【1B】Delphi/C++チュートリアル セッション

「C++Builder 5/6ユーザと初心者のための C++Builder2010入門」

株式会社日本情報システム
筑木　真志

EMBARCADERO TECHNOLOGIES

アジェンダ

- C++Builderについての簡単なおさらい
- C++Builder 2010で簡易CSVビュワーを作る
- C++Builder 5/6からC++Builder 2010への移行ポイント

C++Builderについての簡単なおさらい

C++Builder 2010の特徴

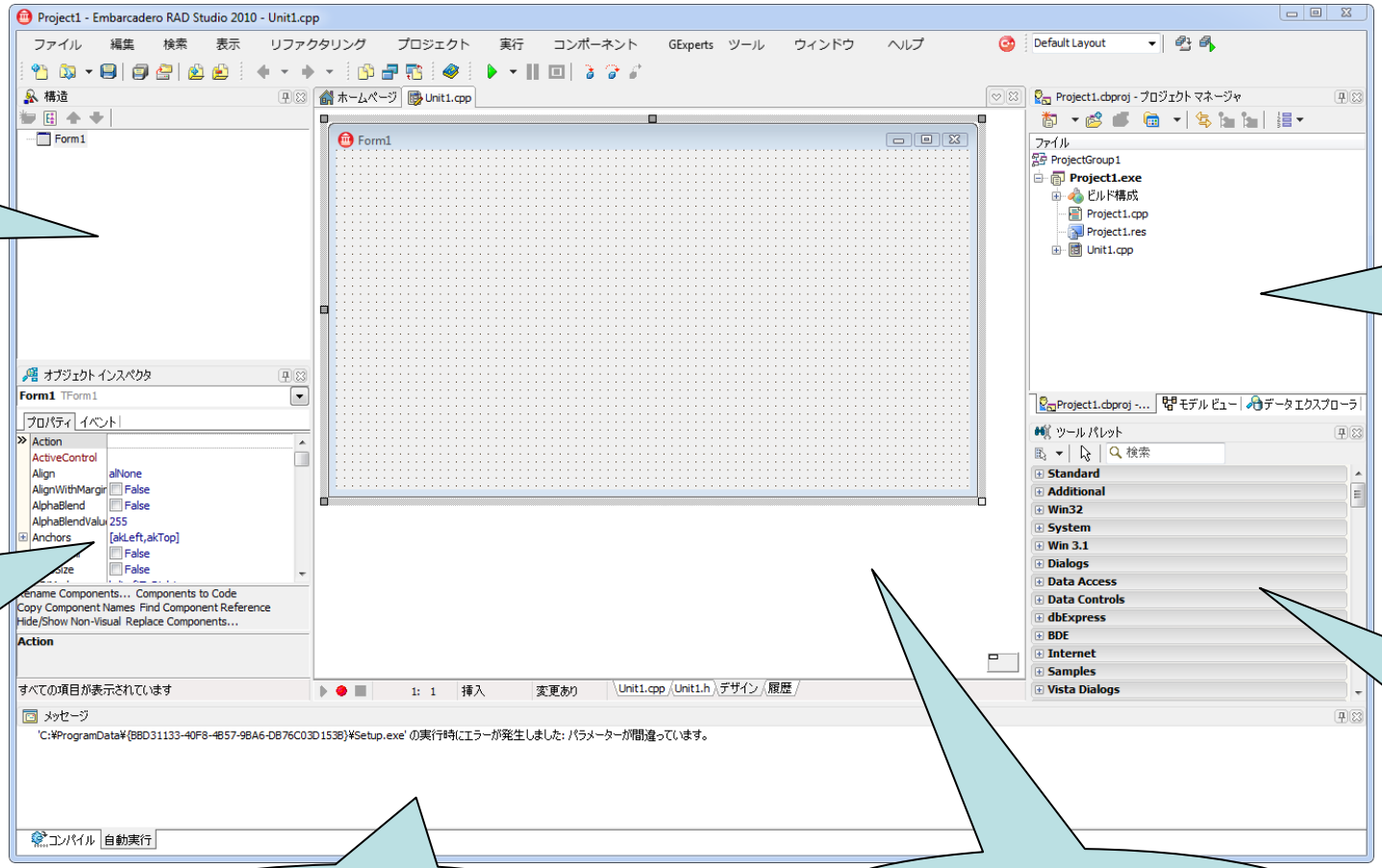
- C++Builder は、高速アプリケーション開発（RAD）用のオブジェクト指向のビジュアルなプログラミング環境
- 「コンポーネント」と呼ばれる部品をウィンドウに貼り付けてアプリケーションが作成できる。
- Unicodeの全面的採用
- 言語の記述はC++
- 膨大なC/C++のソースコード、ライブラリが使用できる
- 次期標準C++であるC++0xに部分的対応
- 次期標準C++ライブラリTechnical Report 1(TR1)に部分的対応
- Boostに部分的に対応（バージョン1.39.0が同梱）

