Komponente

NumEdit

für Delphi 5.0

INHALTSVERZEICHNIS

Komponente NumEdit

Die Komponente NumEdit ist von mir auf den Delphi 5.0 umgesetzt worden. Da zur es zu dieser Zeit, keine mir bekannte Komponente gab. Die es ermöglicht hätte Zahlen von rechts nach links einzugeben. Ich musste auf der Delphi TEdit - Komponente einen Zusatz vorsehen, der dieses überhaupt ermöglicht.
Da es hierbei um eine Ableitung von einer Standard – Komponente handelt, sollte diese ohne Problem bei Ihnen funktionieren. Es sind aber weitere Zusatzfunktionen hinzu gekommen die in dem Handbuch beschreiben werden. Alles andere wird dabei von mir ausgeklammert, da dieses in der Hilfe von Delphi ja vorhanden ist.

Die meisten Funktionen sind in der Protected Struktur versteckt und sollten auch nicht sichtbar gemacht werden durch eine Ableitung. Man muss hier eigentlich nichts ändern.

Die Aufschlüsselung der Komponente wird in den nochfolgenden Seiten erklärt.

Private – Funktion

Wie das Wort Private schon ausdrückt, sind hier alle Variablen hinterlegt, die man tunlichst nicht direkt ändern sollte.

FAbout	: TNumEditAboutInfo;	
FNumRETURN	: boolean;	Ture = Kommastelle prüfen, sowie setzen.
FPointShifting	: integer;	Anzahl der Kommastellen
FFloatvalue	: extended;	Real-Wert von NumEdit
FKomma_Point	: word;	Position von der Kommastelle
FCursor_Pos	: word;	Position vom Cursor
FCursor_MaxPos	: word;	Max Position vom Cursor
FPoint	: integer;	Kommastelle festgelegt. Maximal 255

Die Methode Set_PointString setzt die Kommastelle und die nötigen Stellen hinter dem Komma. Je nach Einstellung von PointShifting

```
function    Set_PointString( Stringdate : string) : string;
```

Die Methode SetPointShifting überprüft ob die aktuellen Werte gesetzt sind.

```
procedure SetPointShifting( Value : Integer );
```

Die Methode SetNumRETURN überprüft ob eine andere Komponente NumEdit mitteilt, das ein Windows-Fenster gelöscht wird und jetzt alle aktuellen Werte überprüft werden müssen.

ACHTUNG !!! Das muss jeder NumEdit – Komponente einzeln mitgeteilt werden.

```
procedure SetNumRETURN( Value : boolean );
```

Protected – Funktion

Botschaftsbehandlung für Ihre Komponenten modifizieren, müssen Sie sich über die gewünschte Wirkung genau im klaren sein.

```
procedure WndProc(var Message: TMessage); override;
```

Das Ereignis KeyPress wird ausgelöst, sobald der Benutzer eine einzelne Zeichentaste drückt. Auch hier muss man sich klaren was erreicht werden soll.

```
procedure KeyPress ( var Key: char ); override;
```

Das Ereignis KeyDown wird ausgelöst, sobald der Benutzer eine Taste drückt. Diese wurde auch verändert für NumEdit.

```
procedure KeyDown ( var Key: Word; Shift: TShiftState ); override;
```

Public – Funktion

Hier sind alle Funktionen und Variablen die für die Öffentlichkeit zugänglich sind.

Die Methode erzeugt und initialisiert ein TNumEdit-Objekt.

constructor Create(AOwner:TComponent); override;

Die anderen Methode sind so geblieben wie im Orginal.

Published – Funktion

Endlich kommt der Objektinspektor zum Zug. Alles was sichtbar sein soll, in der Kompnenten wird hier angezeigt.

