Delphiコンポーネントの基礎

- Delphiの基本のさわり
  - プログラム関連のファイルの説明
  - コンポーネント関連のファイルの説明
  - プロパティエディタ関連のファイルの説明
  - クラスのアクセス権の説明
  - クラスツリーの抜粋とファイル拡張子の説明

- ちょっと作りましたコンポーネント
  - 簡単便利なコンポーネントの実演説明（トラブルなければ）

文責　　寺口　隆

Delphiのファイルと構文 (プログラム)

.dproj

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <ProjectGuid>{155E8CB5-DC40-42BB-B9FF-98F52976F919}</ProjectGuid>
    <ProjectVersion>18.1</ProjectVersion>
    <FrameworkType>VCL</FrameworkType>
    <MainSource>Project1.dpr</MainSource>
    <Base>True</Base>
    <Config Condition="'$(Config)'==''">Debug</Config>
    <Platform Condition="'$(Platform)'==''">Win32</Platform>
    <TargetedPlatforms>1</TargetedPlatforms>
    <AppType>Application</AppType>
```

.dpr → .EXE

```
program Project1;

uses
  Vcl.Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}

begin
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

.dproj プロジェクトファイル XML によるオプション定義
.dpr プロジェクトファイル　メイン プログラムのprogram ソース モジュール
.Pas ユニットファイル　コンパイルの最小単位のunit ソースモジュール
.dfm VCL フォームのファイル .pasに対して1つだけ

.pas → .DCU

```
unit Unit1;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Program　実行プログラムの宣言
　uses　使う　　Unit名とファイルの位置
　　Begin
　　Application (TApplication)の実行
　　end;

unit　ユニットの宣言
interface　このUnitを参照するUnitからも使える
　Uses / Const/　Type /　Var /　Procedure / Fucntion
implementation このUnitのみで使える

.dfm

フォームの定義
　　コンポーネントの配置
最初のﾊﾞｰｼﾞｮﾝでは
ﾊﾞｲﾅﾘﾌｧｲﾙだったが
現在はﾃｷｽﾄﾌｧｲﾙ
日本語は16進表現

Delphiのファイルと構文 (コンポーネント)

.dproj

.dpk → .BPL, .dcp

```
package pkgNext;

{$R *.res}
{$IFDEF IMPLICITBUILDING This IFDEF should not be used by users}
{$ALIGN 8}
・・・・・・・
{$ENDIF IMPLICITBUILDING}
{$DESCRIPTION '新コンポ群'}
{$IMPLICITBUILD OFF}

requires
  rtl,
  vcl;

contains
  RegisterNextCompo in 'RegisterNextCompo.pas';

end.
```

パッケージにはコンポーネント登録用のユニットを指定する
コンポーネント登録用ユニットに使用するユニットを指定して
コンポーネントを登録する.
これの方がコンポーネントの追加削除が簡単です

.pas → .DCU

```
unit RegisterNextCompo;
Interface
uses
  Classes;
procedure Register;
implementation
uses
 ComponentContainer;
procedure Register;
begin
 RegisterComponents('デモ', [TComponentContainer, TComponentsListUp,
TStringsContainerX,TMemoryStreamContainerX]);

end;
end.
```

.pas → .DCU

```
unit ComponentContainer;
interface
uses
  SysUtils, Classes, Vcl.Controls;
type
  TccMode = (ccSaveFile, ccLoadFile, ccSaveMemory, ccLoadMemory);
type
  TcomponentEX = class(TComponent)
  private
    FTagString: string;
  published
    property TagString: string read FTagString write FTagString;
  end;
・・・・・・・
end.
```

Delphiのファイルと構文 (プロパティエディタ)

.dproj

.dpk → .BPL, .dcp

```
package PKGPropertyEditors;
{$R *.res}
{$IFDEF IMPLICITBUILDING This IFDEF should not be used by users}
{$ALIGN 8}
.............
{$ENDIF IMPLICITBUILDING}
{$DESCRIPTION 'テラコンポ専用EDITOR'}
{$DESIGNONLY}
{$IMPLICITBUILD OFF}
requires
  rtl,  pkgTeraCompo;
contains
  registerAllpropertyEditor in 'registerAllpropertyEditor.pas';

end.
```

.pas → .DCU

```
unit RegisterAllPropertyEditor;
interface
uses
  Classes,  Designintf;
procedure Register;
implementation
uses
    Controls,
  uPropEditReg;
procedure Register;
begin
 { プロパティの編集     32ビットでリリースのみ動作するような仕様です。
   設計時にのみ使用する }
  RegisterPropertyEditor(TypeInfo(TCaption), TControl, 'Caption', TStringMultProperty);
  RegisterPropertyEditor (TypeInfo(String), TControl, 'Hint', TStringMultProperty);
   RegisterPropertyEditor(TypeInfo(String), TComponent, 'TagString', TStringMultProperty);
end;
```

→ カスタムプロパティエディタの実装ユニット

独自プロパティエディタを作成すると便利です
ここでは、Tcontrolとその継承したコンポのCaption,Hint に改行を含む文字列を挿入します
独自プロパティエディタを登録するだけのユニットです

Delphiのファイルと構文 (独自のプロパティエディタの実装処理）

.pas → .DCU

