Lazarus – Globale Daten im Projekt speichern

Vorwort

Der nachfolgende Artikel geht auf eine Frage von „kralle“ beim „2. Norddeutschen Lazarustreffen“ zurück.

Frage

Wo würdet Ihr globale Daten in einem Projekt speichern?

Gegenfrage

Was meinst Du mit „Global“?
Es gibt mehrere Möglichkeiten die „Reichweite“ von „globalen“ Variablen betreffend.

1. Nur Zugriff durch alle Prozeduren und Funktionen der aktuellen Unit?

Was bedeutet das?
Man erstellt zum Beispiel ein Formular und platziert zwei Button auf diesem Formular.
Nun erstellt man für die beiden Button jeweils die „OnClick“-Procedure.
Wenn man jetzt in beiden Prozeduren auf die gleichen Werte zugreifen möchte oder muss, dann muss man sich über legen, wie man die Werte zur Verfügung stellt.
Wann die Werte zum Beispiel von einem Edit-Feld kommen, könnte man in beiden Prozeduren natürlich den Wert des Feldes einfach erneut auslesen, aber was wenn sich der Wert in der Zwischenzeit geändert hat?
Da macht das Programm vielleicht etwas was man es nicht machen sollte.

In diesen Fall erstellt man eine „globale Variable“ im Implementations-Teil der Unit

```
implementation
var
  MyGlobalVariable : string;  // Zugriff nur innerhalb der aktuellen Unit
```

und greift in den Prozeduren jeweils aus die Variable zu.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption:=MyGlobalVariable;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Label2.Caption:=MyGlobalVariable;
end;
```

Lazarus – Globale Daten im Projekt speichern

2. Zugriff durch alle Prozeduren und Funktionen der aktuellen Unit und aller anderen Units?

Was macht man aber, wenn man nicht nur Formular mit einer Unit in seinem Programm hat, sondern viele Units die auf diese eine Information zugreifen müssen.
In diesem Fall muss die Deklaration im Internface-Teil der Unit erfolgen:

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls;

...

implementation
```

Die Unit mit der Deklaration, wird in allen anderen Units im „Uses“-Teil verlinkt.
Damit kann es aber zu Problemen geben wenn die Units sich gegenseitig im „Uses“- Teil verlinken.

Lazarus – Globale Daten im Projekt speichern

3. Die Deklaration erfolgt in einer separaten Unit, diese Unit wird dann in allen anderen Units verlinkt.

Also wenn die Möglichkeit 1. und 2. nicht in Frage kommen, was macht man dann?
Man definiert eine Unit, in der man dann alle „globalen Variablen“ deklariert?
Eigentlich definiert man ein Objekt, dessen „Property“ die globalen Werte aufnehmen.

```
unit Configuration;
{
Beispielcode für Antwort 3
Diese Unit wurde von "m.fuchs" auf dem "2. Norddeutschen Lazarustreffen" erstellt.
Sie zeigt wie man eine Klasse erstellt und wie man die Property erzeugt.
Für jede "Globale Variable" benötigt man die jeweils entsprechenden Einträge.
}
{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils;

type

  { TConfig }

  TConfig = class(TObject)
  private
    //
    // Pro "Globale Variable" braucht man diese zwei Zeilen
    //
    FMyGlobalVariable: string;
    procedure SetMyGlobalVariable(AValue: string);
  public
    //
    // Pro "Globale Variable" braucht man diese zwei Zeilen
    //
    property MyGlobalVariable: string read FMyGlobalVariable
      write SetMyGlobalVariable;
  end;

var
  //
  Config: TConfig;

implementation

{ TConfig }
//
// Pro Globale Variable braucht man die Procedure
//
procedure TConfig.SetMyGlobalVariable(AValue: string);
begin
  if FMyGlobalVariable = AValue then Exit;
  FMyGlobalVariable := AValue;
end;
//
initialization
  Config := TConfig.Create;

finalization
  FreeAndNil(Config);
end.
```

Lazarus – Globale Daten im Projekt speichern

3.1 Die Unit aus 3.1 kann noch weiter optimiert werden.

Jetzt ist das nicht sehr Anwenderfreundlich, wenn pro „globale Variable“ 9 Zeilen Code wie in Beispiel 3. benötigt werden.

„m.fuchs“ lieferte deshalb noch eine einfachere Version, bei der man nur noch 3 Zeilen Code pro „globaler Variable“ benötigt.
Das Beispiel aus aus 3.0 wurde hier entsprechend „verkürzt“:

```
 unit Configuration;
{
Beispielcode für Antwort 3.1

Diese Unit ist eine Überarbeitung von 2.2
}
{$mode ObjFPC}{$H+}

interface

uses
  Classes, SysUtils;

type

  { TConfig }

  TConfig = class(TObject)
  private
    //
    // Pro "Globale Variable" braucht man diese Zeile
    //
    FMyGlobalVariable: string;
  public
    //
    // Pro "Globale Variable" braucht man diese zwei Zeilen
    //
    property MyGlobalVariable: string read FMyGlobalVariable
      write FMyGlobalVariable;
  end;

var
  //
  Config: TConfig;

implementation

{ TConfig }

initialization
  Config := TConfig.Create;

finalization
  FreeAndNil(Config);
end.
```