C++Builderの欠点

- コンパイラの設計が古い
- OSSがサポートしていない場合がある
- C++0xや最新のBoostなどに対応できていない

  →新コンパイラが開発中

C++Builder 2010のIDE



構造ビュー

プロジェクトマネージャー

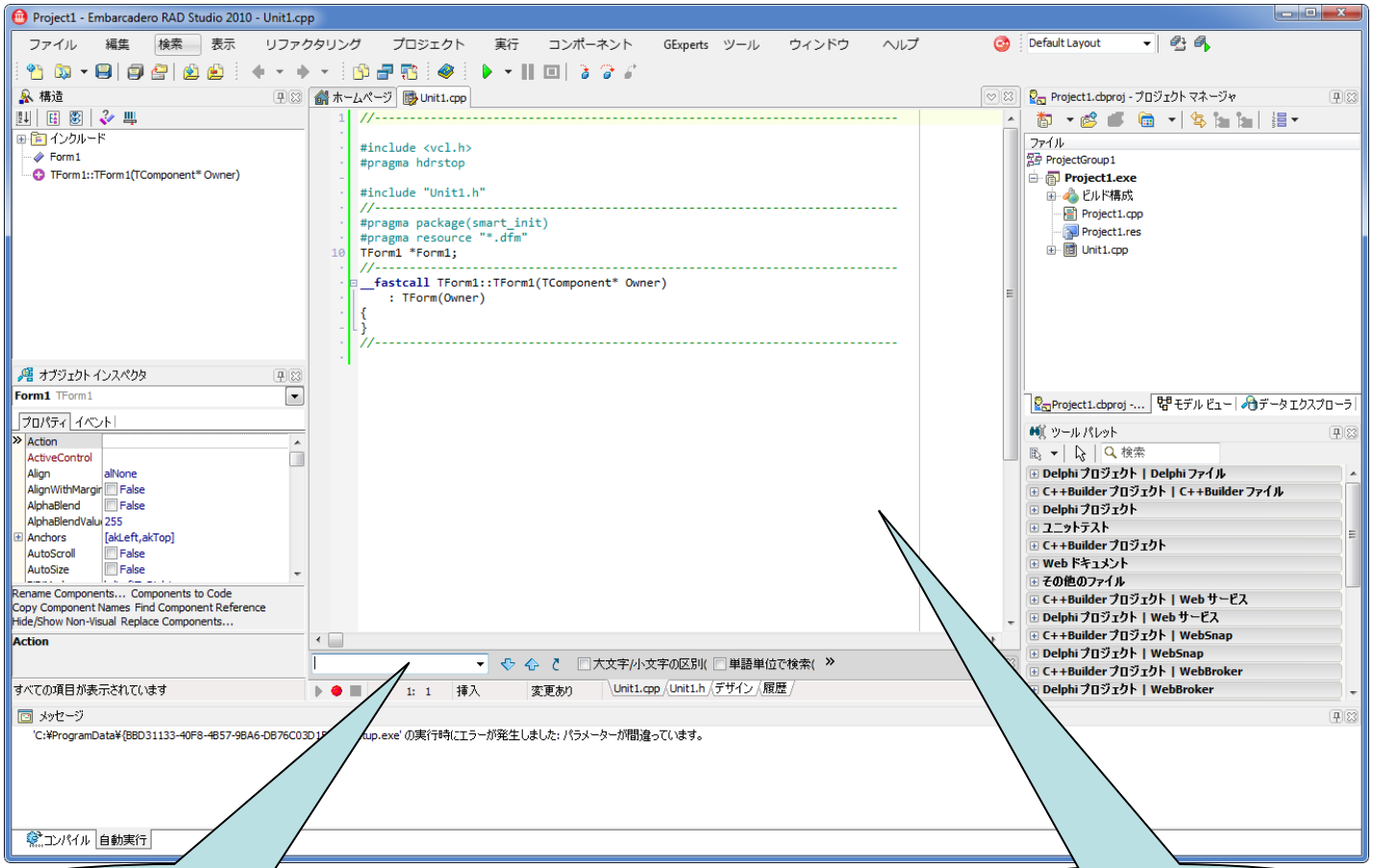
オブジェクトインスペクタ

ツールパレット

メッセージビュー

フォームデザイナ

C++Builder 2010のIDE



検索バー

コードエディタ

C++Builderでのプログラミングの流れ

- 「プロジェクト」を作成する
  - プロジェクトとは、C++Builder を使ったアプリケーション開発の中核であり、アプリケーションを構成するファイルの集まり
  - これらのファイルは設計時に作成されるものもあれば、ソースコードをコンパイルしたときに自動生成されるものもある

- アプリケーションのユーザーインターフェースを作成する
  - ツールパレットより、「フォーム」にコンポーネントを配置する
  - コンポーネントのサイズや位置などを示す「プロパティ」を変更する
    - オブジェクトインスペクタで設計時に変更
    - ソースコード内で動的に変更
  - ユーザーとアプリケーションの対話を取り持つ「イベントハンドラ」を記述する

VCL（ビジュアルコンポーネントライブラリ）

- VCLとは
  - C++Buiderの基本的なフレームワーク
  - WindowsのAPIをベースとしたクラスライブラリ
  - 中身はDelphi言語で実装

ユニットとフォーム

- 「ユニット」
  - ユニットとは、個別にコンパイルされるC++Builderコードのモジュール