```
unit uPropEditReg;
interface
uses
 Designintf,  DesignEditors,  Controls,
 uPropEdtStringMult;
type
  TStringMultProperty = class( TStringProperty)
  public
    function GetAttributes: TPropertyAttributes; override;
    procedure Edit; override;
  end;
implementation
function  TStringMultProperty .GetAttributes: TPropertyAttributes;
begin
  Result := [paDialog, paAutoUpdate];
end;
procedure TStringMultProperty .Edit;
begin
  with TfrmStringMultPropertyEdit.Create(nil) do
  begin
    Value := GetStrValue();
    if ShowModal = mrOk then
    begin
      SetStrValue(Value);
    end;
    Release;
  end;
end;
```

TStringPropertyは文字列のプロパティエディタのクラスです
これを継承して処理をオーバーライドします
複数のプロパティエディタのクラスを継承して便利に変更します

.pas → .DCU

```
unit uPropEdtStringMult;
Interface
uses
  Windows, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls;
type
   TfrmStringMultPropertyEdit = class(TForm)
    pnlBottom: TPanel;
    btnOk: TButton;
    btnCancel: TButton;
    pnlClient: TPanel;
    bvlClient: TBevel;
    memoStringMult: TMemo;
    lblLines: TLabel;
  procedure memoStringMultChange(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
    procedure cbWantTabsClick(Sender: TObject);
  private
    function GetValue: String;
    procedure SetValue(const Value: String);
    procedure UpdateLinesMessage;
  public
    property Value: String read GetValue write SetValue;
  end;
 処理部
```

Stringに改行をいれる処理部分
 TLabel のCaption ﾌﾟﾛﾊﾟﾃｨはENTERで
改行して下に伸びる(裏技的です）

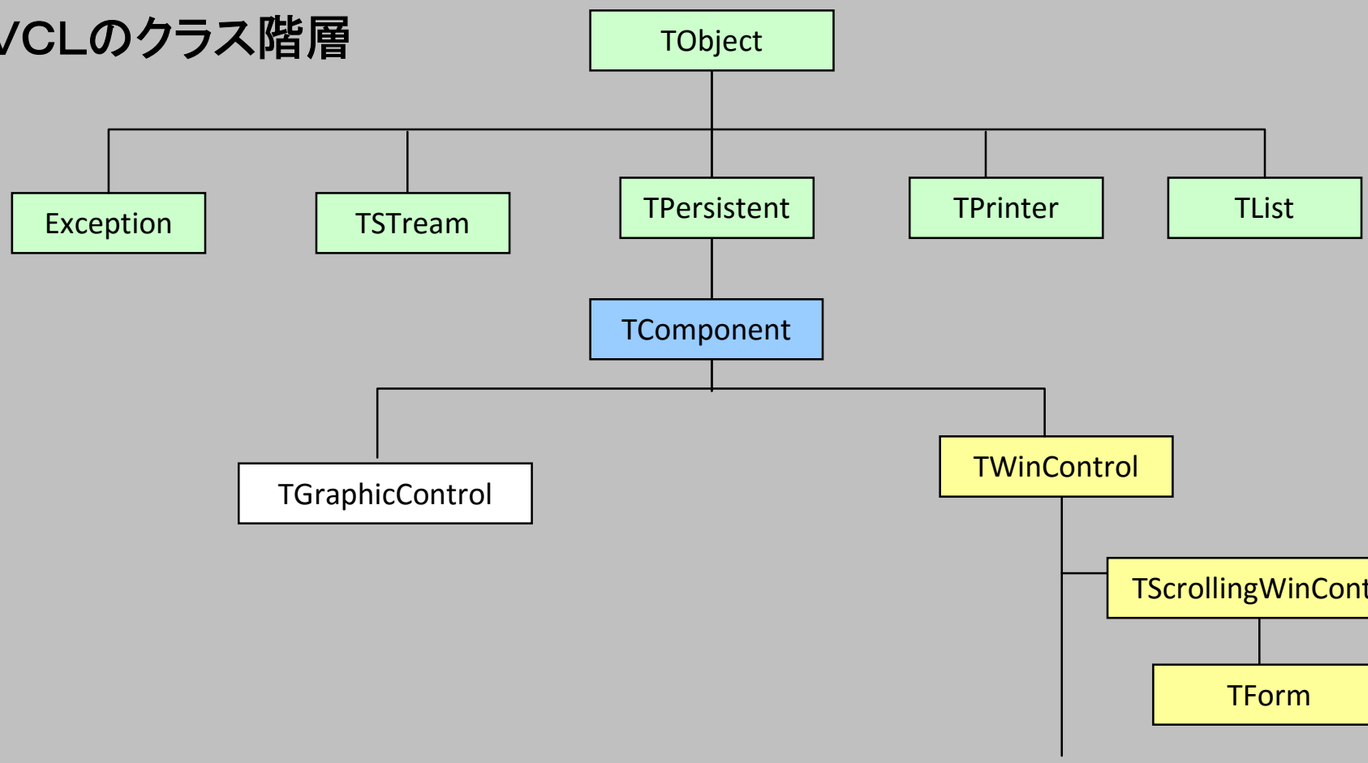
.dfm

文字入力
文字入力
文字入力

UNITファイル上のクラスのアクセス保護

