【A6】C++テクニカルセッション

EMBARCADERO
TECHNOLOGIES®

DEVELOPER CAMP

C++でコンポーネントを作成しよう！

エンバカデロ・テクノロジーズ
エヴァンジェリスト　高橋智宏

アジェンダ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- コンポーネント開発とは?
- はじめてのコンポーネント開発
- コンポーネント開発の基礎
- コンポーネント開発時の注意点

Demo

コンポーネント開発とは?

コンポーネントのメリット

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- アプリケーションをビジュアルに設計・実装できる
- プロパティやイベントを設定することでコンポーネントの動作をカスタマイズできる
- 既存のコンポーネントでは実現できないと分かった場合
  - 既存のイベントやプロパティを駆使しても実現できない、または効率が悪い
- コンポーネントを利用することで開発効率がよくなる
- 共通部品を提供したい場合
  - 複数箇所で同じイベントやプロパティを繰り返すよりも、機能を実装済みのコンポーネントを開発した方が一元管理でき、バグも少なくなる
- 高度な機能をカプセル化したい場合
  - 高度な機能をコンポーネントにカプセル化することで、コンポーネントを利用する開発者は、高度な機能を簡単に利用ことが出来ます。

コンポーネントの開発手順

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- どのようなコンポーネントを開発するか仕様を決める
  - コンポーネントの動作
  - プロパティなどコンポーネントを利用するための手順
  - どのコンポーネント(クラス)を派生(または流用)するか
- C++Builder(C++言語)でコンポーネントをプログラミングする
  - もちろん、Delphi言語でも開発は可能だが…
  - コンポーネントウィザードでソースコードの雛形を作成し、コンポーネントが仕様を満たすよう、コードを実装します
- C++Builder上で、コンポーネントのテストを行う
  - C++Builderのデバッガを使って、コンポーネントをテストします
- 開発したコンポーネントをコンポーネント利用者に配布する
  - C++Builderでコンポーネントを利用できるようにするため、C++Builderにコンポーネントを登録します
  - ヘッダとライブラリのみの配布か、ソースコード一式の配布か

はじめてのコンポーネント開発

TRedLabel

Demo

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

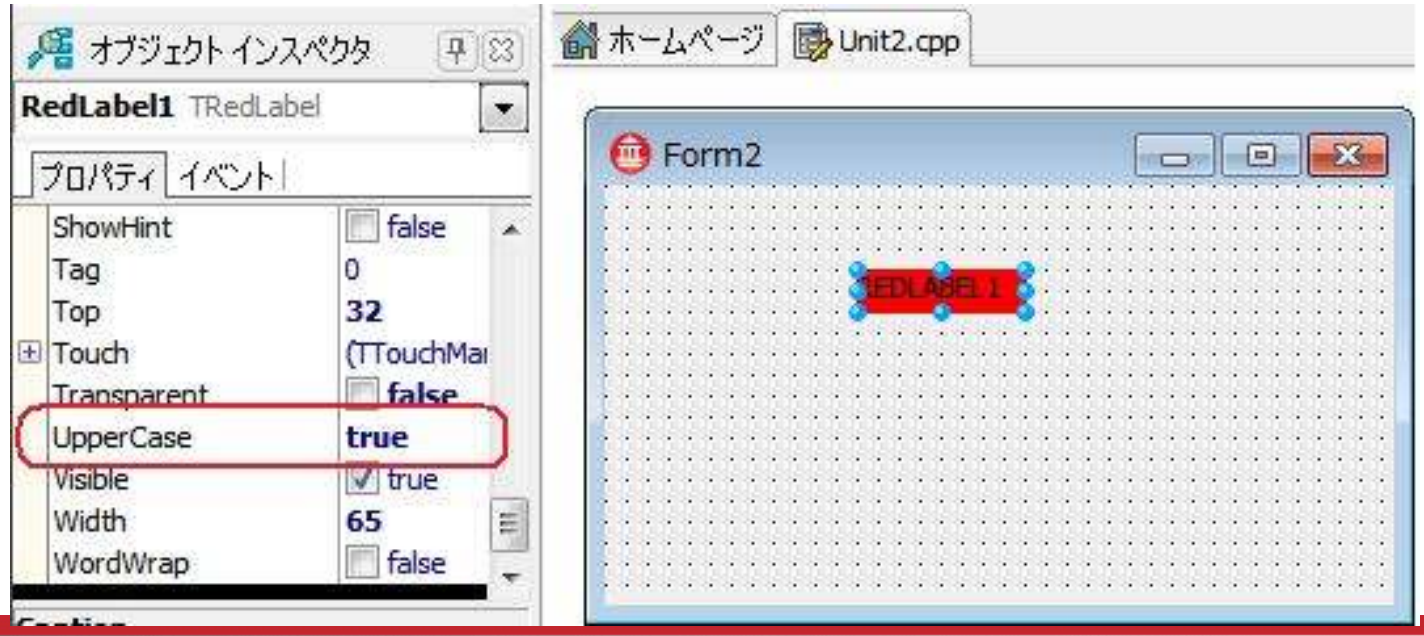
- 背景色が赤のラベル
  - TLableから派生
- パッケージ(.cprojプロジェクト)の新規作成
  - [ファイル]-[新規作成]-[パッケージ – C++Builder]
  - パッケージの説明
- コンポーネント(.h, .cpp)の新規作成
  - [コンポーネント]-[新規VCLコンポーネント]
  - TRedLabel – RedLabel.h, RedLabel.cpp
- Register関数
- Colorプロパティをcl Redに初期化
- ビルド, インストール
  - .bpl, .bpi, .lib
  - [コンポーネント]-[パッケージのインストール]で確認

