
PrintHeader
EditStateEvent
StartSpellCheck
CtrlTabPressed
ShiftTabPressed

TWPAltStatusBar

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TWPAltStatusBar = class(TWPStatusBarBasic)

Anders als WPToolBar und WPRuler ist WPAltStatusBar für jedes Ihrer Projekte geeignet!
WPAltStatusBar unterscheidet sich nur im Design von WPStatusBar
Sie können einen Satz von Strings definieren auf die vom Programm aus einfach zugegriffen werden kann. Sie können auf eine Gauge zurückgreifen, die auch mit verschachtelten Funktionen bestens zurecht kommt. Sie können weiterhin aus jedem String eine Art Button machen, ohne jedoch dafür einen eigenen Handle zu verbrauchen.Und, last but not least: WPStatusBar übernimmt die Anordnung der Strings beim Verändern der Fenstergröße.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg
Gauge Control

Strings
RtfEdit
ShowSizer
ShowGauge
GaugeValue
GaugeWidth
UseGaugeForWP

GaugeRestrictSub
GaugeIn
GaugeOut
GaugeInc
GaugeReset
SetStringStyle
SetStringStyleIndex
SetString
SetStringIndex
SetStringWidth
SetStringWidthIndex

OnSelection
OnUpdate

TWPMemoryFormat

Siehe auch

Deklaration : TWPMemoryFormat = (fmRichText, fmPlainText);

TWPMemoryFormat repräsentiert den Modus in dem WPTools die Daten im Speicher hält.

fmRichText speichert für jedes Zeichen zusätzlich etwa 10 Bytes Attribute.

WPRichText
DataStructures
TAttr

TWPPagePropDlg

Siehe auch Eigenschaften Beispiel

Deklaration : TWPPagePropDlg = class(TComponent)

WPPagePropDlg ist eine VCL die es erleichtert die Dimensionen der Seite einzustellen. Sie müssen nur die Eigenschaft EditBox festlegen und können dann mit Execute den modalen Dialog starten.
Die VCL bemüht übrigens die Funktion TWPRichText.Set_PageSize um die Dimensionen einzustellen.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg

EditBox

```
procedure TRtfTextEditForm.PageSize1Click(Sender: TObject);
begin
   WPPagePropDlg1.Execute;
end;
```

TWPParagraphBorderDlg

Siehe auch Eigenschaften Beispiel

Deklaration : TWPParagraphBorderDlg = class(TComponent

Diese Komonente erleichtert die Definition von den jeweiligen Absatzrahmen. Die Absatzrahmen werden für den aktuellen Absatz bzw. die ausgewählten Absätze definiert.

Die Komponente ruft intern die Prozedur TWPRichText.Set_ParBorder(typ, width, color, space) auf.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg
Set_ParBorderFinish
Set_ParBorder

EditBox

```
procedure TRtfTextEditForm.ParagraphBorders1Click(Sender: TObject);
begin
  WPParagraphBorderDlg1.Execute;
end;
```

TWPParagraphPropDlg

Siehe auch Eigenschaften Beispiel

Deklaration : TWPParagraphPropDlg = class(TComponent)

Diese Komponente dient dazu die Absatzränder und -Abstände festzulegen. Beinflußt wird der aktuelle Absatz bzw die gerade ausgewählten Absätze.

Intern werden die procedurenTWPRichText.Set_ParSpacing(before, between, after) und TWPRichText.Set_ParMargin(first, left, right) aufgerufen.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg

EditBox

```
procedure TRtfTextEditForm.Indentation1Click(Sender: TObject);
begin
  WPParagraphPropDlg1.Execute;
end;
```

TWPRichText

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TWPRichText = class(TWPCustomRtfEdit)

WPRichText erlaubt es, formatierten Text - beinahe beliebiger Länge - zu editieren. Wahlweise kann das Speicherformat auch um die Textattribute reduziert werden und somit unformatierte Texte größer als zwei Megabytes zu bearbeiten.

Sie können Text zur Design Zeit eingeben und so Formatierungen vorgeben. WPRichText unterstützt Zeilenausrichtung, Absatzabstände, Absatzrahmen, Fontauswahl, Farben ....

Um durch Ihr Programm Text eingeben zu lassen können Sie die Funktionen PutParText oder InsertParText oder InputText verwenden. Sie können der Eigenschaft Lines die Strings eines Memos oder einer Liste zuweisen, oder Sie können den Text der in einem TWPRtfStorage gespeichert ist, der property RtfText zuweisen.

ToolBar
DBWPRichText
TWPRichTextLabel

HyperLinkCursor
NoBlockMarking
OneClickHyperlink
MemoryFormat
CaretDisabled
FitToWindowHorz
SaveFormat
LoadFormat
Zooming
Resizing
BackgroundColor
RtfText
Runtime only:
SelText,SelLength,SelStart
LastError
Lines
Modified
CPLineNr
CPLine
CPPosition
IsLastLine
SetModeControl

FontSelect
SaveAs Save
Load Insert
PrinterSetup Print!mPrint Print_xywh
ReplaceDialog FindDialog
DeleteSelection
PasteFromClipboard + Copy/CutToClipboard
LoadFromStream LoadFromStreamWithReader
SaveToStream SaveSelectionToStream
SaveToStreamWithWriter SaveSelectionToStreamWithWriter
SaveToFile LoadFromFile
Clear
LoadTemplate
WriteStatus
Find
FindNext
Spell_...
GetFontNr
GetFontName
Set_PageSize
Set_ParSpacing
Set_ParMargin
Change_ParSpacing
Set_ParBorderFinish
Set_ParBorder
PutParText,InsertParText,GetParText
InputText
GetTextLen GetTextBuf
SetSelTextBuf SetTextBuf
SetPosition

HyperLinkEvent
PrintFooter
PrintHeader
EditStateEvent
StartSpellCheck
CtrlTabPressed
ShiftTabPressed

TWPRichTextLabel

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TWPRichTextLabel = class(TWPCustomRtfLabel)

WPRichTextLabel ist eine Komponente um formatierten Text anzuzeigen, der weder gescrollt werden kann noch editierbar ist. Dafür kann der Text Transparent ausgegeben werden. WPRichTextLabel ist damit ein leistungsfähiges Tool um eindrucksvolle Präsentationen zu erstellen.

WPRichTextLabel stammt von dem Object TGraphicControl ab, wogegen TWPRichText von einem TCustomControl abstammt. Dies bedeutet, daß das RichText Label die Canvas und das Fenster Handle seines Parent-Object benutzt.

WPRichText

HyperLinkCursor
NoBlockMarking
OneClickHyperlink
FitToWindowHorz
RtfText
Runtime only:
CPLineNr
CPLine
CPPosition
IsLastLine
SaveFormat
LoadFormat
Zooming
Resizing
BackgroundColor

GetFontNr
GetFontName
Set_PageSize
Set_ParSpacing
Set_ParMargin
Change_ParSpacing
Set_ParBorderFinish
Set_ParBorder
GetTextLen mGetTextBuf
SetSelTextBuf SetTextBuf
SetPosition

HyperLinkEvent

TWPRtfStorage

Eigenschaften Methoden Beispiel

Deklaration : TWPRtfStorage = class(TComponent)

unit WPRtfIO;

TWPRtfStorage is eine Komponente um RTF Text zu speichern (normalerweise zur Designzeit) um ihn auf Wunsch in eine visuelle Komponente zu übertragen.

TWPRtfStorage besitzt nur eine Eigenschaft:
property RtfText;

sowie die Methoden:

```
procedure    Clear;
function     LoadFromStream(s : TStream): TWPLoadFormat; virtual;
procedure    LoadFromStreamWithReader(s : TStream;Reader:TWPTextReader);
```

RtfText

LoadFromStream
LoadFromStreamWithReader

```
WPRichTextLabel1.ButtonClick(Sender:TObject);
begin
   WPRichTextLabel1.Visible := FALSE;
   WPRichTextLabel1.RtfText.Assign(WPRtfStorage1.RtfText);
   WPRichTextLabel1.Visible := TRUE;
end;
```

TWPRuler

Eigenschaften Ereignisse

Deklaration : TWPRuler = class(TCustomControl)

WPRuler stellt das Lineal von WPRichText dar und wird, sobald die entsprechende Eigenschaft in WPRichText gesetzt wurde, von diesem komplett verwaltet.

Die einzige Eigenschaft, die für Sie wichtig ist, ist Units. Sie können für das Lineal entweder die Einheit `Centimeter' oder `Inch' wählen, auch zur Laufzeit.

Ein Lineal kann mehrere WPRichText bedienen, sogar wenn diese nicht direkt untereinander stehen. Es wird dann einfach entsprechend verschoben.

Units
XOffset
ShowAllTabPos

OnSelection
WhileSelection

TWPSpellCheckDlg

Siehe auch Eigenschaften Beispiel

Deklaration : TWPSpellCheckDlg = class(TComponent)

Dies ist eine Beispielkomponente, die zeigt wie der Text in einer WPRichText oder DBWPRichText Komponente nach Tipfehlern überprüft werden kann. Den Quellcode finden Sie in der unit Wpspdlg1.PAS. Da TWPSpellCheckDlg keinen Zugriff auf ein Lexicon besitzt, ist es nur Anschaungsmaterial.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg

EditBox

This is a part of the sourcecode:

```
procedure TWPSpDialog.IgnoreClick(Sender: TObject);
var
 s : String;
begin
 while TRUE do
 begin
  EditBox.Spell_FromCursorPos;
  s := EditBox.Spell_GetNextWord;
  if s='' then break;
  { test Dictionary .... ##################################
    if not InDictionary(s) then
    begin
       EditBox.Spell_SelectWord;
       WordList.Strings.Assign( GuessList );
       break;
    end
    else continue;

##########################################################
######## }
   { without a dictonary any word is unknown: }
      EditBox.Spell_SelectWord;
      break;
 end;
 Word.Text      := s;
 ReplaceAs.Text := s;
 if (s='') and (fsFirstCall=FALSE) then close;
end;

{ Replace will replace the selected Text.
  If you use the procedure
  EditBox.Spell_ReplaceWord(s : string) the replacment will be done
  invisibly.
  If you have a non modal spell dialog, the use of Spell_ReplaceWord might
  cause errors if the user has changed the text meanwhile.
}

procedure TWPSpDialog.ReplaceClick(Sender: TObject);
begin
  if CompareStr(EditBox.SelText,Word.Text)=0 then  { Replace selceted Text }
     EditBox.SelText := ReplaceAs.Text;
  IgnoreClick(Sender);
end;
```

```
procedure TWPSpDialog.CancelClick(Sender: TObject);
begin
  EditBox.Spell_FromCursorPos;
  Close;
end;
```

TWPStatusBar

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TWPStatusBar = class(TWPStatusBarBasic)

Anders als WPToolBar und WPRuler ist WPAltStatusBar für jedes Ihrer Projekte geeignet!
WPAltStatusBar unterscheidet sich nur im Design von WPStatusBar
Sie können einen Satz von Strings definieren auf die vom Programm aus einfach zugegriffen werden kann. Sie können auf eine Gauge zurückgreifen, die auch mit verschachtelten Funktionen bestens zurecht kommt. Sie können weiterhin aus jedem String eine Art Button machen, ohne jedoch dafür einen eigenen Handle zu verbrauchen.Und, last but not least: WPStatusBar übernimmt die Anordnung der Strings beim Verändern der Fenstergröße.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg
Gauge Control

Strings
RtfEdit
ShowSizer
ShowGauge
GaugeValue
GaugeWidth
UseGaugeForWP

GaugeRestrictSub
GaugeIn
GaugeOut
GaugeInc
GaugeReset
SetStringStyle
SetStringStyleIndex
SetString
SetStringIndex
SetStringWidth
SetStringWidthIndex

OnSelection
OnUpdate

TWPToolBar

Siehe auch Eigenschaften Methoden Ereignisse

Deklaration : TWPToolBar = class(TCustomPanel)

Die Toolbar dient dazu ein oder mehrere TWPRichText oder DBWPRichText Komponenten zu kontrollieren. Zu diesem Zweck stellt die Toolbar etliche vordefinierte SpeedButtons und Listboxen bereit. So wird eine Auswahl der Schriftart oder Größe sowie der Textfarbe stark erleichert.
An Speedbuttons stehen Schalter für Textstyles wie 'fett' oder Ausrichtung 'Blocksatz' zur Verfügung. Wenn Sie weitere Objekte einbinden wollen können Sie dies leicht mit AddControl tun. Die Toolbar wird die eingefügten Objecte zusammen mit den vordefinierten Objecten anordnen.
Die meisten Ereignisse die beim Anklicken eines Elements in der Toolbar auftreten wird von WPRichText/DBWPRichText behandelt. An das zuletzt aktivierte Edit Feld wird das Kommando gesendet.
Möglicherweise möchten Sie jedoch anders reagieren. Dann müssen Sie einen Handler für das OnSelection und OnIconSelection Ereignis bereitstellen. Dies ist insbesondere nötig wenn Sie den New und Close Button 'bedienen' wollen.
Wenn Sie kein WPRichText oder DBWPRichText Object auf Ihrem Formular haben, kann die Toolbar auch die Schriftart eines TMemo, TEdit oder TPanel Objects ändern. Geben Sie den Namen und/oder das Object in der Eigenschaft UpdateObjectName/ UpdateObject an.

