

29 Einblick in Delphi-Pascal

Gliederung

29.1 Vorbemerkungen	2
29.2 Die Delphi-Fenster	X
29.3 Einleitendes Beispiel	X
29.4 xxxxxxxxxxxxxx	X

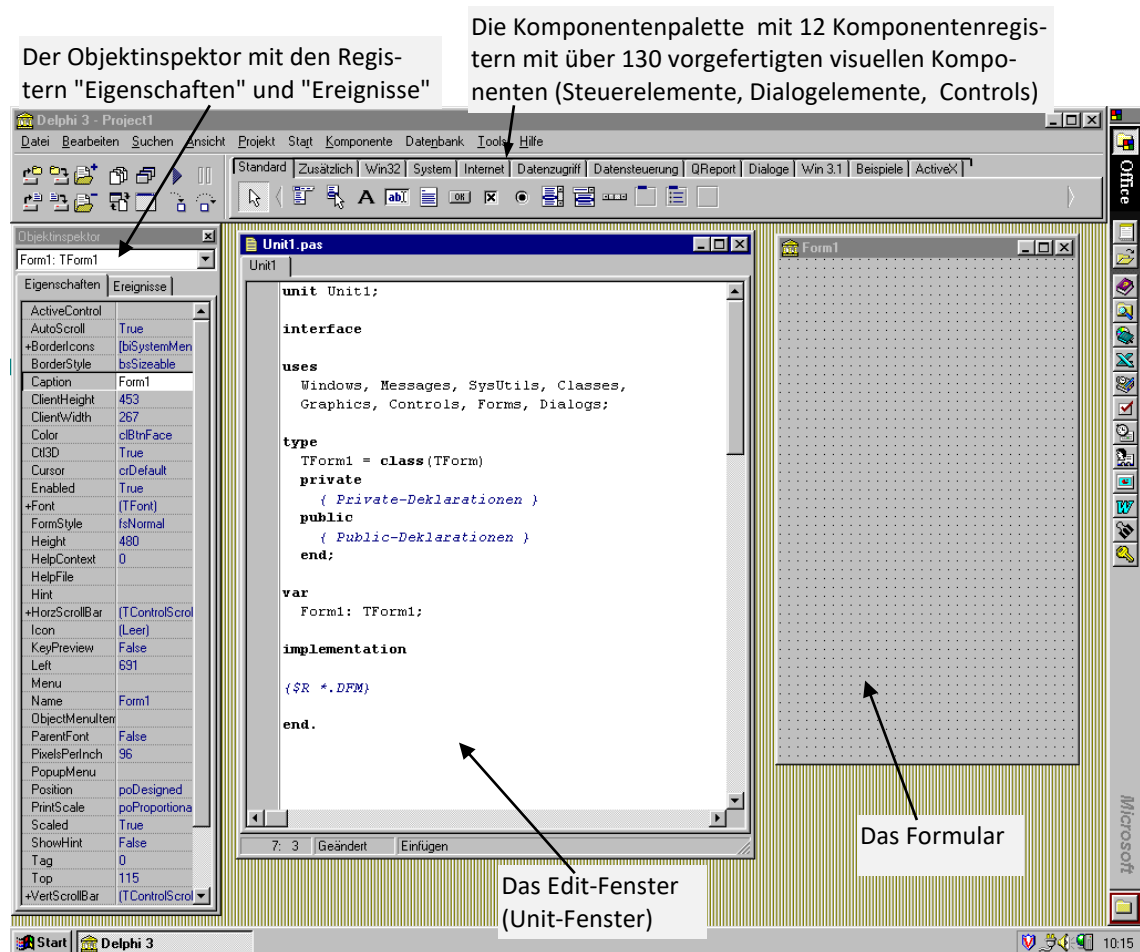
Skriptum noch nicht fertig.
Delphi-Demos finden in den Lehrveranstaltungen statt.

26.1 Vorbemerkungen

Delphi ist von Borland als visuelles Pascal für Windows konzipiert worden und ist die Reaktion von Borland auf "Visual Basic VB" von Microsoft.