プロパティの追加

Demo

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 表示する文字列を大文字にする「UpperCaseプロパティ」
  - bool型 – デフォルトはfalse
  - __published:
    - __property bool UpperCase = {read=FUpperCase, write=SetUpperCase};
- ビルド
  - オブジェクトインスペクタに「UpperCaseプロパティ」が出現

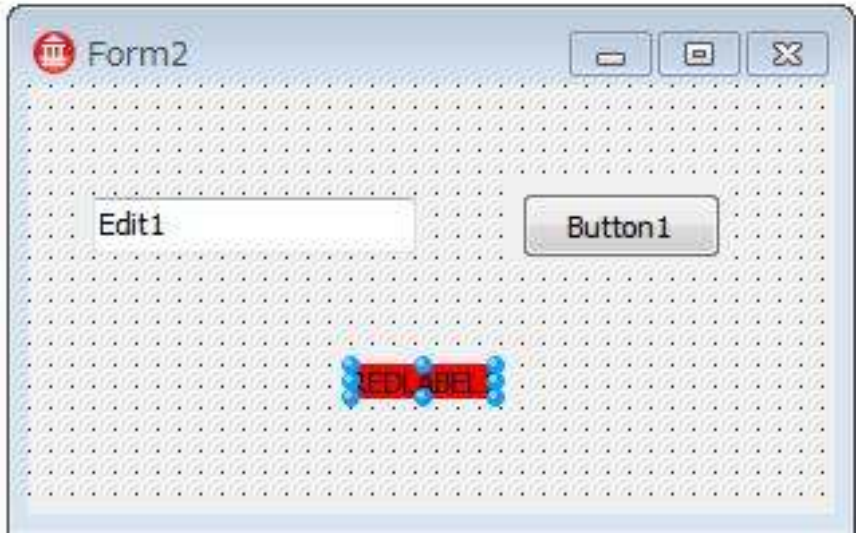


イベントの追加

Demo

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ラベル文字列が変更されたタイミングで呼び出される「OnChange」イベント
  - TNotifyEvent型
  - CM_TEXTCHANGEDメッセージをハンドリング
    - BEGIN_MESSAGE_MAP
    - VCL_MESSAGE_HANDLER(メッセージID, メッセージ構造体型, ハンドラ名)
    - END_MESSAGE_MAP(親クラス)