Wenn die Toolbar aligntop oder alignbottom ausgerichtet ist, wird sich Ihre Höhe entsprechend dem Platzbedarf der enthaltenen Komponenten ändern.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg

UpdateObject
UpdateObjectName
ShowFont
sel_ListBoxes
sel_StatusIcons
sel_ActionIcons
sel_DatabaseIcons
sel_EditIcons
FontSizeFrom
FontSizeTo
Hints

GetIcon
GetStyleBox
GetStyleItems
GetFontBox
GetFontItems
GetSizeBox
GetSizeItems
AddControl
RemoveControl
SelectIcon
DeselectIcon
EnableIcon

OnSelection
OnIconSelection

CtrlTabPressed

Deklaration : property CtrlTabPressed : TNotifyEvent;

Wenn Sie WantTab auf TRUE setzen, ist es nicht mehr möglich das Eingabeformular mit einem Tastendruck zu verlassen. Um dem Anwender trotzdem zu ermöglichen, schnell zu dem nächsten Eingabefeld zu wechseln tragen Sie einen Handler bei CtrlTabPressed ein. Sie können in dem Handler den Focus auf ein beliebiges anderes Objekt setzen, indem Sie einfach ObjectName.SetFocus aufrufen.

EditStateEvent

Beispiel

Deklaration : property EditStateEvent;

Ein EditStateEvent tritt auf, wenn sich der Inhalt der Zwischenablage ändert. Sie können darauf reagieren und Buttons oder Menüeinträge zu disablen bzw. enablen.

```
procedure TForm1.EditStateEvent(Sender: TObject;
  selection_marked, clipboard_not_empty: Boolean);
begin
  { Cut/copy/Paste-Item are of type TMenuItem }
  CutItem.Enabled := selection_marked;
  CopyItem.Enabled := selection_marked;
  PasteItem.Enabled := clipboard_not_empty;
end;
```

HyperLinkEvent

Siehe auch

Deklaration : property HyperLinkEvent : THyperLinkEvent;

Ein Hyperlink Ereignis tritt auf, wenn der Anwender auf den Text klickt, der asl HyperLink markiert wurde.

Ein Handler könnte z.B. so aussehen:

```
procedure TForm1.WPRichText1HyperLinkEvent(Sender: TObject; text,
   stamp: String; LineNumber: Longint);
var
   c : array[0..255] of Char;
begin
   StrPcopy(c,text);
   MessageBox(Handle,c,'Hyperlink clicked',0);
end;
```

Der markierte Text wird in 'text' übergeben, die Zeilennummer in LineNumber. 'Stamp' ist reserviert.

TWPRichText
TWPRichTextLabel

PrintFooter

Siehe auch Beispiel

Deklaration : property PrintFooter : TWPPrintEvent

Dieser Event wird ausgelöst um den Fuß Bereich einer Seite beim Ausdruck mittels der Funktion PRINT zu ermöglichen.

TWPPrintEvents werden von der Funktion Print ausgelöst, immer dann wenn der Kopf- bzw. der Fußbereich einer Seite ausgegeben werden muß. Da dem Eventhandler die Canvas übergeben wird und auch die Position wo darauf zu schreiben ist, ist es sehr einfach Seitennumerierung oder die Ausgabe von Logos zu programmieren.

TWPRichText

```
{ Printing of the header }
procedure TForm1.WPRichText1PrintHeader(Sender: TObject;
  prCanvas: TCanvas; x, y, w, h, PageNumber: Integer; LineNumber: Longint);
begin
  prCanvas.Font.Name := 'Arial';
  prCanvas.Font.Size := 14;
  prCanvas.Font.Color:= clBlue;
  prCanvas.MoveTo(x+(wprCanvas.TextWidth('Report'))div 2,
                  y+(hprCanvas.TextHeight('A'))div 2);
  prCanvas.TextOut(x+(wprCanvas.TextWidth('Report'))div 2,
                  y+(hprCanvas.TextHeight('A'))div 2,
                  'Report');
end;

{Printing of the pagenumber  }
procedure TForm1.WPRichText1PrintFooter(Sender: TObject;
  prCanvas: TCanvas; x, y, w, h, PageNumber: Integer; LineNumber: Longint);
begin
  prCanvas.Font.Name := 'Arial';
  prCanvas.Font.Size := 12;
  prCanvas.Pen.Color := clBlack;
  prCanvas.MoveTo(x,y);
  prCanvas.LineTo(x+w,y);
  prCanvas.MoveTo(x+(wprCanvas.TextWidth(''))div 2,
                  y+(hprCanvas.TextHeight('A'))div 2);
  prCanvas.TextOut(x+(wprCanvas.TextWidth(''))div 2,
                  y+(hprCanvas.TextHeight('A'))div 2,
                  ' '+ IntToStr(PageNumber)+' ');
end;
```

TWPRuler.OnSelection

Beispiel

Deklaration : property OnSelection : TWPRulerSelectEvent;

Dieses Ereignis tritt auf wenn die Marker für den rechten und linken Absatzrand verschoben werden.

Die neuen Werte können Sie in
WPRuler.Header.Layout.indentFirst
WPRuler.Header.Layout.indentLeft
WPRuler.Header.Layout.indentRight
abfragen.

Beachte Sie: Das Lineal ist nur dann in Funktion, wenn es von einer WPRichText Komponente kontrolliert wird.

```
procedure TWPRichText.OnRulerSelection(Sender: TObject; var Typ:
TWpSelNr;
  var group, num: Integer; changing: Boolean);
begin
 if Assigned(FOldOnRulerSelection) then
        FOldOnRulerSelection(Sender,Typ,group,num,changing);
 if FHasFocus then
 begin
   if Typ<>wptNone then
   begin
      if Typ=wptParagraph then
      begin
        if Changing then
        begin
             if Assigned(FOnChange) then FOnChange(Self);
        end;
      end else
      if Typ=wptPage then
      begin

      end;
      Typ:=wptNone;
   end;
 end;
end;
```

OnSelection

Siehe auch Beispiel

Deklaration : OnSelection : TWPStatusBarSelectEvent

Diese Funktion wird aufgerufen, wenn auf einen String in der StatusBar geklickt wird. Übergeben wird die ID des Strings oder sein Index, falls eine ID nicht feststeht.

TWPStatusBar

```
procedure TForm1.WPStatusBar1Selection(Sender: TObject; index: Integer;
  typ: TWPStatusItemCont; name: String; x, y, w: Integer;
  var Button: TMouseButton; var Shift: TShiftState);
begin
  if typ=stUnit then
  begin
   if WPRuler1.Units = Centimeter then
      WPRuler1.Units := Inch else WPRuler1.Units:=Centimeter;
   if WPRuler1.Units = Centimeter then
     WPStatusBar1.SetString(stUnit,'Cent.')
   else WPStatusBar1.SetString(stUnit,'Inch');
  end;
end;
```

OnUpdate

Siehe auch Beispiel

Deklaration : OnUpdate : TWPStatusBarUpdateEvent

Ereignis zur Initialisierung der Statusbar.

TWPStatusBar

```
procedure TForm1.WPStatusBar1Update(Sender: TObject);
begin
   if WPRuler1.Units = Centimeter then
     WPStatusBar1.SetString(stUnit,'Cent.',TRUE)
   else WPStatusBar1.SetString(stUnit,'Inch',TRUE);
end;
```

ShiftTabPressed

Siehe auch

Deklaration : property ShiftTabPressed : TNotifyEvent;

Wenn Sie WantTab auf TRUE setzen, ist es nicht mehr möglich das Eingabeformular mit einem Tastendruck zu verlassen. Um dem Anwender trotzdem zu ermöglichen, schnell zu dem nächsten Eingabefeld zu wechseln tragen Sie einen Handler bei ShiftTabPressed ein. Sie können in dem Handler den Focus auf ein beliebiges anderes Objekt setzen, indem Sie einfach ObjectName.SetFocus aufrufen.

TWPRichText

StartSpellCheck

Siehe auch

Deklaration : property StartSpellCheck : TNotifyEvent;

StartSpellCheck wird aufgerufen, wenn in der Toolbar der SpellCheck Button angewählt wird. Sie können (Sender as TWPRichText) dazu verwenden um herauszufinden welcher Text überprüft werden soll.

SpellCheckInterface

ToolBar.OnIconSelection

Siehe auch Beispiel

Deklaration : property OnIconSelection : TWPIconSelectEvent;

Der Handler für OnIconSelection wird aufgerufen, wenn der Anwender einen der Speedbuttons anwählt.

Dem Handler werden die folgenden Parameter übergeben:

```
var Typ   : TWpSelNr;
var str   : String;
var group : Integer;
var num   : Integer;
var index : Integer
```

typ kann sein wptNone oder wptIconSel bzw. wptIconDeSel, je nachdem ob ein button angewählt oder abgewählt wurde.

str und index is reserviert.

in group wird angegeben, as welcher Gruppe der Speed Button stammt wogegen num die Nummer innerhalb der Gruppe angibt. Folgende Constanten sind definiert in WPDefs:

Gruppen:

```
WPI_GR_STYLE = 1;
WPI_GR_ALIGN = 2;
WPI_GR_EDIT  = 3;
WPI_GR_DISK  = 4;
WPI_GR_PRINT = 5;
WPI_GR_DATA  = 6;
```

Nummern:

```
WPI_CO_Normal=1;    { Group: WPI_GR_STYLE }
WPI_CO_Bold  =2;
WPI_CO_Italic=3;
WPI_CO_Under =4;
WPI_CO_HyperLink = 5;
WPI_CO_StrikeOut = 6;

WPI_CO_Left  =1;  { Group: WPI_GR_ALIGN }
WPI_CO_Right =2;
WPI_CO_Justified =3;
WPI_CO_Center=4;

WPI_CO_Copy  =1; { Group WPI_GR_EDIT }
WPI_CO_Cut   =2;
WPI_CO_Paste =3;
WPI_CO_SelAll =4;
WPI_CO_HideSel=5;
WPI_CO_Find   =6;
WPI_CO_Replace=7;
WPI_CO_SpellCheck=8;

WPI_CO_Exit  =1; { Group: WPI_GR_DISK }
WPI_CO_New   =2;
```

```
WPI_CO_Open   =3;
WPI_CO_Save   =4;
WPI_CO_Close =5;

WPI_CO_Print =1; { Group: WPI_PRINT }
WPI_CO_PrintSetup=2;

WPI_CO_Next    =1; { Group: WPI_DATA }
WPI_CO_Prev    =2;
WPI_CO_Add     =3;
WPI_CO_Del     =4;
WPI_CO_Edit    =5;
WPI_CO_Cancel =6;
WPI_CO_ToStart=7;
WPI_CO_ToEnd   =8;
WPI_CO_Post    =9;
```

Wenn Sie das Ereignis behandelt haben müssen Sie typ := wptNone setzen um eine weitere Bearbeitung zu unterbinden. Um den jeweiligen Button nach der Aktion abzuwählen verwenden können Sie ToolBar.DeselectIcon(0, group, num); aufrufen.

TWPToolBar

The OnToolBarIconSelection wich is defined in TDBWPRichText.

```
procedure TDBWPRichText.OnToolBarIconSelection(Sender: TObject;
  var Typ: TWpSelNr; var str: String; var group, num, index: Integer);
begin
 if (typ=wptIconSel) then
 begin
   if (FDataLink.DataSource<>nil) and  (group=WPI_GR_DATA) then
   begin
     if (FDataLink.DataSource.DataSet<>nil)
        and (FDataLink.DataSource.State<>dsInactive)
     then
     begin
      case num of
       WPI_CO_Next   : FDataLink.DataSource.DataSet.Next; { Group: WPI_DATA
}
       WPI_CO_Prev   : FDataLink.DataSource.DataSet.Prior;
       WPI_CO_Add    : FDataLink.DataSource.DataSet.Insert;
       WPI_CO_Del    : FDataLink.DataSource.DataSet.Delete;
       WPI_CO_Edit   : if FDataLink.editing=FALSE then
                         FDataLink.DataSource.DataSet.Edit;
       WPI_CO_Cancel : FDataLink.DataSource.DataSet.Cancel;
       WPI_CO_Post   : FDataLink.DataSource.DataSet.Post;
       WPI_CO_ToStart: FDataLink.DataSource.DataSet.First;
       WPI_CO_ToEnd  : FDataLink.DataSource.DataSet.Last;
      end;
      WPToolBar.DeselectIcon(0,WPI_GR_DATA,num);
      EnableDataControlIcons;
      typ:=wptNone;
     end;
   end { Group }
   else if Focused then
   begin { handle some of the other Buttons }
       if group=WPI_GR_DISK then
         case num of
           WPI_CO_ClOSE : begin
WPToolBar.DeselectIcon(0,WPI_GR_DISK,num);
                            Typ:=wptNone; { You got the File from Database, you
can't close it ! }
                          end;
         end;
    end;
  end;
 end;
```

ToolBar.OnSelection

Siehe auch Beispiel

Deklaration : property OnSelection : TWPSelectEvent;

Der Handler für OnSelection wird aufgerufen, wenn der Anwender einen der Einträge in den Listboxen der ToolBar ausgewählt hat.