  - ソースファイル(*.cpp)とヘッダファイル(*.h) の組み合わせ

- 「フォーム」
  - フォームとは、アプリケーションのうち、目に見えるウインドウ
  - C++Builderを起動すると、空のメインフォームが作成される
  - アプリーケーションを作成するには、フォームに「コンポーネント」を配置して、その振る舞いをソースコードに記述する
  - 「ユニット」にフォームの内容定義ファイル(*.dfm)が加わったもの

コンポーネント

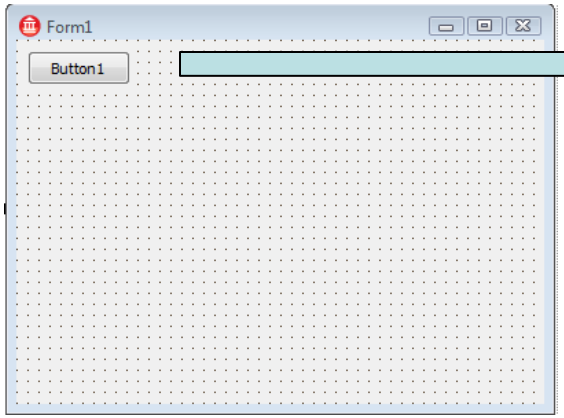
- 「コンポーネント」とはアプリケーションを構成する「部品」
- ビジュアルコンポーネント
  - ボタンやテキストボックス、グラフィックイメージなど「ユーザーの目に見える」部品
- 非ビジュアルコンポーネント
  - タイマーやデータベースとの接続など「ユーザーの目に見えない」部品

プロパティ・メソッド・イベント

- 「プロパティ」とは、コンポーネントの特性を表す
  - Nameプロパティ：コンポーネントを識別する「名前」
  - Captionプロパティ：ウィンドウやボタンのラベル
  - Visibleプロパティ：表示状態を表す
- 「メソッド」とは、オブジェクトの動作を定義する関数
  - Showメソッド：オブジェクトを表示する
  - LineToメソッド：直線を描画する
- 「イベント」とは、プログラムにより検出されるアクションまたは出来事
  - OnClickイベント：ボタンがクリックされたときに発生
  - OnMouseMoveイベント：マウスがコンポーネント上で移動したときに発生
  - OnPaintイベント：コンポーネントが再描画されたときに発生