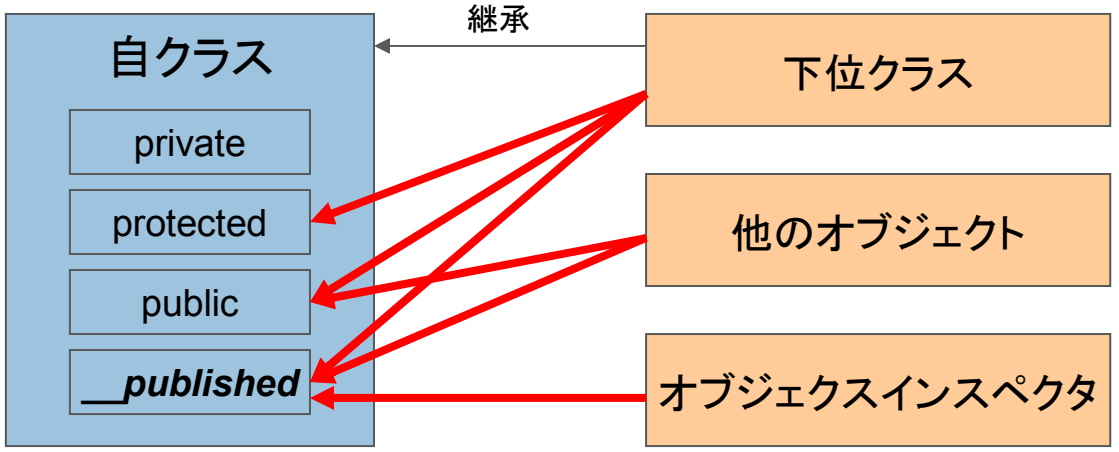


コンポーネント開発の基礎

可視性

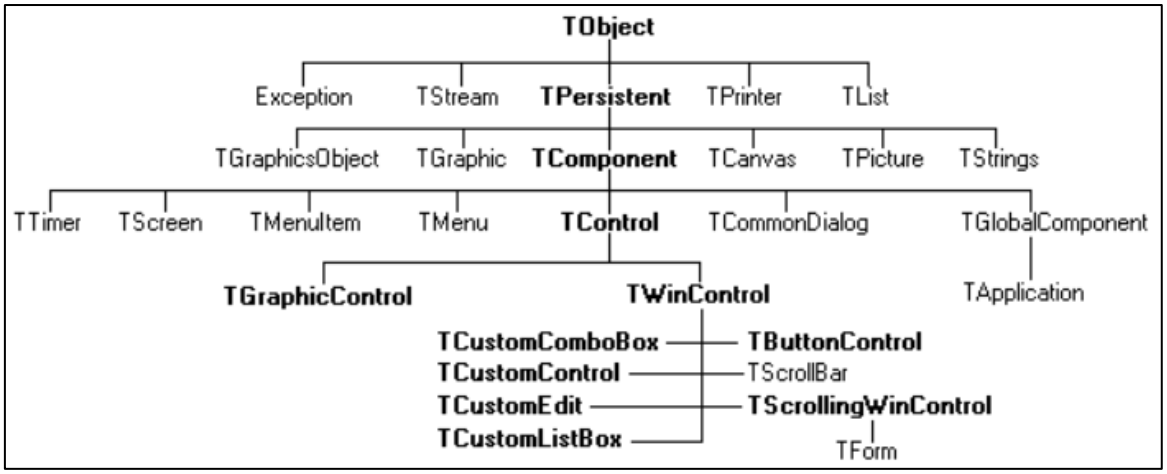
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

可視性	アクセス
private	自クラスからしかアクセスできない。フィールドやクラス内部だけで使用するメソッドはここに宣言します。
protected	自クラスまたは下位クラスからしかアクセスできない。派生関係のあるクラスでしか利用しないメンバをここに宣言します。
public	どのクラスからもアクセスできる。
__published	public同様にどのクラスからもアクセスできる。ここに宣言されたメンバーは、C++Builderでの設計時にオブジェクトインスペクタに表示することもできる。__publishedはコンポーネントのための可視性です。オブジェクトインスペクタに表示したいプロパティを宣言します。