Der Kopf des Handler kann wie folgt aussehen:

```
procedure ToolBar1OnSelection(Sender: TObject;
                  var Typ: TWpSelNr;
                  str: String;
                  num: Integer);
```

Typ gibt an welche Listbox ausgewählt wurde, und kann einen der folgenden Werte annehmen:

- wptNone : Aufruf soll ignoriert werden
- wptName : Neuer Font Name steht in String
- wptSize : Neue Font Größe steht in num
- wptColor : Neue Textfarbe steht in num
- wptBkColor : Neue Texthintergrundfarbe steht in num
- wptTyp : Ein Eintrag aus der Stylebox wurde gewählt

Um auf die Auswahl reagieren, programmieren Sie am besten eine case Anweisung:

```
case Typ of
            wptName :
            wptSize:
            wptTyp:
            wptColor:
            wptBKColor:
end;
```

Wenn Sie das Ereignis abschließend behandelt haben, vergessen Sie bitte nicht wptNone der variable Typ zuzuweisen.

TWPToolBar

This handler is defined for the TWPRichText object:

```
procedure TWPRichText.OnToolBarSelection(Sender: TObject; var Typ:
TWpSelNr;
  var str: String; var num: Integer);
var
  atr    : Tattr;
  what   : TAttrWhat;
begin
{ if Focused then  }
 begin
   if Typ<>wptNone then
   begin
      if (Changing=FALSE) or ReadOnly then begin
LastError:=ecNoChangeAllowed; exit; end;
      atr:=Attr;
      what:=[];
      case Typ of
         wptName : begin
                      atr.Font:=Memo.header.AddFontName(str);
                      what := [awFontNr];
                   end;
         wptSize: begin
                      atr.Size := num;
                      what := [awFontSize];
                  end;
         wptTyp: begin  { StyleBox }
                      { ????     }
                  end;
         wptColor: begin
                     atr.Color := (atr.Color and 240) + (num and 15);
                     what:= [awFontColor];
                   end;
         wptBKColor: begin
                       atr.Color := (atr.Color and 15) + (num * 16);
                       what:= [awFontBKColor];
                     end;
      end; { case }
      Memo.SetActAttr(atr,what);
      Typ:=wptNone;
      Changed;
      invalidate;
   end;
 end;
end;
```

Change_ParSpacing

Siehe auch

Deklaration : procedure Change_ParSpacing(before,between,after:Longint);

Change_ParSpacing "ändert" die Absatz Abstände. Der Wert den Sie jeweils angeben wird zum aktuellen Wert addiert. Wenn Sie einen Wert nicht ändern wollen müßen Sie 0 übergeben, wenn Sie ihn erhöhen wollen einen Wert >0. Wenn Sie einen Abstand verringern wollen muß der Wert < 0 sein.
Change_ParSpacing ist dafür gedacht dem Anwender es zu erlauben die Abstände visuell abzustimmen.

Wenn Sie einen Wert exact setzen wollen ist die function Set_ParSpacing besser geeignet.

WPRichText

Clipboard Support

Siehe auch

Clipboard funktionen:

procedure SelectAll;
Selektiert den gesamten Text.

procedure CopyToClipboard;

Diese Funktion kopiert den ausgewählten Text in den Formaten ANSI und RTF zur Zwischenablage.

procedure CutToClipboard;

Wie CopyToClipboard. Anschliessend wird der Text geloescht.

procedure PasteFromClipboard;

Fügt die Daten die sich in der Zwischenablage befinden an der Cursor Position ein. Wenn ein Text selektiert ist, wird dieser dabei überschrieben.

procedure DeleteSelection;

Löscht den ausgewählten Text.

WPRichText

Find

Siehe auch

Deklaration : procedure Find(x : string;Backwards:Boolean)

procedure Find(x : string;Backwards:Boolean)

Diese Methode sucht nach einem String. Der String darf den Joker `*' enthalten und es wird rückwärts gesucht, wenn backwards=TRUE ist.Wenn Sie den gefunden Text verändern wollen, können Sie auf die Finder Klasse zugreifen. Es ist damit möglich, den gefunden Text zu ersetzen oder nur neue Attribute zu wählen. (Sie können dies natürlich auch tun indem Sie den Text selektieren und dann die Schrift ändern. Vom Programm gesteuert arbeitet Finder jedoch viel effizienter.)

procedure FindNext

Sucht erneut nach dem mit Find angegeben String.

WPRichText

Search + Replace Dialog

Siehe auch

procedure ReplaceDialog;

Diese Funktion zeigt den - nicht modalen - Suchen/Ersetzen Dialog an. Beim Suchen ist der Joker `*' erlaubt, sowie vorwärts und rückwärts Suche. Dauert das ersetzen zu lange, kann der Anwender mit Escape abbrechen.

procedure FindDialog;

Diese Funktion zeigt den - nicht modalen - Suchen Dialog an. Beim Suchen ist der Joker `*' erlaubt, sowie vorwärts und rückwärts Suche.

WPRichText

GetFontNr / GetFontName

Siehe auch Beispiel

function GetFontNr(name : TFontName):Integer;

Die aktuelle Schrift ist in der Struktur Tattr als Nummer gespeichert. GetFontNr gibt für einen Fontnamen die zugehörige Nummer zurück.

function GetFontName(nr : Integer) :TFontName;

Diese Funktion gibt für eine Zeichnesatz nummer den dazugehörigen Namen zurück.

Attributes
WPRichText

```
var
  a      : Tattr;

{ Sets a fontname }

a.font := GetFontNr(`Arial');
WPRichText1.Attr := a;

{ Gets a Fontname }
a      := WPRichText1.Attr;
name   := GetFontName( a.Font );
```

GetTextBuf / GetSelTextBuf

Siehe auch

Deklaration : function GetTextBuf(buffer : Pchar;BufSize: Longint):Longint;

Diese Funktionen kopieren Text in einen Speicherbereich und geben die Länge des kopierten Textes zurück. Der Speicherbereich kann grösser sein als 64KB!

GetTextBuf kopiert den gesamten Text (sofern Platz da ist), GetSelText kopiert nur den markierten Text, sofern überhaupt markiert wurde. Wird als buffer nil übergeben, wird nur die Länge berechnet.

Beachten Sie:
GetSelText fügt am Ende eines jeden Absatzes ein Carriage Return ein. Dadurch wird mehr paltz benötigt, als SelLenght angibt. Um die nötige Größe des Puffers zu ermitteln, sollten Sie daher GetSelTextBuf(nil,0) verwenden;

WPRichText

GetTextLen

Siehe auch

Deklaration : function GetTextLen: Longint;

Diese Funktion gibt die Länge des Textes zurück.

WPRichText

InputText

Siehe auch

Deklaration : procedure InputText(s:Pchar);

Mit InputText können Sie eine Text ab der Cursorposition im aktuellen Schreibstil einfügen. Der übergebene Text wird so behandelt, als ob er über die Tastatur eingegeben wurde.

WPRichText
CPPosition

Dialog Load / Save procedures

Siehe auch

procedure SaveAs;

Diese procedur fragt nach einem Dateinamen und speichert dann. Ist die Endung des Dateinamen `RTF' wird im RTF Format gespeichert. Sonst wird im ANSI Format gespeichert.

procedure Save;

Diese procedur fragt falls nicht vorher durch Load ein Dateinamen angegeben wurde, nach einem Dateinamen und speichert dann.

procedure Load;
procedure Insert;

Beide Funktionen fragen nach Dateinamen und laden dann. Insert fügt den Text ab Cursorposition ein, Load löscht vorher den gesamten Text. Der Dialog wird nicht angezeigt, wenn ReadOnly=TRUE ist.

Achtung! Keine Function fragt vor dem Überschreiben einer Datei nach! Um zu prüfen ob das Sicher / Laden erfolgreich war, können Sie LastError auf <> ecOk testen.

WPRichText

Loading and Saving

Siehe auch

function LoadFromStream(s : TStream): TWPLoadFormat; virtual;
procedure SaveToStream(s : TStream); virtual;
procedure SaveSelectionToStream(s : TStream); virtual;

Diese Methoden schreiben auf Streams bzw. lesen davon. LoadFromStream gibt das Format zurück, in dem gelesen wurde. Das Schreib / Leseformat wird entsprechend der Eigenschaft LoadFormat und SaveFormat gewählt. Zur Auswahl stehen RTF oder ANSI Format.
Um zuprüfen ob der Save Vorgang erfolgreich war, können Sie unmittelbar nachher WPRichText.LastError auf <>ecOk testen.

procedure LoadFromStreamWithReader(s : TStream;Reader:TWPTextReader);

Wenn Sie ein anderes Format lesen wollen, als ANSI oder RTF dann können Sie eine neue Reader-Klasse von TWPTextReader ableiten. Übergeben Sie LoadFromStreamWithReader eine Instanz der neuen Klasse.

procedure SaveToStreamWithWriter(s : TStream; Writer:TWPTextWriter);

procedure SaveSelectionToStreamWithWriter(s : TStream; Writer:TWPTextWriter);

Wenn Sie ein anderes Format schreiben wollen, als ANSI oder RTF dann können Sie eine neue Writer-Klasse von TWPTextWriter ableiten. Übergeben Sie SaveToStreamWithReader eine Instanz der neuen Klasse.

procedure SaveToFile(const name : string);
procedure LoadFromFile(const name : string);

Diese Funktionen schreiben in eine Datei bzw lesen von Ihr. SaveToFile stellt fest ob der Datename mit `RTF' endet und speichert dann im RTF Format.

WPRichText

Print

Siehe auch

Deklaration : procedure Print;

Diese Funktion druckt den Text aus. Falls gewünscht können die Kopf und Fuß Bereiche bedruckt werden, indem Handler für den TWPPrintEvent geschrieben werden.
Während des Drucken ist eine Bildschirmausgabe nicht mehr möglich, da interen der Text für den Drucker umformatiert wurde.

WPRichText

Print_xywh

Siehe auch

unit WPWinCtr;

function TWPCustomRtfEdit.Print_xywh(outCanvas:TCanvas;x,y,w,h:Integer;
line_nr:Longint;mode : TWPPrintMode):Longint;

Diese Funktion druckt auf ein Canvas in dem Rechteck, das durch x,y,w,h angegeben wird. line_nr ist die erste Zeile die gedruckt werden soll. Die Funktion gibt 0 oder die Nummer der ersten Zeile die nicht mehr gedruckt werden konnte zurueck.
Wenn die Option pmUsePageBreakes in der Eigenschaft Print_xywh_Mode gesetzt ist, kehrt die function beim nächsten Seitenumbruch zurück.

Mode kann sein:
wpPrintCalc: es wird nicht gedruckt, nur die Zeilen gezählt
wpNormalPrint: es wird normal gedruckt
wpFastPrintInit: die Funktion wird wird initialisiert
wpFastPrint: es wird gedruckt
wpFastPrintExit: die Initailisierung wird rückgängig gemacht.

Wenn Sie kurze Texte drucken können Sie bedenkenlos wpNormalPrint verwenden. Für längere Texte sollten Sie FastPrint benutzen:

1. Initialisieren mit FastPrintInit
2. Solange wie nötig mit FastPrint drucken,
3. FastPrintExit aufrufen.

Example

PutParText / InsertParText / GetParText

Siehe auch

Mit diesen Funktionen können Sie einzelne Absätze in den Text einfügen oder an diesen anhängen. Dem hinzugefügten Text werden die aktuellen Attribute zugewiesen.

procedure PutParText(Index:Longint;const Buffer:Pchar;Size:Integer);

Setzt den Absatz mit der Nummer Index gleich dem angebenem Speicherbereich.

procedure InsertParText(Index:Longint;const Buffer:Pchar;Size:Integer);

Fügt vor Absatz Nummer Index einen neuen Absatz ein.

Beachten Sie: Um den neuen Text auch anzuzeigen müssen Sie noch invalidate aufrufen!

Um einen Absatz einzulesen verwenden Sie:

function GetParText(Index:Longint;Buffer:Pchar;Size:Integer):Integer;

Liest den Absatz mit der Nummer Index in den angebenen Speicherbereich.

WPRichText
InputText
Attr

WPRtfStorage.LoadFromStream

Deklaration : LoadFromStream

function LoadFromStream(s : TStream): TWPLoadFormat;

Dies Methode liest aus einem Stream im ANSI oder RTF Format.

procedure LoadFromStreamWithReader(s : TStream;Reader:TWPTextReader);
Wenn Sie ein anderes Format lesen wollen, als ANSI oder RTF dann können Sie eine neue Reader-Klasse von TWPTextReader ableiten. Übergeben Sie LoadFromStreamWithReader eine Instanz der neuen Klasse.

StatusBar.SetString

Siehe auch

Deklaration : function SetString(typ : TWPStatusItemCont;name : string) : Boolean;

Um die angezeigtene Texte in der Statusbar zu ändern können Sie auf die einzelnen Felder mit der Funktion SetString zugreifen. Sie müssen der Funktion lediglich die ID des Strings übergeben sowie der neuen Wert und der String wird sofort neu gezeichnet.

Gültige IDs sind:
'stHelp','stIndex','stOutline','stStatus','stInsert','stCaps','stNum','stXPos','stYPos','stDate','stTime','stPage','stLine','stChanged','stFont','stStyle', 'stUnit', 'stTxtFmt', 'stName', 'stFileName','stPathName'

Beispiel:
StatusBar1.SetString(stStatus,'Text wurde geladen');

TWPStatusBar

StatusBar.SetStringIndex

Siehe auch

Deklaration : function SetStringIndex(index : Integer;name : string): Boolean;

Um die angezeigtene Texte in der Statusbar zu ändern können Sie auf die einzelnen Felder mit der Funktion SetString zugreifen. Sie müssen der Funktion lediglich den Index des jeweiligen Strings in der Eigenschaft Strings übergeben sowie den neue Wert und der String wird sofort neu gezeichnet.