29.3 Die Delphi-Fenster

Die folgende Abbildung zeigt den typischen Delphi-Bildschirm zu Beginn eines neuen Projektes. In der Abbildung befinden sich nach den Menüleisten links der Objektinspektor, der mit "F11" oder "Ansicht/Objektinspektor" angezeigt wird. In der Mitte befindet sich das Editfenster (Unit-Fenster) mit dem Programmcode, Standarddateiname "Unit1.PAS", rechts ist das noch leere Formular, Standardname "Form1" angezeigt. Beim Anlegen eines neuen Projektes enthält das Unitfenster bereits einen Standard-Programmcode, der dann entsprechend dem weiteren Vorgehen erweitert wird.



Das Editfenster liegt beim Start von Delphi hinter dem Formularfenster. Mit "F12" kann zwischen den beiden Fenstern umgeschaltet werden. Es empfiehlt sich für die ersten Beispiele die Fenster in der gezeigten Weise aufzuziehen, daß das Editfenster und das Formular nebeneinander stehen. Ein vollständiges Delphi-Projekt besteht aus einer oder mehreren Unit-Dateien ".PAS", einer oder mehreren Formulardateien (*.DFM" (Delphi-Formular), einer Projektdatei "*.DPR" (Delphi-Project, Quelltext), einer Ressourcendatei "*.RES", dem ausführbarem Programm "*.EXE" und evtl. noch weiteren Dateien. Es empfiehlt sich, für jedes Projekt einen eigenen Ordner anzulegen. Das Anlegen

eines neuen Ordners kann in der Delphi-Sitzung mit "Datei/Speichern unter ..." nach der Vorgängerordnerauswahl mit der rechten Maustaste und "Neu/Ordner" angelegt werden oder via Taskleiste mit "Programme/Windows Explorer/Datei/Neu ..." im Vorgänger-Ordner.

Der Standardcode der Unit "Unit1.PAS":

```
unit Unit1;  
  
interface  
  
uses  
    Windows, Messages, SysUtils, Classes,  
    Graphics, Controls, Forms, Dialogs;  
  
type  
    TForm1 = class(TForm)  
    private  
        { Private-Deklarationen }  
    public  
        { Public-Deklarationen }  
    end;  
  
var  
    Form1: TForm1;  
  
implementation  
  
{ $R *.DFM }  
  
end.
```

Der Standardcode des Projekts "Unit1.PRJ":

```
program Project1;  
  
uses  
    Forms,  
    Unit1 in 'Unit1.pas' {Form1};  
  
{ $R *.RES }  
  
begin  
    Application.Initialize;  
    Application.CreateForm(TForm1, Form1);  
    Application.Run;  
end.
```

Der Projekt kann mit "Ansicht/Projekt-Quelltext" angezeigt werden. Bei einfachen Anwendungen braucht der Projektcode nicht händisch bearbeitet werden.

Zur Komponentenpalette

Als Oberbegriff für Steuerelemente, Dialogfelder, Fenster und Anwendungselemente wird in Delphi der Begriff "Komponente" benutzt. Das Formular ist eine spezielle Komponente, die andere Komponenten enthält. Die meisten Komponenten sind sichtbar; es gibt aber auch unsichtbare Komponenten, wie z.B. Datenbanken.

Die Komponentenpalette enthält 12 Register mit über 330 vorgefertigten Komponenten, nach Themen in den Registern (Seiten) geordnet. Die "alltäglichen" Komponenten befinden sich im ersten Register, bezeichnet mit "Standard" und im Register "Zusätzlich". Das Register "Standard" wird beim Delphi-Start als aktuelles Register angezeigt.

Nach ca. 2 Sekunden Verweildauer des Cursors auf einem Symbol erscheint der Namen des Steuerelementes.