プロパティ・メソッド・イベントの例

- フォームでボタンが押されたとき、線と楕円を描画する



Button1でOnClick
イベントが発生

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  this->Canvas->Pen->Width = 3;       // ペンの幅
  this->Canvas->MoveTo(100,0);        // 座標を移動する
  this->Canvas->LineTo(300,200);      // 線を描画する
  this->Canvas->Ellipse(100, 0, 300, 200); // 楕円を描画する
}
```

- プロパティへのアクセスやメソッドの呼び出しはアロー演算子(->)を使用する

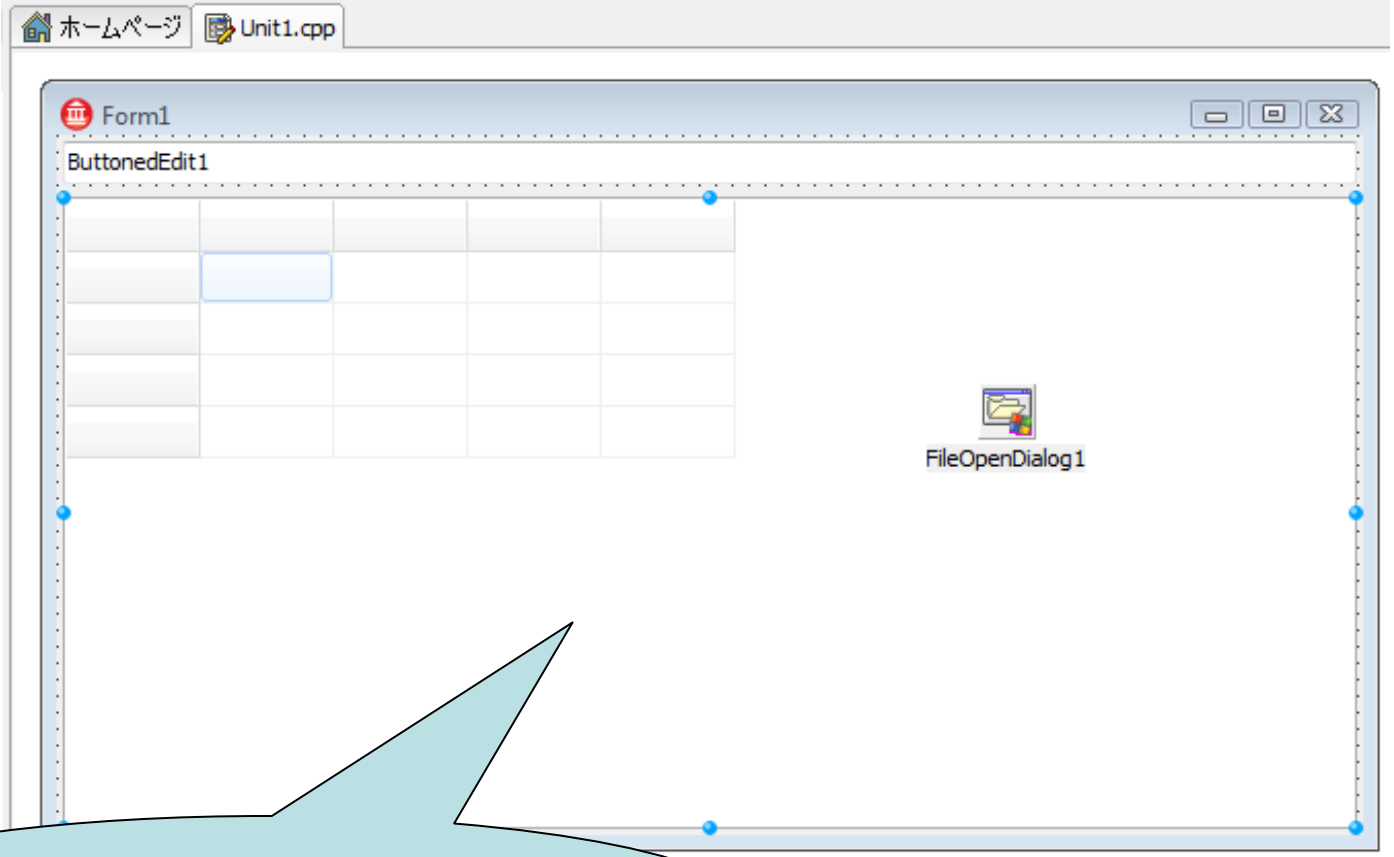
プロパティとメンバー変数の違い

- プロパティはアクセスメソッドと呼ばれる特殊なメソッドを介して変数にアクセスする