Normalerweise werden Sie SetString(ID ...) verwenden, da Sie dadurch unabhängig von der Reihenfolge der Strings sind.

Beispiel:
StatusBar1.SetStringIndex(1,'Feld 1 wurde geändert');

TWPStatusBar

StatusBar.SetStringStyle

Siehe auch

Deklaration : SetStringStyle(typ : TWPStatusItemCont;style:TWPStatusItemStyle;color:TColor)

Diese Funktion setzen den Style und die Farbe des Strings. Style ist folgendermaßen definiert:

TWPStatusItemStyle = set of (sttButton, sttDown, sttDisabled, sttColor, sttBKColor);

wenn Sie sttColor angeben, wird der String in der übergebenen Farbe gezeichnet, wenn Sie alternativ sttBKColor angeben wird die Hintergrundfarbe geändert.

sttButton, sttDown und sttDisabled können Sie auch zur Designzeit auswählen, indem Sie den Strings @,$ oder % voranstellen. Dies erledigt der Property Editor aber auch für Sie.

Wenn Sie keinen vordefinierten String verwendet haben, können Sie auch die Funktion

function SetStringStyleIndex(index : Integer;style:TWPStatusItemStyle;color:TColor) : Boolean;
verwenden.

TWPStatusBar

SetStringWidth

Siehe auch

Deklaration : function SetStringWidth(typ : TWPStatusItemCont;width : Integer) : Boolean;

Mit dieser Funktion setzten Sie die Breite eines Strings in der Statusbar. Die Breite die Sie angeben wird relativ zu den anderen Breiten gewertet.

Gültige IDs sind:
'stHelp','stIndex','stOutline','stStatus','stInsert','stCaps','stNum','stXPos','stYPos','stDate','stTime','stPage','stLine','stChanged','stFont','stStyle', 'stUnit', 'stTxtFmt', 'stName', 'stFileName','stPathName'

TWPStatusBar

StatusBar.SetStringWidthIndex

Siehe auch

Deklaration : function SetStringWidthIndex(index : Integer ;width : Integer) : Boolean;

Mit dieser Funktion setzten Sie die Breite eines Strings in der Statusbar. Die Breite die Sie angeben wird relativ zu den anderen Breiten gewertet. Sie müssen der Funktion lediglich den Index des jewiligen Strings in der Eigenschaft Strings übergeben sowie die neue Breite und der String wird sofort neu gezeichnet.

TWPStatusBar

Set_PageSize

Siehe auch

Deklaration : procedure
Set_PageSize(nmargl,nmargr,nmargt,nmarg,npagew,npageh : Longint);

Mit Set_PageSize können Sie die Seitengrösse sowie die Ränder angeben. Alle Angaben sind in twips (1/1440 Zoll) anzugeben. Wenn Sie einen Wert nicht angeben wollen, setzen Sie als Wert `KeepOldValue'.

nmargl ist der linke Rand, nmargr der Rechte.
nmargt ist der obere Rand, nmargb der Untere.

Die Funktion print wird den oberen und unteren Rand nicht bedrucken. Sie können ihn für Kopf- oder Fußzeilen nutzen.
npagew gibt die Papierbreite an,
npageh die Papierlänge.

WPRichText

Set_ParBorder

Siehe auch Beispiel

Deklaration : Set_ParBorder(LineType : TBorderType; Thick : Integer; Color, Space : Integer);

Set_ParBorder definiert den Typ des Absatzrahmens um den aktuellen Absatz / ausgewählte Absätze.

LineType kann enthalten:
BlTop,
BLBottom,
BlLeft,
BlRight,
BLBox,
BLDouble,
BLDot,
BlTabLines

Thick gibt in twips an, wobei die kleinste Einheit 1pt ist (20 twips) wie dick die Linie sein soll.

Color gibt den Index der Farbe an (0..15)

Space gibt den Abstand zwischen Text und Rahmen in mm an.

WPRichText
Set_ParBorder

{ part of the source of TWPParagraphBorderDlg }

```
procedure TWPParagraphBord.BitBtn2Click(Sender: TObject);
var
  typ : TborderType;
  col,wid,spc : Integer;
begin
  if fsEditBox<>nil then
  begin
    typ := [];
    if box.Checked then include(typ,blBox);
    if topbox.Checked then include(typ,blTop);
    if leftbox.Checked then include(typ,blleft);
    if bottombox.Checked then include(typ,blbottom);
    if rightbox.Checked then include(typ,blright);
    if double.Checked then include(typ,blDouble);
    if dotted.Checked then include(typ,blDot);
    if typ<>[] then
    begin
      if ColorText.Text<>'' then
         col := StrToInt(ColorText.Text)
         else col:=0;
      if SpaceText.Text<>'' then
         spc := StrToInt(SpaceText.Text)
         else spc:=0;
      if WidthText.Text<>'' then
         wid := StrToInt(WidthText.Text)*20
         else wid :=1;
      fsEditbox.Set_ParBorder(typ, wid, col,spc);
    end
    else fsEditbox.Set_ParBorder([], 0, 0, 0);
  end;
  ModalResult := IDOK;
end;
```

Set_ParBorderFinish

Siehe auch

Deklaration : procedure TWPRichText.Set_ParBorderFinish;

Set_ParBorderFinish beendet die Rahmen Einstellung für den aktuellen Absatz. Der Absatzrahmen wird nicht für die folgenden Absätze weitergeführt.
Set_ParBorderFinish löscht keine Absatzrahmen, sondern unterteilt sie nur.

Absatzrahmen löschen können Sie mit Set_ParBorder([],0,0,0).

TWPRichText
TAttr
Set_ParBorder

Set_ParMargin

Siehe auch

Deklaration : procedure Set_ParMargin(first,left,right:Longint);

Mit dieser Funktion können Sie die Absatzabstände setzen. Die Angaben erfolgen in twips, erlaubt sind Werte von 0 bis 32000 wenn Sie Delphi 1 (16 Bit) verwenden. Mit Delphi 2 gibt es keine Beschränkung.

WPRichText

Set_ParSpacing

Siehe auch

Deklaration : procedure Set_ParSpacing(before,between,after:Longint);

Dieser Prozedur übergeben Sie die neuen Absatzabstände oder KeepOldValue. Die Funktion wird den aktuellen Absatz ändern oder die, die garde markiert sind.

before ist der Abstand vor einem Absatz, angegeben in twips.

between ist die Zeilenhöhe in twips. Wenn Sie 0 ist wird die Zeilenhöhe nach dem höchsten Buchstaben berechnet.

after ist der Abstand nach dem Absatz.

Um einen Wert nicht zu ändern, übergeben Sie KeepOldValue.

Die Angaben erfolgen in twips, erlaubt sind Werte von 0 bis 32000 wenn Sie Delphi 1 (16 Bit) verwenden. Mit Delphi 2 gibt es keine Beschränkung.

WPRichText

SetPosition / GetPosition

Siehe auch

procedure SetPosition(pos,lin,par,pos_in_par:Longint);

Diese Funktion setzt die Cursor Position neu. Sie können angeben:
pos, als absolute Position vom Textanfang. Wenn der Wert grösser ist als die Textlänge, wird zur letzten Position im Text gesprungen. - oder -
lin, als Zeilennummer -oder -
par als Absatznummer und pos_in_par als Position im Absatz

procedure GetPosition(var pos,lin,par,pos_in_par:Longint);

Diese Methode ermittelt die Cursorposition.

WPRichText

SetSelTextBuf

Siehe auch

Deklaration : procedure SetSelTextBuf(s : Pchar);

Diese Funktion ersetzt den ausgewählten Text durch den Text im Speicherbereich auf den s zeigt. Er kann grösser sein als 64KB und entweder Text im ANSI oder im RTF Format enthalten.
RTF Format wird erwartet wenn der Text mit "{\rtf" beginnt.

WPRichText
SetTextBuf

SetTextBuf

Siehe auch

Deklaration : procedure SetTextBuf(s : Pchar);

Diese Funktion ersetzt allen Text durch den Text im Speicherbereich auf den s zeigt. Er kann grösser sein als 64KB und entweder Text im ANSI oder im RTF Format enthalten. Hinweis: Intern arbeiten SetSelTextBuf und SetSelText mit Streams. Wenn Sie also die Daten in Formar eines Stream vorliegen haben, sollten Sie lieber gleich LoadFromStream bzw LoadFromStreamWithReader aufrufen.

WPRichText
SetSelTextBuf

SpellCheck Interface

Siehe auch Beispiel

Interface Proceduren für Textkorrektur

procedure Spell_ReplaceWord(s : string);

Ersetzt das zuletzt ausgewählte Wort.

procedure Spell_SelectWord;

Markiert das zuletzt ausgewählte Wort.

function Spell_GetNextWord : string;

Wählt das nächste Wort aus und gibt es zurück. Am Textende wird `' zurückgegeben.

procedure Spell_FromStart;

Spell_GetNextWord wird wieder am Anfang des Textes anfangen.

procedure Spell_FromCursorPos;

Spell_GetNextWord wird an der Cursorposition anfangen und nicht dort, wo es zuletzt aufgehört hat. Dies ist besonder bei nicht modalen Textkorrektur Fenstern sinnvoll.

Start SpellCheck Event

```
{ This source can be used as a template to build a
  non modal spell check dialog }

unit Wpspdlg1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, WPWinCtr;

type
  TWPSpDialog = class(TForm)
    WordList: TListBox;
    Word: TEdit;
    ReplaceAs: TEdit;
    Ignore: TBitBtn;
    Replace: TBitBtn;
    Cancel: TBitBtn;
    procedure IgnoreClick(Sender: TObject);
    procedure ReplaceClick(Sender: TObject);
    procedure CancelClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    fsFirstCall : Boolean;
    fsEditBox   : TWPCustomRtfEdit;
    procedure SetEditBox(x : TWPCustomRtfEdit);
  public
    property EditBox : TWPCustomRtfEdit read fsEditBox write SetEditBox;
  end;

TWPSpellCheckDlg = class(TComponent)
private
  dia       : TWPSpDialog;
  fsEditBox : TWPCustomRtfEdit;
public
  destructor  Destroy;
  procedure Execute;
published
  property EditBox : TWPCustomRtfEdit read fsEditBox write fsEditBox;
end;

var
  WPSpDialog: TWPSpDialog;

implementation
```

```
{$R *.DFM}

destructor TWPSpellCheckDlg.Destroy;
begin
  if dia<>nil then
  begin
        if dia.visible then dia.Close;
        dia.Free;
  end;
end;

procedure TWPSpellCheckDlg.Execute;  { Will not free the Dialogbox because
it is nnon modal ! }
begin
 if assigned(fsEditBox) then
 begin
   if dia=nil then dia := TWPSpDialog.Create(Self);
   dia.EditBox := fsEditBox;
   fsEditBox.Spell_FromCursorPos;
   dia.Show;
 end;
end;

procedure TWPSpDialog.SetEditBox(x : TWPCustomRtfEdit);
begin
  fsEditBox := x;
  if x<>nil then x.Spell_FromCursorPos;
end;

procedure TWPSpDialog.IgnoreClick(Sender: TObject);
var
 s : String;
begin
 while TRUE do
 begin
  EditBox.Spell_FromCursorPos;
  s := EditBox.Spell_GetNextWord;
  if s='' then break;
  { test Dictionary .... ##################################
    if not InDictionary(s) then
    begin
       EditBox.Spell_SelectWord;
       WordList.Strings.Assign( GuessList );
       break;
    end
```

```
    else continue;

#########################################################
######## }
   { without a dictonary any word is unknown: }
      EditBox.Spell_SelectWord;
      break;
 end;
 Word.Text      := s;
 ReplaceAs.Text := s;
 if (s='') and (fsFirstCall=FALSE) then close;
end;

{ Replace will replace the selected Text.
  If you use the procedure
  EditBox.Spell_ReplaceWord(s : string) the replacment will be done
  invisibly.
  If you have a non modal spell dialog, the use of Spell_ReplaceWord might
  cause errors if the user has changed the text meanwhile.
}

procedure TWPSpDialog.ReplaceClick(Sender: TObject);
begin
  if CompareStr(EditBox.SelText,Word.Text)=0 then  { Replace selceted Text }
     EditBox.SelText := ReplaceAs.Text;
  IgnoreClick(Sender);
end;

procedure TWPSpDialog.CancelClick(Sender: TObject);
begin
  EditBox.Spell_FromCursorPos;
  Close;
end;

procedure TWPSpDialog.FormShow(Sender: TObject);
begin
  fsFirstCall := TRUE;
  IgnoreClick(Sender);
  fsFirstCall := FALSE;
end;

end.
```

ToolBar.AddControl

Siehe auch Beispiel

Deklaration : procedure AddControl(comp : TControl);

Sie können mittels AddControl eine beliebige Komponente in die ToolBar einbinden.
Mit RemoveControl können Sie die Komponente wieder entfernen.

procedure AddControl(comp : TControl);
procedure RemoveControl(comp : TControl);