Register	Bedeutung
1. Standard	Standard-Windows-Steuerelemente, Menüs
2. Zusätzlich	Angepaßte Steuerelemente
3. Win32	Allgemeine Elemente für 32-Bit-Windows (ab Windows 95)
4. System	Für Zugriff auf Systemebene
5. Internet	Für Internet-Zugriff
6. Datenzugriff	Nicht-visuelle Komponenten für Zugriff auf Datenbanken, Tabellen, Abfragen, Berichte
7. Datensteuerung	Visuelle Komponenten zur Datenbereitstellung
8. QReport	QuickReport. Zum Erstellen von eingebetteten Berichten
9. Dialoge	Allgemeine Dialogelemente von Windows
10. Win3.1	Komponenten, kompatibel zu Delphi-1.0-Projekten (Windows 3.1)
11. Beispiele	Beispiele für angepaßte Komponenten
12. ActiveX	

1. Die Steuerelemente der Registers "Standard"

Standard Zusätzlich Win32 System Internet Datenzugriff Datensteuerung QReport Dialoge Win 3.1 Beispiele ActiveX		
Name und Symbol	Visuell	Erklärung
 MainMenu	N	Nichtsichtbare Komponente zum Erstellen von Menüleisten und Drop-Down-Menüs
 PopupMenu	N	Nichtsichtbare Komponente für Erstellen von lokalen Pop-Up-Menüs, die von Formularen und Steuerelementen benutzt werden können, wenn der Anwender die rechte Maustaste drückt.

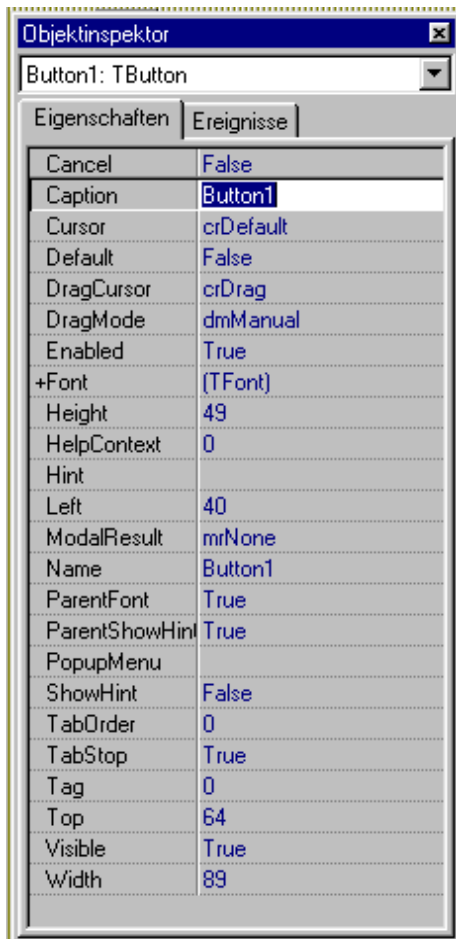
Label 	J	
Edit 	J	
Memo 	J	
Button 	J	
CheckBox 	J	
RadioButton 	J	
ListBox 	J	
ComboBox 	J	
ScrollBar 	J	
GroupBox 	J	
RadioGroup 	J	
Panel 	J	

2. Die Steuerelemente der Registers "Zusätzlich"

Standard Zusätzlich Win32 System Internet Datenzugriff Datensteuerung QReport Dialoge Win 3.1 Beispiele ActiveX            		
Name und Symbol	Visuell	Erklärung
BitBtn 	J	

SpeedBtn 	J	
MaskEdit 	J	
StringGrid 	J	
DrawGrid 	J	
Image 	J	
Shape 	J	
Bevel 	J	
ScrollBar 	J	
CheckListBox 	J	
Splitter 	J	
StaticText 	J	
Chart 	J	

Zur Registerkarte "Eigenschaften" des Objektinspektors:



Jede Komponente (Steuerelement) besitzt eine Anzahl von Eigenschaften. Damit kann z.B. Größe und Aussehen von Steuerelementen festgelegt werden. Viele Komponenten besitzen eine Anzahl von gemeinsamen Eigenschaften. Es gibt aber auch erhebliche Unterschiede. Um die möglichen Eigenschaften zu sehen, klickt man im Formular das gewünschte Steuerelement an und wählt dann im Objektinspektor die Registerkarte "Eigenschaften". Man kann aber auch in der Combobox des Objektinspektors das gewünschte Steuerelement auswählen. In den meisten Fällen sind die Eigenschaften mit Standardwerten vorbelegt.