Info über die Komponente

property About : TNumEditAboutInfo read FAbout write FAbout;

Die Eigenschaft NumRETURN überprüft ob eine andere Komponente NumEdit mitteilt, das ein Windows-Fenster gelöscht wird und jetzt alle aktuellen Werte überprüft werden müssen. ACHTUNG das muss jedem NumEdit einzeln mitgeteilt werden.

property NumRETURN : boolean read FNumRETURN write SetNumRETURN;

Mit der Eigenschaft PointShifting, wird die Kommastelle festgelegt.

property PointShifting : integer read FPointShifting write SetPointShifting;

Mit der Eigenschaft Floatvalue wird der ermittelte Wert an die Komponente übergeben.

property Floatvalue : Extended read FFloatvalue write FFloatvalue;

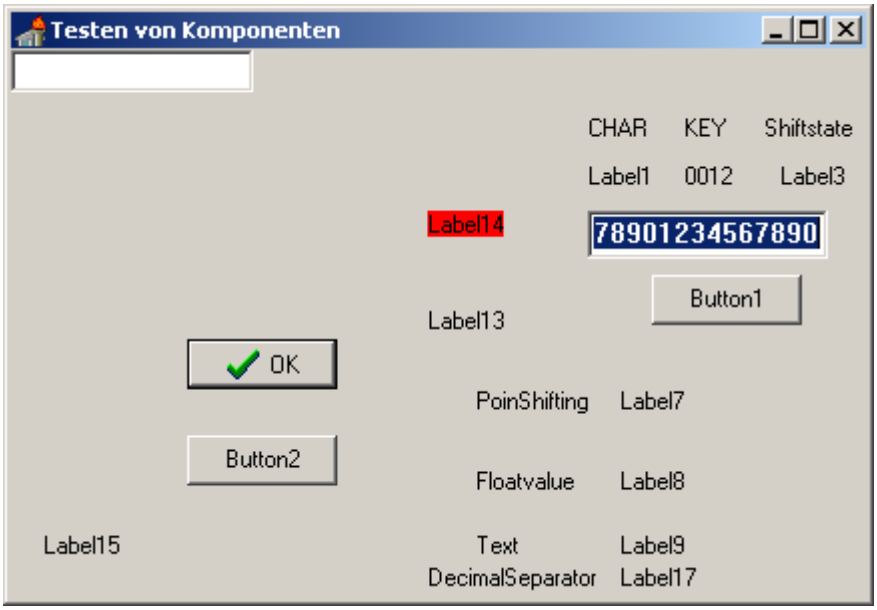
Kompnenten – Fehler

Standard Fehlerarten können bei der Verwendung der Komponente auftreten.

Beispiel – Anwendung

Eine Beispielanwendung ist der Komponente beigelegt und sollte die grundsätzlichen Funktionen erklären.

Es wurden ein Paar NumEdit – Komponenten und Buttons verwendet.



Das von mir geschriebene Beispiel, ist eine klassische Komponenten – Testumgebung. Wenn die Komponente in der Komponentenleiste eingebunden ist, so muss man die SELF – Variante nicht machen.

Sie auf den folgenden Seiten das Beispiel.

Und viel Spaß beim ausprobieren.

Ich übernehme keine Haftung bei der Fehlerhaften Bedienung der Komponente. Sollte es durch die Komponente, einen gravieren Schaden auf Ihren PC geben, so kann ich auch hier nicht Haften, da nicht ausgeschlossen werden kann, das andere Programme die Sie verwenden, hierfür die Ursache sind.

Beispiel – Quellcode Teil 1

```
unit Testkomp_Unit;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  NumEdit, StdCtrls, Buttons, ExtCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    BitBtn1: TBitBtn;
    Label14: TLabel;
    Label15: TLabel;
    Button2: TButton;
    Label16: TLabel;
    Label17: TLabel;
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure Edit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);


  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;
  xNum : TNumEdit;

implementation

{$R *.DFM}

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
     label1.caption := inttostr(ord(key));
end;
```

Beispiel – Quellcode Teil 2

```
procedure TForm1.Edit1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
  if assigned(OnExit) then
     begin
          beep;
          sleep(1000);
          beep;
          sleep(1000);
          beep;
     end;

  label2.caption := inttohex( key, 4);
  if key = VK_HOME then beep;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
     xNum := TNumEdit.create(self);
     XNum.parent := self;
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
     XNum.NumRETURN := true;
     setfocus;
     label7.caption := inttostr( xNum.PointShifting );
     label8.caption := FormatFloat( '0.00', xNum.Floatvalue );
     //Label9.caption := xNum.text;
     label17.caption := DecimalSeparator;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
     xNum.free;
end;

end.
```