TWPToolBar

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  WPToolBar1.AddControl(Preview);
end;
```

ToolBar.GetIcon - StyleBox - GetFontBox ...

Siehe auch Beispiel

Deklaration : function GetIcon(index,group,num:Integer):TSpeedButton;

Oft werden Sie auf die in der Toolbar angezeigten Komponenten direkt zugereifen wollen. Dies ist insbesonder dann nötig, wenn Sie den Inhalt der Listboxen ändern wollen oder Eigenschaften wie 'Hint' ändern wollen.
Die folgenden Funktionen liefern die jeweiligen Object zurück:

GetIcon(index,group,num:Integer):TSpeedButton;

Übergeben Sie die gruppe und die Nummer des gewünschten Buttons. Prüfen Sie ob das Ergebnis nil ist, da der Button eventuell nicht vorhanden ist!
Bitte beachten Sie, daß sie den 'tag' des jeweiligen Objectes nicht ändern dürfen.

GetStyleBox : TComboBox;
GetStyleItems: TStrings;
GetFontBox : TComboBox;
GetFontItems : TStrings;
GetSizeBox : TComboBox;
GetSizeItems : TStrings;

TWPToolBar

```
{ define Hints for the SpeedButtons in the Toolbar }

procedure TForm1.FormCreate(Sender: TObject);
type
 TOneHint = record
   group, number : Integer;
   Hint          : String;
  end;

var
 i : Integer;
 button : TSpeedButton;

const Hints : array[1..6] of TOneHint =
((group : WPI_GR_STYLE; number : WPI_CO_Normal; hint : 'reset styles'),
 (group : WPI_GR_STYLE; number : WPI_CO_Bold; hint : 'bold'),
 (group : WPI_GR_STYLE; number : WPI_CO_Italic; hint : 'italic'),
 (group : WPI_GR_STYLE; number : WPI_CO_Under; hint : 'underlined'),
 (group : WPI_GR_STYLE; number : WPI_CO_Hyperlink; hint : 'hyperlink style'),
 (group : WPI_GR_STYLE; number : WPI_CO_StrikeOut; hint : 'strike out'));

begin
  for i:=1 to 6  do with Hints[i] do
  begin button := WPToolBar1.GetIcon(0,group,number);
        if button<>nil then
        begin button.Hint     := Hint;
              button.ShowHint := TRUE;
        end;
  end;
end;
```

ToolBar.SelectIcon,DeselectIcon,EnableIcon

Siehe auch Beispiel

Deklaration : function SelectIcon(index,group,num:Integer):Boolean;

Mit diesen Funktionen können Sie einen bestimmten SpeedButton aus/ab wählen oder ihn disablen.
Der vorige Status wird zurückgegeben.

SelectIcon(index,group,num:Integer):Boolean;
DeselectIcon(index,group,num:Integer):Boolean;
EnableIcon(index,group,num:Integer;enabled:Boolean):Boolean;

TWPToolBar

```
procedure TForm1.UpdateTableState;
begin
  with Table1 do
  begin
    WPToolBar1.EnableIcon(0,WPI_GR_DATA,WPI_CO_Next,not EOF);
    WPToolBar1.EnableIcon(0,WPI_GR_DATA,WPI_CO_ToEnd,not EOF);
    WPToolBar1.EnableIcon(0,WPI_GR_DATA,WPI_CO_Prev,not BOF);
    WPToolBar1.EnableIcon(0,WPI_GR_DATA,WPI_CO_ToStart,not BOF);
    WPToolBar1.EnableIcon(0,WPI_GR_DATA,WPI_CO_Edit,not Readonly);
  end;
end;
```

WriteStatus

Siehe auch

Deklaration : procedure WriteStatus(s:Pchar);

Schreibt eine Meldung in das StatusLabel bzw. in den Eintrag `stStatus' der StausBar.

WPRichText

Attr

Siehe auch Beispiel

Deklaration : property Attr : TAttr;

Attr ist der aktuelle Schreibmodus. Sie können Ihn durch Zuweisung einer Struktur vom Typ TAttr ändern. Wenn gerade ein Block markiert ist, wird die Änderung auf den gesamten Block angewendet.
Sie können (da es sich bei TAttr um einen Record und nicht um eine Klasse handelt) nicht direkt auf die Elemente von TAttr zugreifen. Z.B. ist include(WPRichText1.Attr.Style,afsBold) nicht möglich.

TAttr
DataStructures

```
TForm1.Button1Click(Sender : TObject)
var
  a : Tattr;
  i : Integer;
begin
  a := WPRichText1.Attr;
  a.Font := WPRichText1.GetFontNr('arial');
  a.Size := 4;
  a.Style := [afsItalic];
  i := 0;
  while i<Memo1.Lines.Count do
  begin
     WPRichText1.Attr := a;
     WPRichText1.Lines.Add(Memo1.Lines.[strings]);
     a.Size := a.Size + 2;
     inc(i);
  end;
end;
```

BackgroundColor

Siehe auch

Deklaration : property BackgroundColor : TColor;

Mit BackgroundColor können Sie - am besten zur Laufzeit des Programmes - bestimmen welche Hintergrund der neu einzufügenden Text haben soll. Wenn gerade ein Block ausgewählt ist, wird die Hintergrundfarbe des gesamten Blockes geändert.
Wenn Sie die Lines Eigenschaft zur Zuweisung von Text verwenden, wird auch die angegebene Hintergrundfarbe benutzt.

TWPRichText

CPLine

Siehe auch

Deklaration : property CPLine : string

CPLine kann dazu benutzt werden den Text der aktuellen Zeile als String zu bekommen. Es kann auch ein String zugewiesen werden. Da der Text jedoch sofort umgebrochen wird, ist im folgenden meistens string a nicht gleich string b!

```
var
  a,b : String;
begin
  a := 'Hallo';
  WPRichText1.CPLine := a;
  b := WPRichText1.CPLine;

  if CompareText(a,b)=0 then
  begin
      { reiner Zufall }
  end
end;
```

TWPRichText

CPLineNr

Siehe auch

Deklaration : property CPLineNr : Longint;

CPLineNr kann dazu benutzt werden die Cursor Zeile abzufragen oder zu setzen.

Anmerkung:
Wenn Sie auf die Daten der aktuellen Zeile zugreifen wollen (mit Vorsicht), können Sie die Arbeitsvariable Memo.active_paragraph und Memo.active_line benutzen. Siehe auch Datenstrukturen.

TWPRichText

CPPosition

Siehe auch

Deklaration : property CPPosition : Longint

CPPosition kann dazu benutzt werden die Cursor Position (in Zeichen ab Text-Anfang) abzufragen oder zu setzen.

TWPRichText

CaretDisabled

Siehe auch

Deklaration : property CaretDisabled : Boolean;

Wenn CaretDisabled = TRUE ist, wird das TWPRichText oder TDBWPRichText Object keine Schreibmarke anzeigen (obwohl sie logisch immer noch vorhanden ist).

TWPRichText

EditBox

Siehe auch Beispiel

Deklaration : property EditBox : TWPCustomRtfEdit;

EditBox gibt die TWPRichText oder TDBWPRichText componente an, die den Text entält, der durch die Komponenten TWPPagePropDlg , TWPParagraphPropDlg und TWPParagraphBorderDlg geändert werden soll.
EditBox muß definiert werden bevor der Dialog mit Execute gestartet werden kann.

TWPRichText
TWPPagePropDlg
TWPParagraphBorderDlg
TWPParagraphPropDlg

```
try
WPPagePropDlg := TWPPagePropDlg.Create(Self);
WPPagePropDlg.EditBox := WPRichText1;
WPPagePropDlg.Execute;
finally
WPPagePropDlg.Free;
end;
```

ExternHorScrollBar

Siehe auch

Deklaration : property ExternHorScrollBar : TScrollBar;

ExternHorScrollBar kann angegeben werden, wenn der Scrollbalken nicht direkt neben dem RichText fenster erstellt werden soll oder der ScrollBalken ein andere Breite haben soll.

TWPRichText

ExternVerScrollBar

Siehe auch

Deklaration : property ExternVerScrollBar : TScrollBar;

ExternVerScrollBar kann angegeben werden, wenn der Scrollbalken nicht direkt neben dem RichText fenster erstellt werden soll oder der ScrollBalken ein andere Höhe haben soll.

TWPRichText

WPRichText.Finder

Siehe auch Beispiel

Deklaration : property Finder : TWPTextFinder ;

Finder erlaubt den Zugriff auf das Object das zum Suchen im Text verwendet wird. Sie können sogar ein andere Object ableiten und ein paar Definitionen änderen. Dadurch wird es möglich auf beliebige Art zu suchen.
Indem Sie dirket auf Finder zugreifen, haven Sie wesentlich mehr Möglichkeiten zu suchen als über die Find prozedur.

Beispiel für die Benutzung:

```
procedure TForm1.MarkDatafields;
var
  newattrr: TAttr;
begin
  newattr := WPRichText1.Attr;
  newattr.Style := afsItalic;

  WPrichText1.Finder.WildCard := '*';
  WPrichText1.Finder.ToStart;
  while WPRichText1.Finder.Next('[@*@]') do
  begin
      WPRichText1.Finder.foundAttr := newattr;
  end;
end;
```

TWPRichText

```
{ Definition of the Finder Class }

WPTMatchAttr = set of (wmtFont,wmtSize,wmtBold,wmtUnderlined,wmtItalic,
 wmtHyperlink,wmtColor,wmtBkColor);
WPTMatchChar = (wpmtNormal,wpmtMatchCase);

TWPTextFinder = class
protected
  prior_lin  : PTLine;      { Searching variables. Will be restored using  }
  prior_par  : PTParagraph; { act_pos MoveTo(pos)  }
  prior_pa   : PTAttr;
  prior_pc   : Pchar;
  prior_cp   : Integer;  { Position in Line }
  prior_pos  : Longint;  { changed during search }
  {-------------------------------------------------------}
  actpos  : Longint;  { Last Position                     }
  actlen  : Integer;  { If found, then Lenght of found text }
  {-------------------------------------------------------}

  DidFind     : Boolean; { Something found          }
  endOfText   : Boolean; { End of Text and not found }
  backwards   : Boolean; { Direction to serach       }
  comp_text   : array[0..256] of Char;

  last_found_string : String;
  last_found_attr   : TAttr;
private
  function   GetFoundPos   :  Longint;  { returns the position from start
}
  function   GetFoundLen   :  Longint;  { Length of the Text (this time just on
one line!  }
  function   GetFoundText:  String;   { will try to put found text to a string
}
  procedure  SetFoundText(s : String);{ will replace the found text and keep it
as found }
  function   GetFoundAttr  :Tattr;      { will return the FIRST attr of the found
text     }
  procedure  SetFoundAttr(s :Tattr);    { will set ALL attr of the found text
}
  procedure  SetActPos(x : Longint);
protected
  function   MoveTo(pos : Longint):Longint; virtual;
  procedure  MoveNext(s : string); virtual;
  procedure  MoveBack(s : string); virtual;
  function   CompareChar(ca,cb:Char;apa,bpa:PTattr):Boolean; virtual;
{ FALSE if not equal }
```

```
  function   EndAtWordChar(c : Char):Boolean; virtual;
public
  { Searches in the text forewards: }
  function    Next(s : String): Boolean; virtual; { TRUE if found    }
  { Seraches in the text backwards: }
  function    Prev(s : string): Boolean; virtual; { search backwards }
  { Moves Serach position to End/Start of the Text }
  procedure   ToEnd;
  procedure   ToStart;
  { moves the start position od the found text }
  procedure   MoveFoundPos(diff : integer);
public      { These properties are set befor the call of WriteHeader /
WriteText }
  FHeader       : TTextHeader; { for page and font definition }
  fsWordProc    : TWPRTFText; { needed for replace }
  first_par     : PTParagraph; { The first paragraph }
  match_attr    : WPTMatchAttr; { may be () }
  match_char    : WPTMatchChar;
  wildcard      : Char; { #0 or wildcard }
  endchar       : Char; { #0 to scan to end, or Whitespace or whatever (only
forward scanning!) }
  EndAtWord     : Boolean; { if true than stop at '.,?!")' ... }
  cmp_attr   : Tattr;  { attributes to match WPTMatchAttr }
  property     Memo : TWPRTFText read fsWordProc write fsWordProc;
  property     position : Longint write SetActPos;
  property     foundPosition : Longint read GetFoundPos;
  property     foundLength : Longint read GetFoundLen;
  property     foundText: String  read GetFoundText write SetFoundText;
  property     foundAttr: Tattr read GetFoundAttr write SetFoundAttr;
end;
```

FitToWindowHorz

Siehe auch

Deklaration : property FitToWindowHorz : Boolean;

Wenn Sie hier TRUE angeben, wird der Text immer entsprechend der aktuellen Fensterbreite neu formatiert. Seiteneinstellungen werden auf dem Bildschirm ignoriert aber beim Ausdruck mittels 'Print' ausgewertet.

TWPRichText
TWPRichTextLabel

HyperLinkCursor

Siehe auch

Deklaration : property HyperLinkCursor : TCursor;

Wählen Sie die Mauszeigerform aus, die der Mauszeiger annehmen soll, wenn er über einem Text steht, der als Hyperlink markiert ist. Wenn hier crDefault steht wird die Mauszeigerposition nicht ausgewertet, was unter Umständen viel Rechenzeit sparen kann.

TWPRichText
TWPRichTextLabel

HyperLinkEvent

Siehe auch Beispiel

Deklaration : property HyperLinkEvent : THyperLinkEvent;

Tragen Sie hier die Behandlungsfunktion für Hyperlink Ereignisse ein. Sie können So z.B. an eine andere Stelle im Text oder der Datenbank springen.

WPRichText wird folgende Werte an den Handler übergeben:
text : String = der markierte Text
stamp: String ist reserviert
LineNumber: Longint = die Zeilennummer in der der markierte Text steht.

TWPRichText
TWPRichTextLabel