```
unit ComponentContainer;

interface
type
  TStringsContainerX = class(TcomponentEX)

  private          このクラスだけがアクセスできる

    FStrings: TStrings;
    procedure SetStringsMemo(const Value: TStrings);

  protected      これを継承したクラスだけがアクセスできる


public           このUNITを参照するすべてでアクセスできる
    constructor Create(aowner: TComponent); override;
    destructor Destroy; override;

  published      このUNITを参照するすべてでアクセスできる

    property Strings: TStrings read FStrings write SetStringsMemo;       ここのプロパティはIDE上で書き込み読み出しができる

  end;
```

VCLのクラス階層



コードのみ記入

ｺﾝﾎﾟｰﾈﾝﾄﾊﾟﾚｯﾄに登録できる

ｺﾝﾎﾟｰﾈﾝﾄﾊﾟﾚｯﾄに登録できて
位置のプロパティを持つ

基本のﾌｧｲﾙ拡張子

- BPL (Borland Package Library) 設計時に Delphi IDEで使用されるファイル
  これで設計時にコンポーネントが動作できる（DLLと同じ）
- DCP (Delphi Component Package) パッケージにコンパイルしたコードのシンボル情報を含むﾌｧｲﾙ
  コンパイルしたコードはDCUにある コンパイルに必要
- DCU (Delphi Compiled Unit) PASファイルをコンパイルしたもの
- DFM (Delphi Form) フォームファイル(テキスト形式）
- DPK (Delphi Package) パッケージのプロジェクトソースコードファイル
- DPR (Delphi Project) プログラムのプロジェクトソースコードファイル
- EXE (Executable) Windows実行ファイル
- PAS (Pascal) Pascalユニットのソースコードファイル

新コンポーネントの狙いと機能

新コンポーネント名	機能	親コンポーネント	追加　ﾌﾟﾛﾊﾟﾃｨ	使いみち
TComponentEX	親ｺﾝﾎﾟｰﾈﾈﾄに 文字型のTagを追加した	Tcomponent	property TagString: string 文字列のTAG	・コンポの説明文の記述 ・プログラムで使用する 一次的な文字列の保存
TStringsContainerX	文字列リストを保存する コンテナー TMEMOはForm上しか貼れないため	TComponentEX	property Strings: Tstrings 文字列リスト	・プログラムで使用する文字列リストを保管する ・データの並び変え ・条件データの保存
TMemoryStreamContainerX	指定したコンポーネントの プロパティをメモリ上に 保存・復元するコンテナー	TComponentEX	property TargetComponent: TComponent 対象コンポーネント property ExecuteMode: TmcMode 退避・復元の指定 property Execute: Boolean 実行する	コンポーネントのプロパティの 一次退避・復元
TComponentContainer	指定したコンポーネントの プロパティをファイル上に 保存・復元するコンテナー	TComponentEX	property TargetComponent: TComponent 対象コンポーネント property ExecuteMode: TmcMode 退避・復元の指定 property Execute: Boolean 実行する	コンポーネントのプロパティの ファイル退避・復元 例）　DBの設定ﾌﾟﾛﾊﾟﾃｨ Printer設定ﾌﾟﾛﾊﾟﾃｨ
TComponentsListUp	指定したコンポーネントから 子のコンポーネントの情報を 取り出す	TComponentEX	property ownerCompo: Tcomponent 親のコンポを指定 property Execute: Boolean 実行する property ComponentsList: TStringsContainerX 実行結果の保存	使っているコンポーネントの 一覧情報の取得