Die Eigenschaften der obersten Komponente, des Formulars, sind aus der ersten Abbildung zu sehen. Die Liste ist so lang, daß auch bei vollem Fenster Bildlaufleisten bemüht werden müssen, um alle Eigenschaften dieser Komponente anzeigen zu können.

In der folgenden Tabelle werden stellvertretend für viele anderen die Eigenschaften des Steuerelements "TButton" dargestellt:

Eigenschaft	Bedeutung
Cancel	Mit dieser Eigenschaft kann eine Schaltfläche zur Abbruch-Schaltfläche gemacht werden. Voreinstellung ist "False". Bei "True" wird die Ereignisbehandlungsroutine OnClick (siehe "Ereignisse") auch ausgeführt, wenn der Benutzer die ESC-Taste drückt. Weitere Details siehe Online-Hilfe..
Caption	Bei den meisten Steuerelementen ist Caption der Titel (nicht der logische Name, siehe Eigenschaft "Name") und wird als Beschriftung für das Steuerelement eingeblendet. In diesem Fall sollte man den Standardeintrag mit einem sinnvollen Eintrag überschreiben. Wenn man einem Zeichen im Caption-String das Zeichen "&" (kaufmännisches Und), dann kann dieser Zeichen in Verbindung mit der Alt-Taste als Tastenkürzel (Hotkey) benutzt werden. Das Hotkey-Zeichen wird dem Benutzer durch Unterstreichung visualisiert.
Cursor	Voreinstellung "CrDefault".
Default	
DragCursor	

DragMode	
Enabled	
Font	
Height	
HelpContext	
Hint	
Left	
ModalResult	
Name	
ParentFont	
ParentShowHint	
PopUpMenu	
ShowHint	
TabOrder	
TabStop	
Tag	
Top	
Visible	
Width	

Zur Registerkarte "Ereignisse" des Objektinspektors



Die möglichen "Ereignisse" sind bei den einzelnen Komponenten (Steuerelemente) ebenfalls verschieden.

Hier werden die Eigenschaften der Komponente "**TButton**" dargestellt, die sich aber auch bei vielen anderen Komponenten finden und somit einigermaßen typisch sind. Es kommen aber durchaus bei anderen Komponenten noch weitere mögliche Ereignisse hinzu oder es werden welche entfernt.

Zur Eigenschaft "**OnClick**": Hier kann festgelegt werden, was bei einem Mausklick des Anwenders auf das gewählte Steuerelement geschehen soll (oder mit der Enter-Taste, wenn das Steuerelement fokussiert ist). Es kann der Name einer (selbst zu schreibenden) Prozedur eingetragen werden. Der Unit-Quelltext wird dann sofort automatisch aktualisiert und der Kopf der Prozedur eingetragen. Durch Doppelklick auf das Eingabefeld von "OnClick" oder durch Auswahl aus dem zugehörigen Untermenü wird der Standardvorschlag "Button1Click" für den Button1, "Button2Click" für

den Button 2 usw. eingetragen. Wenn man die Standardvorschläge übernimmt, erscheint

im Implementierungsteil der Unit die folgende "Roh-Prozedur". Gibt man in das Eingabefeld von "OnClick" einen eigenen Namen ein, dann erscheint dieser im Prozedurnamen an Stelle von "Button1Click".

```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;
```

Tabellarische Zusammenstellung der Ereignisse der Komponente TButton:

Ereignis	Ereignisprozedur bei Standardeinstellung ("begin" und "end" werden hier aus Platzgründen nicht mehr ausgedruckt). Bemerkungen
OnClick	Ereignis tritt ein, wenn das Steuerelement angeklickt wird oder die Enter-Taste bei fokussiertem Steuerelement betätigt wird. procedure TForm1.Button1Click(Sender: TObject);
OnDragDrop	Zu "Drag and Drop" (Ziehen und Ablegen): Es werden Aktionen festgelegt, wenn der Anwender ein Objekt ablegt. Details siehe Online-Hilfe. procedure TForm1.Button1DragDrop(Sender, Source: TObject; X, Y: Integer);
OnDragOver	Zu "Drag and Drop" (Ziehen und Ablegen): Es werden Aktionen festgelegt, wenn ein Anwender irgend etwas über das Steuerelement zieht. Details siehe Online-Hilfe. procedure TForm1.Button1DragOver(Sender, Source: TObject; X, Y: Integer; State: TDragState; var Accept: Boolean);
OnEndDrag	Zu "Drag and Drop" (Ziehen und Ablegen): Es werden Aktionen festgelegt, wenn der Anwender ein Steuerelement über ein anderes zieht und ablegt, es aber von diesem nicht akzeptiert wird. Details siehe Online-Hilfe. procedure TForm1.Button1EndDrag(Sender, Target: TObject; X, Y: Integer);
OnEnter	Ereignis wird ausgelöst, wenn das Steuerelement den Eingabefokus erhält. Details siehe Online-Hilfe. procedure TForm1.Button1Enter(Sender: TObject);
OnExit	Wird aktiviert, wenn der Fokus von einem Steuerelement zu einem anderen wechselt. procedure TForm1.Button1Exit(Sender: TObject);
OnKeyDown	Ereignis wird ausgelöst, sobald der Benutzer eine Taste drückt. Besonders wichtig beim Steuerelement TEdit. Siehe auch Ereignisse "OnKeyPress" und "OnKeyUp". procedure TForm1.Button1KeyDown(Sender: TObject; var Key: Word; Shift: TShiftState); Die Variable "Key" liefert die Ordnungsnummer des Zeichens, nicht das Zeichen als Char selbst, im Gegensatz zum Ereignis "OnKeyPress". Es können alle Tasten, Funktionstasten und Tastenkombinationen erfaßt werden. Für nichtnumerische Zeichen siehe Online-Hilfe. Der Parameter Shift ist eine (Pascal-) Menge mit folgenden Elementen: ssShift Taste UMSCHALT (Umsch, Shift) ist gedrückt. ssAlt Taste ALT ist gedrückt. ssCtrl Taste STRG ist gedrückt. ssLeft Linke Maustaste ist gedrückt. ssMiddle Mittlere Maustaste ist gedrückt. ssDouble Doppel-Mausklick
OnKeyPress	Ereignis wird ausgelöst, sobald der Benutzer eine Taste drückt. Besonders wichtig beim Steuerelement TEdit. Siehe auch Ereignis "OnKeyDown" und "OnKeyUp". procedure TForm1.Button1KeyPress(Sender: TObject; var Key: Char); Im Gegensatz zum Ereignis "OnKeyDown" hat der Parameter "Key" den Datentyp Char. Tasten ohne ASCII-Char-Wert (z.B. Umschaltasten, Cursortasten, Funktionstasten, ..) gene-