```
procedure TForm1.WPRichText1HyperLinkEvent(Sender: TObject; text,
  stamp: String; LineNumber: Longint);
var
  c : array[0..255] of Char;
begin
  StrPcopy(c,text);
  MessageBox(Handle,c,'Hyperlink clicked',0);
end;
```

IsLastLine

Siehe auch

Deklaration : property IsLastLine : Boolean;

IsLastLine ist TRUE wenn sich der Cursor in der letzten Zeile befindet.

TWPRichText

TWPRichTextLabel.Transparent

Siehe auch

Deklaration : property Transparent : Boolean;

Die Eigenschaft Transparent bestimmt ob ein RichTextLabel transparent gezeichnet wird. Wenn es transparent ist wird sowohl die Hintergrundfarbe des objects (Color) als auch die Hintergrundfarbe für den Text ignoriert.

TWPRichTextLabel

Language

Siehe auch

Deklaration : property Language : TWPToolLanguage;

Sie können wählen:

twpEnglish oder twpGerman

Die Auswahl wird in einer globalen variable, die in der unit WPDefs definiert is, gespeichert:

```
var
GlobalLanguage : TWPToolLanguage;
```

TWPRichText

LastError

Siehe auch

Deklaration : property LastError : ecErrCode;

unit WPDefs;

Wenn LastError ungleich ecOk ist, dann wurde die letzte Dateioperation nicht komplett durchgeführt.

TWPRichText

Lines

Siehe auch Beispiel

Deklaration : property Lines : TStrings;

Nur zur Laufzeit!

Die Eigenschaft Lines ist hevorragend dazu geeignet um den Text einer Listbox/Memo einer RichText Komponente zuzuweisen.
Im umgekehrten Fall beachten Sie bitte, daß in Lines nicht die einzelnen Zeilen enthalten sind, sondern die einzelnen Absätze. Ein Eintrag in der Eigenschaft Line ist jedoch begrenzt auf 255 Zeichen, was in vielen Fällen dazu führen wird, daß Text verloren geht. Desweiteren enthält der Text in der Lines Eigenschaft keine Formatierungen.

Ansonsten ist verhält sich Lines wie ein TStrings Object.

Wenn Sie Text zuweisen können sie den aktuellen Schreibmodus ändern um so z.B. jeden Zeile in einem anderen Modus erscheinen zu lassen.

TWPRichText

```
TForm1.Button1Click(Sender : TObject)
var
  a : Tattr;
  i : Integer;
begin
  a := WPRichText1.Attr;
  a.Font := WPRichText1.GetFontNr('arial');
  a.Size := 4;
  a.Style := [afsItalic];
  i := 0;
  while i<Memo1.Lines.Count do
  begin
     WPRichText1.Attr := a;
     WPRichText1.Lines.Add(Memo1.Lines.[strings]);
     a.Size := a.Size + 2;
     inc(i);
  end;
end;
```

LoadFormat

Siehe auch

Deklaration : property LoadFormat   : TWPLoadFormat

Mit LoadFormat legen Sie fest ob der Text in der RichText Komponente im ANSI Format oder im RTF Format im Steam oder in der Datei vorliegt.

TWPLoadFormat =
 AutomaticFormat
 LastFormat
 FormatRTF
 FormatANSI

AutomaticFormat (beste Einstellung)
Anhand der ersten 5 Zeichen wird festgestellt ob RTF oder ANSII Format vorliegt. Sind die ersten Zeichen '{\rtf' dann wird RTF Format angenommen.

TWPRichText
Load + Save

MemoryFormat

Siehe auch

Deklaration : property MemoryFormat : TWPMemoryFormat

Ändern Sie - auch zur Laufzeit - wie der Text im Speicher verarbeitet wird. Sie können fmRichText oder fmPlainText wählen. Wenn Sie von fmRichText auf fmPlainText umschalten, verlieren Sie alle Formatierungen bis auf die Einrückungen, Abstände und Textrahmen. Wenn Sie von fmPlainText nach fmRichText umschalten und Sie haben einen sehr großen Text im Speicher, kann es sein, daß das RAM nicht ausreicht. Während der Operation des Umwandelns kann der Benutzer daher mit ESC abbrechen.

Bedenken Sie: Für das fmRichText Format werden ungefähr 10Bytes zusätzlich pro Zeichen beansprucht.

TWPRichText

Modified

Siehe auch Beispiel

Deklaration : property Modified : Boolean;

Nur zur Laufzeit!

Die Eigenschaft Modified kann abgefragt werden um zu ermitteln ob sich der Text in RichText Komponente verändert hat.
Modified kann auch eine Wert zugewiesen werden.

TWPRichText

```
{ save the text }
if WPRichText.Modified then
begin
    WPRichText.Save;
    if WPRichText.LastError=ecOK then
            WPRichText1.Modified := FALSE;
end;
```

NoBlockMarking

Siehe auch

Deklaration : property NoBlockMarking : Boolean

Wenn hier der Wert TRUE steht werden Blockmarkierung nicht mehr unterstützt.

TWPRichText

OneClickHyperlink

Siehe auch

Deklaration : property OneClickHyperlink : Boolean

Wenn hier TRUE eingetragen ist, kann der Anwender mit einem anstatt einens Doppelklick einen Hyperlink aktivieren.

TWPRichText

Print_xywh_Mode

Siehe auch Beispiel

Deklaration : property Print_xywh_Mode : TWPPrint_XYWH_Mode;

Mit Print_xywh_Mode können Sie einige Optionen für Das drucken auf ein Canvas Object einstellen.

```
TWPPrint_XYWH_Mode = Set of  (    pmUsePaper,
      pmWhiteTransparent,
      AllTransparent,
      pmUseBKColor,
      pmIgnoreZooming,
      pmUsePageBreaks);
```

pmUsePaper : löscht nicht unbedruckte Bereiche
pmWhiteTransparent : die default Hintergrundfarbe wird nicht ausgewertet.
pmAllTransparent : Der Text ist transparent.
pmUseBKColor : Die default Hintergrundfarbe wird gedruckt
pmIgnoreZooming : Zooming wird nicht ausgewertet.
pmUsePageBreakes : normalerweise druckt print_xywh den angegebenen Bereich voll, mit dieser Option wird die Procedur beim nächsten Seitenumbruch den Druck beenden.

TWPRichText
Print_XYWH

```
{ Example for zoomable Print Preview }

unit PrinPr;
{ This Unit shows how to use buttons and strings in the statusbar
  and how to use the function print_xywh(Canvas, ...

  I Use the wpFastPrint Method in this unit:
    1. Call   print_xywh(Canvas, ... , wpFastPrintInit)
    2. Then call  print_xywh(Canvas, ... , wpFastPrint)   so often as you like
    3. Call   print_xywh(Canvas, ... , wpFastPrintExit)
You shoul NEVER forget to call wpFastPrintExit !
The FastPrint Method is much faster wich large Text with lots of pages to print
because initialisation is only done one time.

If you have only short Text ore more time
then you may call print_xywh(Canvas, ... , wpNormalPrint) and you dont have
to worry.

If you want to check if there is enoug space to print all lines you can call
print_xywh(Canvas, ... , wpPrintCalc). You may do this after wpFastPrintInit,
too.
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, WPRich, WPWinCtr, WPStatus,
WPDefs;

type
  TPrintPreview = class(TForm)
    WPStatusBar1: TWPStatusBar;
    Panel1: TPanel;
    PaintBox1: TPaintBox;
    procedure FormCreate(Sender: TObject);
    procedure WPStatusBar1Update(Sender: TObject);
    procedure PaintBox1Paint(Sender: TObject);
    procedure WPStatusBar1Selection(Sender: TObject; index: Integer;
      typ: TWPStatusItemCont; name: String; x, y, w: Integer;
      var Button: TMouseButton; var Shift: TShiftState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Resize(Sender: TObject);
  private
    { Private-Deklarationen }
    initialized : Boolean; { wpFastPrintInit called ? }
```

```
    Zoom    : Integer;
  public
    { Public-Deklarationen }
    last_line,first_line : Longint;
    WordProcessor : TWPRichText;
  end;

var
  PrintPreview: TPrintPreview;

implementation

{$R *.DFM}

procedure TPrintPreview.FormCreate(Sender: TObject);
begin
  WPStatusBar1.SetStringWidthIndex(0,75);
  WPStatusBar1.SetStringWidthIndex(1,75);
  WPStatusBar1.SetStringWidthIndex(1,75);
end;

procedure TPrintPreview.WPStatusBar1Update(Sender: TObject);
begin
  WPStatusBar1.SetStringIndex(0,IntToStr(first_line));
  WPStatusBar1.SetStringIndex(0,'');
  WPStatusBar1.SetStringIndex(0,'');
end;

procedure TPrintPreview.PaintBox1Paint(Sender: TObject);
begin
   if not initialized then                              { I Use the FastPrintMethod ! }
   begin WordProcessor.Print_xywh(PaintBox1.Canvas,0,0,
            PaintBox1.Width,PaintBox1.Height, first_line,wpFastPrintInit);
         initialized := TRUE;
   end;

   { Set zooming }
   WordProcessor.Zooming := Zoom;

   { Options: pmUsePaper,pmWhiteTransparent,
        pmAllTransparent,pmUseBKColor,pmIgnoreZooming,
        pmUsePageBreaks
     all in unit WPWinCtr. }

   WordProcessor.Print_XYWH_Mode := [pmUsePageBreaks];
   last_line := WordProcessor.Print_xywh(PaintBox1.Canvas,0,0,
```

```
        PaintBox1.Width,PaintBox1.Height, first_line,wpFastPrint);

   with PaintBox1.Canvas do
   begin
     Pen.Width := 1;
     Pen.Color := clBlack;
     Pen.Style := psDot;
     MoveTo(0,0);
     LineTo(PaintBox1.Width-1,0);
     LineTo(PaintBox1.Width-1,PaintBox1.Height-1);
     LineTo(0,PaintBox1.Height-1);
     LineTo(0,0);
   end;

   { The last FALSE means that output shoud be done, not only calculation }
end;

procedure TPrintPreview.WPStatusBar1Selection(Sender: TObject;
  index: Integer; typ: TWPStatusItemCont; name: String; x, y, w: Integer;
  var Button: TMouseButton; var Shift: TShiftState);
begin
   if index=3 then Close { selected END }
   else if index=2 then   { selected NEXT }
   begin
        first_line := last_line;
        PaintBox1.invalidate;
        WPStatusBar1.SetStringIndex(0,IntToStr(first_line));
   end;
end;

procedure TPrintPreview.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
   if initialized then                              { Never forget to call Exit !  }
   begin WordProcessor.Zooming := 100;
        WordProcessor.Print_xywh(PaintBox1.Canvas,0,0,
            PaintBox1.Width,PaintBox1.Height, first_line,wpFastPrintExit);
        initialized := TRUE;
   end;
end;

procedure TPrintPreview.Resize(Sender: TObject);
var
  xm, ym : Real;
  papxsiz, papysiz,lmarg,tmarg : Longint;
  x,y,w,h : Integer;
```

```
begin
  papxsiz := (WordProcessor.Memo.Header.Layout.paperw *
       Screen.PixelsPerInch) div 1440;
  papysiz := (WordProcessor.Memo.Header.Layout.paperh *
       Screen.PixelsPerInch) div 1440;
  xm := Width / papxsiz;
  ym := (Height-WPStatusBar1.Height-50) /  papysiz;
  if xm < ym then ym := xm;

  w := round(papxsiz * ym);
  H := round(papysiz * ym);

  x := (Width - Panel1.Width) div 2;
  y  := (Height -(WPStatusBar1.Height + h + 10)) div 2;

  Panel1.SetBounds(x,y,w,h);

  lmarg := (WordProcessor.Memo.Header.Layout.margl *
       Screen.PixelsPerInch) div 1440;
  tmarg := (WordProcessor.Memo.Header.Layout.margt *
       Screen.PixelsPerInch) div 1440;

  y := round(tmarg*ym);
  x := round(lmarg*ym);
  lmarg := (WordProcessor.Memo.Header.Layout.margr *
       Screen.PixelsPerInch) div 1440;
  tmarg := (WordProcessor.Memo.Header.Layout.margb *
       Screen.PixelsPerInch) div 1440;

  w :=  w - x - round(lmarg*ym);   { w= Panel1.Width }
  h :=  h - y - round(tmarg*ym);   { h= Panel1.Height }

  PaintBox1.SetBounds(x,y,w,h);

  if not Panel1.Visible then Panel1.Visible:= TRUE;
  Zoom := round(100 * ym);