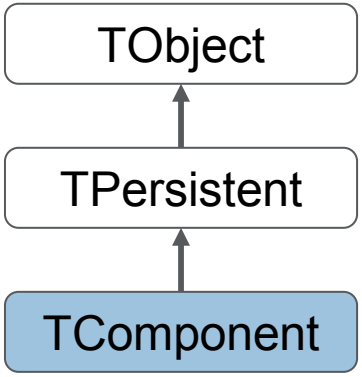
VCLクラス階層

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP



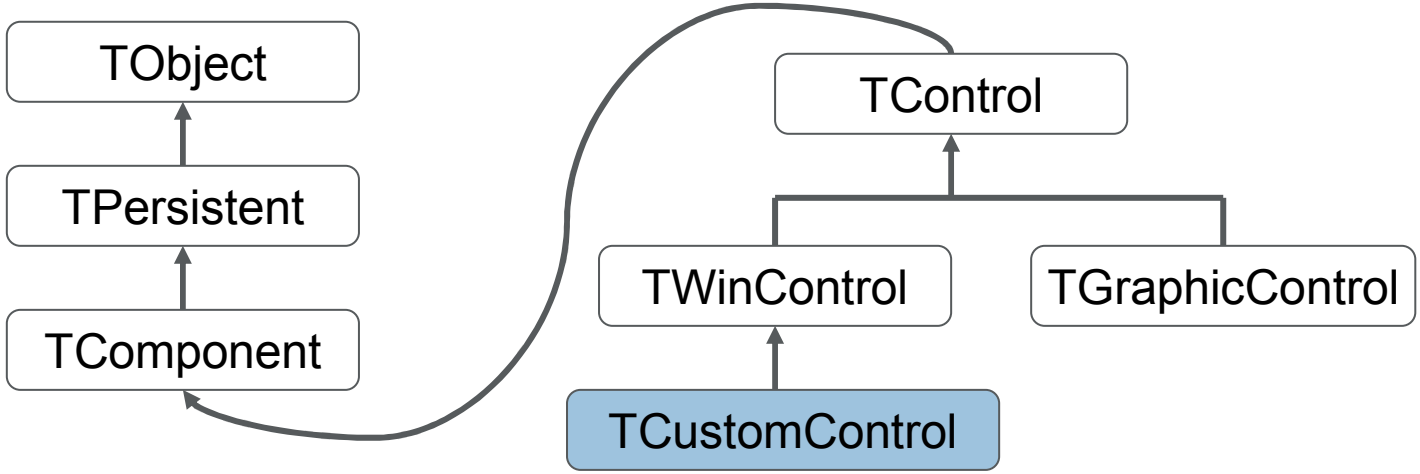
クラス名	特徴・用途
TComponent	全コンポーネントに共通する上位クラス。TComponentから派生することで、C++Builderでの設計時に利用することができます。主な下位クラス：**非ビジュアルコンポーネント**
TControl	ビジュアルコンポーネントの基本クラス。ビジュアルコンポーネントに共通するプロパティ、メソッド、およびイベントが実装されています。主な下位クラス：**一般的にこのクラスから直接派生しません。**
TGraphicControl	**ウィンドウハンドルを持たない**コントロールの基本クラス。ウィンドウハンドルを持たないため、描画とマウスへの反応のみで、フォーカス設定やキー入力ができません。 主な下位クラス：TLabel、TSpeedButton、TShape
TWinControl	**ウィンドウハンドルを持つ**コントロールの基本クラス。ウィンドウハンドルを持つため次のことができます。 ・フォーカスを受け取る ・親コントロールになれる ・キー入力を受け付けられる 主な下位クラス：TEdit、TButton、TStringGrid

所有関係



- TComponentには、「所有関係」という機能がある
  - 所有関係は、主にコンポーネントを廃棄する責任関係を表す
- TComponentクラスのコンストラクタの「TComponent* AOwner」パラメータに、オーナーコンポーネントを指定
  - 例：Button1 = new TButton(OwnerCompo);
- フォーム上に配置されたコンポーネントにも所有関係は設定される
  - デザイン時、フォーム上に配置されたコンポーネント群のオーナーコンポーネントは、フォームになる(親子関係とは別)
  - フォームを廃棄するだけで、フォーム上のコンポーネント群も廃棄することができる

TCustomControlクラス



- 独自のウィンドウコントロールを作成するための基本クラス
  - TCustomControlクラスは、TWinControlクラスから派生し、描画を行うためのCanvasプロパティが追加されている
  - 主な下位クラス：TPanel、TRadioGroup、TStringGrid
  - OSでは提供されていない独自のウィンドウコントロールを作成する場合、TCustomControlクラスから派生する