	rieren kein Ereignis OnKeyPress. Shift-Tastenkombinationen (z.B. Shift+A, Umsch+A) führen nur zu einem Ereignis, wenn die Feststelltaste deaktiviert ist. Sollen Nicht-ASCII-Tasten oder andere Tastenkombinationen behandelt werden, müssen die Ereignisse "OnKeyDown" oder "OnKeyUp" an Stelle von "OnKeyPress" benutzt werden.
OnKeyUp	Ereignis OnKeyUp wird ausgelöst, sobald der Benutzer eine gedrückte Taste losläßt. Sonst wie Ereignis "OnKeyDown". procedure TForm1.Button1KeyUp(Sender: TObject; var Key: Word;
OnMouseDown	Ereignis tritt ein, wenn der Benutzer eine Maustaste drückt, während sich der Mauszeiger über dem Steuerelement befindet. procedure TForm1.Button1MouseDown(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer); Die Routine kann auf das Drücken der linken, der rechten oder der mittleren Maustaste und auf das Drücken einer Sondertaste zusammen mit einer Maustaste reagieren. Als Sondertasten gelten die Tasten UMSCHALT, STRG und ALT. X und Y geben die Pixel-Koordinaten des Mauszeigers im Steuerelement. Zum Parameter "Shift" siehe Ereignis "OnKeyDown". Für den Parameter "Button" stehen die vordefinierten Konstanten "mbLeft" (Mouse Button Left), "mbRight" und "mbMiddle" zur Verfügung.
OnMouseMove	Ereignis tritt ein, wenn sich der Mauszeiger über dem Steuerelement befindet. procedure TForm1.Button1MouseMove(Sender: TObject; Shift: TShiftState; X, Y: Integer); Zum Parameter "Shift" siehe Ereignis "OnKeyDown". X und Y sind die Pixel-Koordinaten des Mauszeigers im Steuerelement.
OnMouseUp	Ereignis tritt ein, wenn der Benutzer eine Maustaste losläßt, während sich der Mauszeiger über dem Steuerelement befindet, sonst wie Ereignis "OnMouseDown". procedure TForm1.Button1MouseUp(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
OnStartDrag	Ereignis tritt ein, wenn der Benutzer beginnt, das Steuerelement oder ein darin befindliches Objekt mit der linken Maustaste zu ziehen. Details siehe Online-Hilfe. procedure TForm1.Button1StartDrag(Sender: TObject; var DragObject: TDragObject);

29.3 Einleitende Beispiele

1. Beispiel: Ordner "DP-29031"

```

unit Unit1;

interface

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

type

```

```

TForm1 = class (TForm)
    Label1: TLabel;           { automat. Eintrag }
    Label2: TLabel;           { automat. Eintrag }
    Label3: TLabel;           { automat. Eintrag }
    Button1: TButton;          { automat. Eintrag }
    procedure Beenden(Sender: TObject); { automat. Eintrag }
private
    { Private-Deklarationen }
public
    { Public-Deklarationen }
end;

var
    Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Beenden(Sender: TObject); { automat. Eintrag }
begin
    Close; { Händischer Eintrag, kha }
end;

end.

```

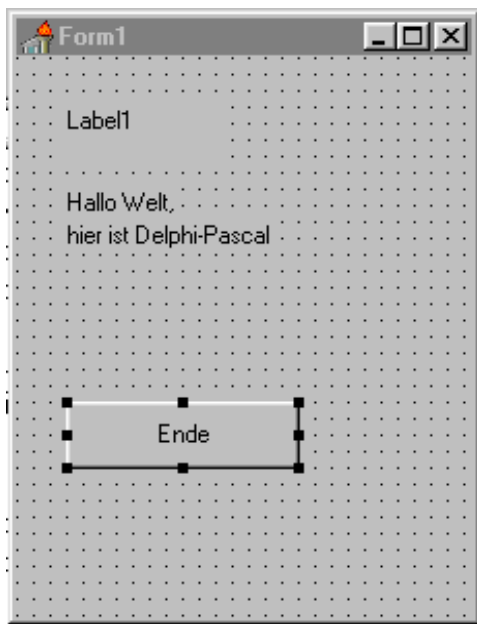
Die Erstellung des Programms

- Objekt "Label1" in Formular erstellen. Dazu Symbol "A" anklicken und dann im Formular an gewünschter Stelle ein Fenster aufziehen. Im Objektinspektor Register "Eigenschaften" anklicken, wenn nicht bereits angezeigt. Es wird das aktuelle Objekt angezeigt, hier "Label1". In der Eigenschaft "Caption" steht der Standardtext "Label1". Ausnahmsweise so belassen. Der Quellcode wird automatisch erweitert, hier mit dem Eintrag "Label1: TLabel;"
- Objekt "Label2" erstellen ähnlich wie vorher, nur den Standardtext in "Caption" durch den Text "Hallo Welt," überschreiben. Die Anzeige im Formular ändert sich zugleich.
- Objekt "Label3 erstellen ähnlich wie vorher, nur den Standardtext in "Caption" durch den Text "hier ist Delphi-Pascal" überschreiben.
- Objekt "Button1" (Schaltfläche für Beenden) erstellen. Bei "Caption" den Text "Beenden" eintragen. Dann Register "Ereignisse" des Objektinspektors anklicken und bei "OnClick" den selbstgewählten Namen (hier ebenfalls "Beenden") für die Prozedur eintragen, die bei Mausklick auf die Schaltfläche aufgerufen werden soll. Der Quelltext ist jetzt im Implementierungsteil zusätzlich automatisch um die noch leere Prozedur "procedure TForm1.Beenden(Sender: TObject)" erweitert worden. Im vorliegenden Fall ist lediglich händisch die gewünschte Reaktion einzutragen: Die Methode "Close" (Methode von Komponente "TForm"). Nebenbei: Die Standardprozedur "Close()" für Dateischließen in Turbo-Pascal kann auch in Delphi weiterhin für