  WPStatusBar1.SetString(stStatus,IntToStr(Zoom)+'%');
end;

end.
```

WPRuler.ShowAllTabPos

Siehe auch

Deklaration : property ShowAllTabPos : Boolean

Wenn ShowAllTabPos = TRUE ist, werden alle - bis zu 31 - Tabulatorpositionen angezeigt. Diejenigen, die im aktuellen Absatz benutzt werden, werden blau, die anderen grau gezeichnet.

TWPRuler

WPRuler.Units

Siehe auch

Deklaration : property Units : PTWPRulerUnit;

Sie können angeben in welcher Einheit das Lineal beschriftet wird:

Centimeter oder Inch.

Alle anderen Ereignisse und Eigenschaften werden Sie nicht benötigen, das ein TWPRichText Object die komplette Kontrolle übernimmt.

TWPRuler

WhileSelection

Siehe auch

Deklaration : property WhileSelection : TWPRulerSelectEvent;

Dieses Ereignis tritt auf während die Marker für den rechten und linken Absatzrand verschoben werden.

Die neuen Werte können Sie in
WPRuler.Header.Layout.indentFirst
WPRuler.Header.Layout.indentLeft
WPRuler.Header.Layout.indentRight
abfragen.

Beachte Sie: Das Lineal ist nur dann in Funktion, wenn es von einer WPRichText Komponente kontrolliert wird.

TWPRuler

WPRuler.XOffset

Siehe auch

Deklaration : property XOffset : Integer;

XOffset scrollt das Lineal.

TWPRuler

Resizing

Siehe auch

Deklaration : property Resizing : Integer;

Mit Resizing kann eingestelt werden (in %) wie stark der Zeichensatz im RichText object vergößert werden soll.Iim Gegensatz zu Zooming wird die Auflösung nicht geändert. Resizing verändert nicht den MapMode (Umrechnung der Koordinaten durch Windows) sondern führt einen Multiplikator ein und verursacht eine Neuformatierung des Textes.

Resizing kann dann verwendet werden, wenn Sie feststellen, daß der Text beim Drucken nicht auf eine Seite passt.

WPRichText

RtfText

Siehe auch

Deklaration : property RtfText : TWPRTFTextIO;

In der Eigenschaft RtfText können Sie zur Designzeit mittels des Eigenschafts Editors formatierten Text eingeben. Ebenfalls können Sie mit Hilfe des Eigenschafts Editors das Seitenlayout festlegen, sowie die Farbtabelle verändern. Diese Eingaben werden jedoch nur gespeichert, wenn der Text wenigstens 1 Zeichen lang ist.
Dann wird im Object Inspector der Text 'RTF text ...' angezeigt. Wenn kein Text gespeichert ist, wird 'empty.' angezeigt. Sie können den Text und alle Definitionen löschen, wenn Sie 'empty!' in das Feld eintragen.

Sie können auch durch RTF Text zuweisen. Dazu verwenden Sie die Anweisung
WPRichText1.RtfText.Assign(WPRichText2.RtfText);

praktisch ist auch die Zuweisung aus einem unsichtbaren TWPRtfStorage Objekt an ein sichtbares RichText Objekt.

z.B.:
```
WPRichTextLabel1.ButtonClick(Sender:TObject);
begin
    WPRichTextLabel1.Visible := FALSE;
    WPRichTextLabel1.RtfText.Assign( WPRtfStorage1.RtfText);
    WPRichTextLabel1.Visible := TRUE;
end;
```

Achtung: Nutzen Sie bitte die Möglichkeit des property Editors Ihren Text abzuspeichern. Sie ersparen Sich dadurch bei einer Fehlfunktion Arbeit.

TWPRtfStorage
TWPRichText
TWPRichTextLabel

The Gauge in the StatusBar

Siehe auch Beispiel

Sie ärgern Sich darüber daß es keine vernünftigen Gauges gibt, die notwendige Berechungen für Sie übernehmen? Hier ist die Lösung: Eine Fortschrittsanzeige die einen Stack mitführt, so daß jede Unter-Funktion auf den Bereich von 0-100 zugreifen kann ohne sich darum kümmen zu müssen, ob nach andere Funktionen später aufgerufen werden müssen, die noch mehr Zeit brauchen:

procedure GaugeReset;
Setzt alle Werte zurück.

function GaugeIn(end_value : Longint) : Longint;
Initialisiert eine neue Ebene auf dem Stack (0-100%). end_value muß zu diesem Zeitpunkt bereits festehen und wird 100% repräsentieren.

procedure GaugeRestrictSub(val : Integer);

Es soll eine Unterfunktion aufgerufen werden, die selbts GaugeIn aufruft. Erfahrungsgemäß wird die Unterfunktion 1/3 der Zeit der aktuellen Funktion (Gesamtzeit) brauchen. Daher muß RestrictSub(33) aufgerufen werden.

procedure GaugeInc(act_value : Longint);
Es wurden act_value Werte bearbeitet, die Fortschrittsanzeige entsprechend verschieben.

procedure GaugeOut;
Die aktuelle Ebene ist abgeschlossen. Die Fortschrittsanzeige auf 100% erhöhen. Wieviel 100% wirklich ist, hat die aufrufenden Funktion möglicherweise mit GaugeRestrictSub festgelegt!

TWPStatusBar

```
procedure DoSomething;
var
  pos : Integer;
  procedure SubRoutine1(actpos : Integer);
  var
    a,b : Longint;
  begin
    b := SizeOf(Block[actpos]);
    GaugeIn(b);
    for a:=0 to b do
      begin
        ..... do something with Block[actpos]
        GaugeInc(a);
      end;
    GaugeOut;
  end;
begin
  GaugeIn(10);
  GaugeRestrictSub(10);  you know, that one call of subroutine1
       takes 10% of the time for the DoSomething will take.
  for pos:=1 to 10 do
      subroutine1(pos);
  GaugeOut;
end;
```

GaugeValue

Siehe auch

Deklaration : GaugeValue : Integer

Dieser Wert muß zwischen 0 und 100 % liegen.

TWPStatusBar
Gauge

GaugeWidth

Siehe auch

Deklaration : GaugeWidth : Integer

Die Initialisierungsbreite des Gauge. Initialisierungsbreiten werden verwendet um die Elemnente bei Bedarf im Verhältnis richtig zu kürzen. Der längste Eintrag bleibt damit immer der längste.

TWPStatusBar
Gauge

ShowGauge

Siehe auch

Deklaration : ShowGauge : Boolean

TRUE wenn die Fortschritts Anzeige eingeschaltet werden soll.

TWPStatusBar
Gauge

ShowSizer

Siehe auch

Deklaration : ShowSizer : Boolean;

Mögen Sie Windows 4? Hier haben Sie die wunderschöne Ecke.

TWPStatusBar

Strings

Siehe auch

Deklaration : Strings : TStringList

Geben Sie Hier eine Liste von Strings an. Die Anzahl bestimmt, wieviele Felder die Statusbar haben wird. Verwenden Sie vordefinierte Strings `stStatus', `stCAPS' um auf IDs zurückgreifen zu können.
Für die Strings steht ein einfacher Property Editor zur Verfügung, der Ihnen hilft die Syntax der Strings einzuhalten. Wenn Sie Strings zur Laufzeit ändern, werden alle Einträge neu initialisiert, also alle Breiten und Inhalte verworfen. Verwenden Sie SetString um den Text eines Strings zu ändern.

Gültige IDs sind:

'stGauge','stHelp','stIndex','stOutline','stStatus','stInsert','stCaps','stNum','stXPos','stYPos','st
Date','stTime','stPage','stLine','stChanged','stFont','stStyle',
'stUnit','stTxtFmt','stName','stFileName','stPathName','stString'

TWPStatusBar

StringsAlign

Siehe auch

Deklaration : StringsAlign : TWPStatusItemAlign

Sollen die Strings ab oberen Anschlag oder am unteren angezeigt werden, oder sollen sie in der Mitte der Statusbar gezeichnet werden?

TWPStatusBar

StusBar.UseGaugeForWP

Siehe auch

Deklaration : property UseGaugeForWP : Boolean ;

Wenn diese variable TRUE ist, wird die Gauge in der StatusBar als Fortschrittsanzeige für alle RichText Elemente im aktuellen Project verwendet.
Wenn Sie selbst noch auf die Gauge zugreifen, ist das Ergebnis nicht definiert.

TWPStatusBar
Gauge

SaveFormat

Siehe auch

Deklaration : property SaveFormat : TWPLoadFormat

Mit Save Format legen Sie fest ob der Text in der RichText Komponente in ANSI Format oder im RTF Format gespeichert werden soll.

TWPLoadFormat =
 AutomaticFormat
 LastFormat
 FormatRTF
 FormatANSI

AutomaticFormat:
Beim Speichern in Datei wird anhand der Dateinendung festgestellt in welchem Format gespeichert werden soll.
Ansonsten wird in dem Format gespeichert in dem geladen wurde.

Achtung!
Wenn nicht explizit FormatANSI angegeben wurde wird meistens im RTF Format gespeichert. Dies kann beim Datenbank zugriff gefährlich sein, da so normale Memo Felder für andere Programm unlesbar werden!

LoadFromStream
SaveToStream

WPRichText.SelText

Siehe auch

Deklaration : property SelText : String

SelStart : Longint;

Gibt den Anfang der Blockmarkierung an oder verändert sie.

SelLength : Longint;

Gibt die Länge der Blockmarkierung an oder ändert sie.

SelText : String ;

Enthält die max 255 ersten Zeichen des gerade ausgewählten Textes.

Um einen Text auszuwählen können Sie auch
WPRichText.Memo.SetSelPosLen(StartPos,Länge)
verwenden.

WPRichText

WPRichText.SetModeControl

Siehe auch Beispiel

Deklaration : property SetModeControl : TWPSetModeControl;

SetModeControl erlaubt den Zugriff auf das Object zum Setzen oder Abfragen von Textattributen:
SetModeControl

Normalerweise verwenden Sie SetModeControl nicht. Sie stellt ein Interface for allem für das Texteingabe object Memo (verborgen in TWPRichText) dar, Statuswechsel anzuzeigen.

TWPRichText

```
{ Definition of the TWPSetModeControl }

TWPSetModeControl = class
private
  FBorder    : TBorder;
protected
  function  GetAlign   : TParAlign;
  procedure SetAlign(x : TParAlign);
  function  GetTabs : Longint;
  procedure SetTabs(x : Longint);
  function  GetBorderProp : PTBorder;
  procedure SetBorderProp(x : PTBorder);
  function  GetStyle   : WrtStyle;
  procedure SetStyle(x : WrtStyle);
  function  GetColor   : TTextColorType;
  procedure SetColor(x : TTextColorType);
  function  GetBKColor : TTextColorType;
  procedure SetBKColor(x : TTextColorType);
  function  GetFontName : TFontName;
  procedure SetFontName(x : TFontName);
  function  GetSize   : Integer;
  procedure SetSize(x : Integer);
  function  GetFirstIndent : Longint;
  function  GetRightIndent : Longint;
  function  GetLeftIndent : Longint;
  procedure SetFirstIndent(x : Longint);
  procedure SetRightIndent(x : Longint);
  procedure SetLeftIndent(x : Longint);
public
  property  Align      : TParAlign read GetAlign write SetAlign;
  property  BorderProp : PTBorder read GetBorderProp write SetBorderProp;
  property  Style      : WrtStyle read GetStyle write SetStyle;
  property  Color      : TTextColorType read GetColor write SetColor;
  property  BKColor    : TTextColorType read GetBKColor write SetBKColor;
  property  FontName   : TFontName read GetFontName write SetFontName;
  property  Size       : Integer read GetSize write SetSize;
  property  Tabs       : Longint read GetTabs write SetTabs;
  property  LeftIndent : Longint read GetLeftIndent write SetLeftIndent;
  property  RightIndent: Longint read GetRightIndent write SetRightIndent;
  property  FirstIndent: Longint read GetFirstIndent write SetFirstIndent;
public
  procedure AddStyle(x : WrtStyle);
  procedure DeleteStyle(x : WrtStyle);
  procedure ParagraphUpdate(Sender: TObject); virtual;
  procedure CharUpdate(Sender: TObject); virtual;
  procedure SetRulerOffset(NewXOffset : Longint); virtual;
```

```
public
   RTFText : TWPRTFTextIo; { _must_ be assigned }
end;

{ TWPRichText uses a TWPRichTextSetMode class, wich
  overrides some procedures: }

procedure TWPRichTextSetMode.ParagraphUpdate(Sender: TObject);
begin
   if RTFText.Header.Layout.Tabs <> Tabs then
   begin
      RTFText.Header.Layout.Tabs := Tabs;
      if assigned(FRuler) then FRuler.invalidate;
   end;
   if assigned(FToolBar) then
   begin
      case Align of
           parAlLeft      : FToolbar.SelectIcon(0,WPI_GR_ALIGN,WPI_CO_LEFT);
           parAlCenter    :
FToolbar.SelectIcon(0,WPI_GR_ALIGN,WPI_CO_CENTER);
           parAlBlock     :
FToolBar.SelectIcon(0,WPI_GR_ALIGN,WPI_CO_JUSTIFIED);
           parAlRight     : FToolBar.SelectIcon(0,WPI_GR_ALIGN,WPI_CO_RIGHT);
           end;
      end;
   if assigned(FRuler) then
FRuler.SetIndent(FirstIndent,LeftIndent,RightIndent);
end;

procedure TWPRichTextSetMode.CharUpdate(Sender: TObject);
var
  NewStyle   : WrtStyle;
  norm       : Boolean;
begin
  NewStyle := Style;
  if assigned(FToolBar) then
  begin
    norm := TRUE;
    if afsBold in NewStyle then
    begin  FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_BOLD);
           norm := FALSE;
    end
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_BOLD);
    if afsItalic in NewStyle then
    begin FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_ITALIC);
           norm := FALSE;
```