.DFMファイル

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 基本的にコンポーネントのプロパティの値は、フォームファイル(.DFM)に保存される
- 保存および読み込みの対象となるもの
  - __publishedとして宣言されているプロパティ(__property)
  - 設計時のプロパティ値が、default指令の値と異なる
    - default指令が無い場合は、0, NULL, 空文字 以外に設定されたもの
  - プロパティのstored指令がtrue(未指定時はtrueと同じ)のもの
- .DFMファイルの読み込み終了のタイミング
  - TComponentクラスの
    - virtual void __fastcall Loaded(void); をオーバーライドする
- コンポーネント自身の状態を動作時に知るには?
  - TComponentクラスのComponentStateプロパティを検査する

default指令

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- すべてのプロパティの値を保存すると、フォームファイル(.DFM)のサイズが肥大化すると同時に、フォームファイルをリンクする.exeファイルのサイズも大きくなってしまう
  - 設計時、初期値のままで変更されていないプロパティは、.DFMファイルに保存しないようにして、ファイルのサイズを減らすことが可能
  - 例：default指令でデフォルト値を指定する
    - __property int Value = { read=FValue, write=FValue, **default=100** };
- 注意点
  - default指令は、あくまでも.DFMファイルのサイズを減らすためだけに使用する
  - default指令使用したからといって、プロパティ値が初期化されるわけではない
  - プロパティ値の初期化は、コンストラクタなどで行う必要あり
    - default指令で指定したデフォルト値とプロパティの初期値は、必ず同じにする!!
  - default指令できるプロパティの型は限られてる
    - 整数型、文字列、論理型、列挙型、集合型、ポインタ
    - 実数型は ×

イベントプロパティの設計

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- イベントハンドラの型を宣言
  - イベントハンドラとやりとりする情報が無い場合
    - Classesユニットで宣言されているTNotifyEvent型を使用すればよい
      - typedef void __fastcall (__closure *TNotifyEvent)(System::TObject* Sender);
  - イベントハンドラとやりとりする情報(特別な引数・戻り値)がある場合
    - 独自にイベントハンドラ型を設計しなければならない
      - typedef 戻り値型 __fastcall (__closure *イベントハンドラ型名)(パラメータリスト);
        - » イベントハンドラ型名の命名規則:「T + イベントの名前 + Event」
        - » 第1パラメータには、イベントが発生したオブジェクトを表すTObject*型のSenderパラメータを指定する

```
typedef void __fastcall (__closure *TChangeCaptionEvent)(System::TObject* Sender, TCaption NewCaption);
...
class PACKAGE TSampleLabel : public TCustomLabel {
private:
  // イベントハンドラへのメソッドポインタを格納するためのフィールドの宣言
  TChangeCaptionEvent FOnChangeCaption;
...
__published:
  // イベントの宣言
  __property TChangeCaptionEvent OnChangeCaption = { read=FOnChangeCaption, write=FOnChangeCaption };
...
};
```

オブジェクト型のプロパティ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- クイズ

```
...
private:
  TPen* FPen;
protedted:
  void __fastcall TBoxShape::SetPen(TPen* Value) {
    ～～～  // TPenのオブジェクトをセットするには?
  }
public:
  __fastcall TBoxShape(TComponent* Owner) : TGraphicControl(Owner) {
    FPen = new TPen(); // オブジェクトを作成
  }
  virtual __fastcall ~TBoxShape() {
    delete FPen; // オブジェクトの破棄
    FPen = NULL;
  }
__published:
  __property TPen* Pen = { read=FPen, write=SetPen };
...
};
```

```
(イ)
FPen = Value;

(ロ)
delete FPen;
FPen = Value;

(ハ)
if( FPen != Value ) {
  delete FPen;
  FPen = Value;
}

(ニ)
FPen->Assign(Value);
```

コンポーネント型のプロパティ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- リンクしているコンポーネントが破棄された場合、もともとプロパティのフィールドに格納されているオブジェクト参照は、破棄されたオブジェクトを指したままになりますよね？
  - VCLには、この問題に対処するために「コンポーネントが削除されたことを通知する」機能が備わっている

```
...
private:
  TWinControl* FFocusControl;
protected:
  void __fastcall SetFocusControl(TWinControl* Value) {
    FFocusControl = Value;
  }
  void __fastcall Notification(TComponent* Acomponent, TOperation Operation) {
    TCustomLabel::Notification(Acomponent, Operation);
    // リンクしていたコンポーネントが削除される場合は NULL を代入する
    if( (Operation == opRemove) && (AComponent == FFocusControl) )
      FFocusControl = NULL;
  }
...
__published:
  // ウィンドウコントロールへの関連付けを行うためのプロパティ宣言
  __property TWinControl* FocusControl = { read=FFocusControl, write= SetFocusControl };
...
```

Q&A