Dateischließen benutzt werden; um Verwechslungen zu vermeiden, sollte man aber für diese Zwecke in Delphi die neue Prozedur "CloseFile()" benutzen.

- Das Programm kann im Editfenster durch Klick auf das Symbol 8 kompiliert und ausgeführt werden. Andere Möglichkeiten: "F9" oder über Menü mit "Start/Start". Mit "Strg+F9" wird das Projekt nur kompiliert, aber nicht ausgeführt. Beim Abbruch muß das Programm mit "Strg+F2" oder "Start/Programm zurücksetzen" zurückgesetzt werden.

Das Formular:



Die Informationen über das Formular (Größe, Farbe, Beschriftungen, Ereignisse usw.) werden nicht im Pascal-Quelltext, sondern in der (Binär-) Formulardatei "UnitX.DFM" gespeichert. Durch Klick mit der rechten Maustaste auf das Formular zur Entwicklungszeit wird ein Kontextmenü angezeigt, in dem u.a. enthalten ist "Ansicht als Text". Damit kann der Textextrakt der Formulardatei angezeigt, editiert und gedruckt werden. Die Syntax der Formulardatei ist aber nicht ganz Pascal-like.

2. Beispiel: Ordner "DP-29032"

3. Beispiel: Ordner "DP-29033"

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs,
  StdCtrls, Grids;

type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    Button1: TButton;
    Edit1: TEdit;
    Button2: TButton;
    procedure Klick(Sender: TObject);
    procedure KlickEnde(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Klick(Sender: TObject);
const
  ZeMax = 6;
  SpMax = 4;
  s = '      Huber      ';

var
  Ze, Sp: Byte;
  Summe: Word;
  x: Real;
  xStr: string[20];

begin
  { "StringGrid" befindet sich im Register "Zusätzlich" }
  { Achtung: Bei "StringGrid.Cells" zuerst Spalten,      }
  { dann Zeilen. Beginn mit 0, 0 für Tabellenkopf      }
  for Sp := 1 to SpMax do { "Break" ist zulässig }
    StringGrid1.Cells[Sp, 0] := 'Sp ' + IntToStr(Sp);
  for Ze := 1 to ZeMax do
    StringGrid1.Cells[0, Ze] := 'Ze ' + IntToStr(Ze);
  for Ze := 1 to ZeMax do
```

```

    for Sp := 1 to SpMax do
        StringGrid1.Cells[Sp, Ze] := IntToStr(Sp);
    { In Delphi gibt es endlich die Trimm-Funktion }
    StringGrid1.Cells[0, 0] := '' + Trim(s) + '';

    Summe := 0;
    for Ze := 1 to ZeMax do
        for Sp := 1 to SpMax do
            Summe := Summe + StrToInt(StringGrid1.Cells[Sp, Ze]);
        Edit1.Text := 'Summe: ' + IntToStr(Summe);
    end;

procedure TForm1.KlickEnde(Sender: TObject);
begin
    Close;
end;

end.

```

Das zugehörige Formular:

"Huber"	Sp 1	Sp 2	Sp 3	Sp 4
Ze 1	1	2	3	4
Ze 2	1	2	3	4
Ze 3	1	2	3	4
Ze 4	1	2	3	4
Ze 5	1	2	3	4
Ze 6	1	2	3	4

Starten

Summe: 60

Beenden

29.4 xxxxxxxxxxxx

....