```
    end
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_ITALIC);
    if afsUnderline in NewStyle then
    begin FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_UNDER);
          norm := FALSE;
    end
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_UNDER);
    if afsHyperlink in NewStyle then
    begin FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_HYPERLINK);
          norm := FALSE;
    end
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_HYPERLINK);
    if afsStrikeOut in NewStyle then
    begin FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_STRIKEOUT);
          norm := FALSE;
    end
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_STRIKEOUT);
    if norm then
      FToolBar.SelectIcon(0,WPI_GR_STYLE,WPI_CO_NORMAL)
    else FToolBar.deSelectIcon(0,WPI_GR_STYLE,WPI_CO_NORMAL);

    FToolBar.UpdateSelection(wptName,FontName,0);
    FToolBar.UpdateSelection(wptSize,'',Size);
    FToolBar.UpdateSelection(wptColor,'',Color);
    FToolBar.UpdateSelection(wptBKColor,'',BKColor);
  end;
end;

procedure TWPRichTextSetMode.SetRulerOffset(NewXOffset : Longint);
begin
   if assigned(FRuler) then
   begin
      FRuler.Header  := (Parent as TWPRichText).Memo.Header;
      FRuler.XOffset := NewXOffset;
      FRuler.LeftOff := (Parent as TWPRichText).Left-FRuler.Left;
   end;
end;
```

WPRichText.StatusBar

Siehe auch

Deklaration : property StatusBar   : TWPStatusBarBasic;

TWPRichText wird diese StatusBar verwenden um einige kurze Meldungen anzuzeigen, sofern die StatusBar einen 'stStatus' string entält.

TWPRichText

WPRichText.StatusLabel

Siehe auch

Deklaration : property StatusLabel : TLabel;

StatusLabel kann für Debugging Zwecke verwendet werden. In dem angegebenen Label wird die RichText Komponente Meldungen ausgeben.

Sie können in dem angegeben Label auch Meldungen ausgeben, wenn sie die Methode WriteStatus(s:Pchar); aufrufen. Falls eine StatusBar existiert wird in dieser die Meldung sofort erscheinen, sofern ein Bereich für Status Meldungen vorgesehen ist.

TWPRichText

ToolBar.FontSizeFrom

Siehe auch

Deklaration : property FontSizeFrom : Integer;

FontSizeFrom gibt an welche Schriftgröße als kleinste in der Schriftgrößen Listbox bereitgestellt werden soll.

TWPToolBar

ToolBar.FontSizeTo

Siehe auch

Deklaration : property FontSizeTo   : Integer;

FontSizeTo gibt an welche Schriftgröße als größte in der Schriftgrößen Listbox bereitgestellt werden soll.

TWPToolBar

ToolBar.RtfEdit

Siehe auch

Deklaration : property RtfEdit : TWPCustomRtfEdit;

RtfEdit wird von TWPToolBar zur Kommunikation mit einem TWPRichText oder TDBWPRichText Object benutzt.

Wollen Sie die Kommunikation unterbrechen können Sie >nil< zuweisen. Wenn Sie andere Eingabefelder als TWPRichText verwenden sollten Sie in dem Enter Handler der jeweiligen Komponente der ToolBar.RtfEdit nil zuweisen. Dadurch ist gewährleistet, daß die ToolBar nicht auf Object wirkt, die gar nicht den Focus haben.

Es kann in MDI Objecten jedoch auch sinnvoll sein durch Abfragen dieser variable herauszufinden, welche Komponente als letze den Eingabefocus hatte.

TWPToolBar

ToolBar.ShowFont

Siehe auch

Deklaration : property ShowFont : Boolean;

Wenn diese Eigenschaft TRUE ist, wird die Listbox für die Fontauswahl die Fontnamen in dem jeweiligen Zeichensatz darstellen.

TWPToolBar

ToolBar.UpdateObject

Siehe auch

Deklaration : property UpdateObject : TComponent;

In dem Object das Sie hier angeben wird die ToolBar automatisch die Schriftart und -Farbe ändern, wenn der Anwender eine Auswahl trifft. Das Object muß vom Typ TPanel, TEdit oder TMemo sein. Wenn Sie ein anderes Object ändern wollen, können Sie einfach diese Object auf einem Panel plazieren und das Panel angeben.

TWPToolBar

ToolBar.UpdateObjectName

Siehe auch

Deklaration : property UpdateObjectName : String;

In dem Object dessen Name Sie hier angeben wird die ToolBar automatisch die Schriftart und -Farbe ändern, wenn der Anwender eine Auswahl trifft. Das Object muß vom Typ TPanel, TEdit oder TMemo sein. Wenn Sie ein anderes Object ändern wollen, können Sie einfach diese Object auf einem Panel plazieren und das Panel angeben.

TWPToolBar

ToolBar.sel_ActionIcons

Siehe auch

Deklaration : property sel_ActionIcons : TWpTbIcons2;

Wählen Sie für weche Aktionen Speedbuttons bereitgestellt werden sollen:

SelExit
SelNew
SelOpen
SelSave
SelClose
SelPrint
SelPrintSetup

Das TWPRichText Object wird zwar auf Ereignisse (bis auf New und Exit) reagieren aber wahrscheinlich nicht so wie Sie es wünschen. Sie sollten eine eigenen Handler in OnIconSelection bereitstellen um auf andere Weise auf eine Anwahl zu reagieren.

TWPToolBar

ToolBar.sel_DatabaseIcons

Siehe auch

Deklaration : property sel_DatabaseIcons: TWpTbIcons3;

Wählen Sie aus welche Datenbank Aktionen Sie benötigen:

SelToStart
SelNext
SelPrev
SelToEnd
SelEdit
SelAdd
SelDel
SelCancel
SelPost

Wenn Sie eine TDBWPRichText Komponente verwenden, wird diese auf die Auswahl des Benutzers die nötigen Befehle an die jeweilige DataSource weiterleiten.

TWPToolBar

ToolBar.sel_EditIcons

Siehe auch

Deklaration : property sel_EditIcons : TWpTbIcons4;

Wählen Sie die Buttons die Sie für das Editieren bereitstellen wollen:

SelCopy
SelCut
SelPaste
SelSelAll
SelHideSel
SelFind
SelReplace
SelSpellCheck

Beachten Sie: Wenn SelSpellCheck von Anwender angeklick wird, wird von der TWPRichText Komponente ein SpellCheckEvent generiert. Sie müssen einen Handler dafür bereitstellen. Dies macht es leichter für Sie den Überblick darüber zu behalten, welche Komponente gerade den Focus hat.

TWPToolBar

ToolBar.sel_ListBoxes

Siehe auch

Deklaration : property sel_ListBoxes : TWpTbListboxen;

Wählen Sie aus welche Listbox angezeigt werden soll:

SelFontName = Zeichsatz auswahl
SelFontSize = Schrifthöhe
SelFontColor = Textfarbe
SelBackgroundColor = Text Hintergrundfarbe
SelStyle = reserviert, z.B. für Absatzlayouts

TWPToolBar

ToolBar.sel_StatusIcons;

Siehe auch

Deklaration : property sel_StatusIcons : TWpTbIcons;

Wählen Sie welche Schriftmodus Buttons Sie benutzen wollen:

SelNormal
SelBold
SelItalic
SelUnder
SelHyperLink
SelStrikeOut

SelLeft
SelRight
SelBlock
SelCenter

Die vom Anwender getroffene Auswahl wird von dem TWPRichText Object verarbeitet.

TWPToolBar

TemplateName

Siehe auch

Deklaration : property TemplateName : string

Die hier angegebene Date wird geladen nachdem der gesamte Text in der TWPRichText componente gelöscht wurde. (durch Clear!mClear)

Sie sollten besser ein TWPRtfStorage Objekt verwenden.

WPRichText
LoadTemplate!mLoadTemplate
Clear!mClear
TWPRtfStorage

TextColors

Siehe auch Beispiel

Deklaration : property TextColors[Index : TWPTextColorsIndex]: TColor;

Mittels der Eigenschaft TextColors können Sie die Farbpalette des formatierten Textes ändern.

Beachten Sie:
Den Index der aktuelle Schreibfarbe erhalten Sie so:

```
index := WPRichText1.SetModeControl.Color;
```

oder so:

```
var a: TAttr;
a := WPRichText1.Attr;
index := a.color and 15;
```

WPRichText

```
procedure Form1.ChangeTextColorClick(Sender: TObject);
var
  ind : Integer;
begin
  ind := WPRichText1.SetModeControl.Color;
  ColorDialog1.Color := WPRichText1.TextColors[ind];
  if ColorDialog1.Execute then
  begin WPRichText1.TextColors[ind] :=  ColorDialog1.Color;
        WPRichText1.Invalidate;
        WPToolBar1.Invalidate;
  end;
end;

procedure Form1.ChangeBackgroundColorClick(Sender: TObject);
var
  ind : Integer;
begin
  ind := WPRichText1.SetModeControl.BKColor;
  ColorDialog1.Color := WPRichText1.TextColors[ind];
  if ColorDialog1.Execute then
  begin WPRichText1.TextColors[ind] :=  ColorDialog1.Color;
        WPRichText1.Invalidate;
        WPToolBar1.Invalidate;
  end;
end;
```

WPRichText.WPRuler

Siehe auch

Deklaration : property WPRuler : TWPRuler;

Geben Sie hier wenn nötig das Lineal an, das die TWPRichText / TDBWPRichText Komponente ansprechen soll.

WPRichText

WPRichText.WPToolBar

Siehe auch

Deklaration : property WPToolBar : TWPToolBar;

Geben Sie hier wenn nötig die ToolBar an, die die TWPRichText / TDBWPRichText Komponente ansprechen soll.

WPRichText

WordBox

Siehe auch

Deklaration : property WordBox : TWPRTFTextInput;

Verwenden Sie nicht diese Eigenschaft!

Sie wurde nur bereitgestellt um die Kompatibilität zur Version 1.0 und 1.1 von WordProcessing Tools zu verbessern.

Verwenden Sie statdessen die Eigenschaft Memo um auf das innere Object der RichText Fenster zuzugreifen.
Beachten Sie daß ein solcher Zugriff nur in wenigen Fällen dokumentiert ist und eventuell in späteren Versionen nicht mehr funktionieren wird.
Da die innere Funktion sehr viele Funktionen publiziert können Sie die RichText Komponente sehr beeinflussen.

WPRichText

Zooming

Siehe auch

Deklaration : property Zooming : Integer;

Mit Zooming kann eingestelt werden (in %) wie stark der Zeichensatz im RichText object vergößert werden soll. Gleichzeitig wird die (im Gegensatz zu Resizing) Auflösung verringert. Zooming verändert nicht den MapMode (Umrechnung der Koordinaten durch Windows) sondern führt einen Multiplikator ein und verursacht eine Neuformatierung des Textes.

WPRichText

PrintHeader

Siehe auch

Deklaration : property PrintHeader : TWPPrintEvent

Dieser Event wird ausgelöst um den KopfBereich einer Seite beim Ausdruck mittels der Funktion PRINT zu ermöglichen.

TWPPrintEvents werden von der Funktion Print ausgelöst, immer dann wenn der Kopf- bzw. der Fußbereich einer Seite ausgegeben werden muß. Da dem Eventhandler die Canvas übergeben wird und auch die Position wo darauf zu schreiben ist, ist es sehr einfach Seitennumerierung oder die Ausgabe von Logos zu programmieren.

TWPRichText

ToolBar.Hints

Siehe auch Beispiel

Die Eigenschaft Hints erlaubt Ihnen, die Schnellhilfen für die Buttons in der ToolBar anzupassen.
Damit die Hilfen angezeigt werden muß ShowHint = TRUE sein.

Beispiel:
WPI_Normal=Normal
WPI_Bold=Fett
WPI_Italic=Kursiv
WPI_Under=Unterstrichen
WPI_Hyperlink=Hyperlink
WPI_StrikeOut=Durchgestrichen
WPI_Left=Links Ausgerichtet
WPI_Right=Rechts Ausgerichtet
WPI_Justified=Blocksatz
WPI_Center=Zentrierter Text
WPI_Copy=Kopieren
WPI_Cut=Ausschneiden
WPI_Paste=Einfügen
WPI_SelAll=Alles auswählen
WPI_HideSel=Markierung löschen
WPI_Find=Suche
WPI_Replace=Ersetze
WPI_Spell=Wörterbuch
WPI_Exit=Programm beenden
WPI_New=Neue Datei
WPI_Open=Dokument öffnen
WPI_Save=Speichern unter ...
WPI_Close=Dokument schliessen
WPI_Print=Drucken
WPI_PrintSetup=Drucker einrichten
WPI_Next=Nächster Datensatz
WPI_Prev=Vorheriger Datensatz
WPI_Add=Neuer Datensatz
WPI_Del=Lösche Datensatz
WPI_Edit=Editiere Datensatz
WPI_Cancel=Änderungen rückgängig
WPI_ToStart=Zum ersten Datensatz
WPI_ToEnd=Zum letzten Datensatz
WPI_Post=Änderungen übernehmen

TWPToolBar