```
class aClassWithAProperty
{
private:
    int myMemberVariable;
protected:
    int __fastcall GetMemberVariable(void) {
        return myMemberVariable;
    };
    void __fastcall SetMemberVariable(int theMemberVariable) {
        myMemberVariable = theMemberVariable;
    };
public:
    __property int MemberVariable = {
        read=GetMemberVariable,write=SetMemberVariable
    }
};
```

C++Builder 2010で簡易CSVビュワーを作る

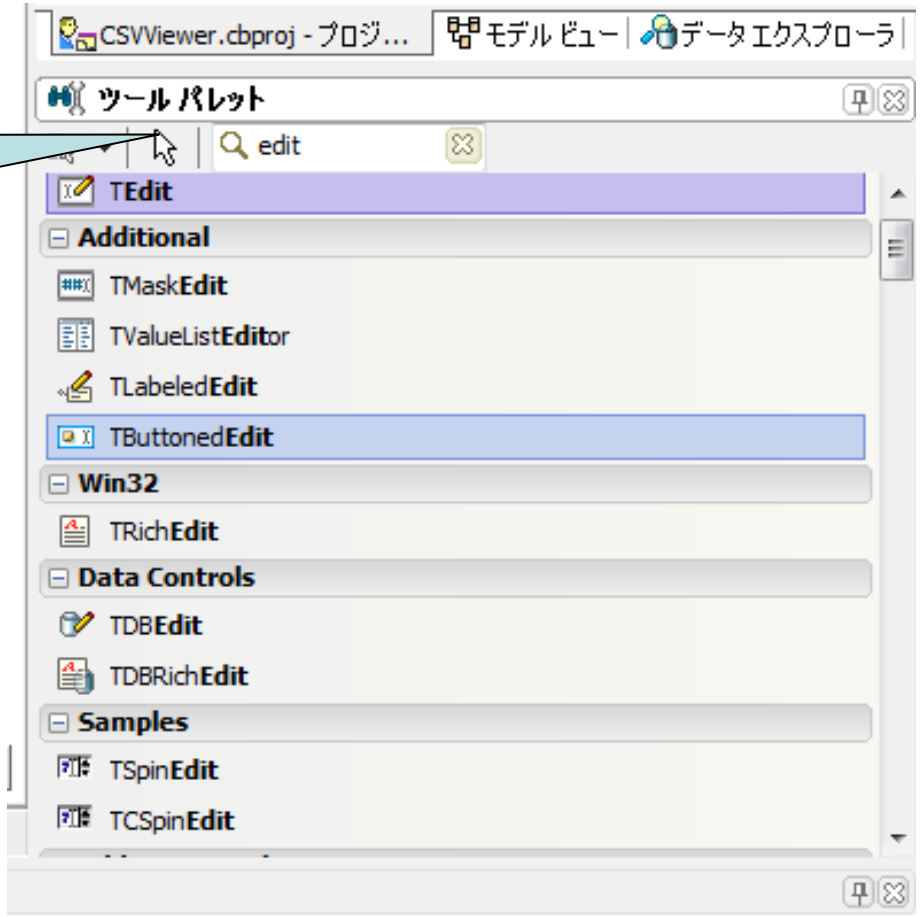
フォームにコンポーネントを配置する



コンポーネントパレットから、
ドラッグアンドドロップして配置

ツールパレットの検索

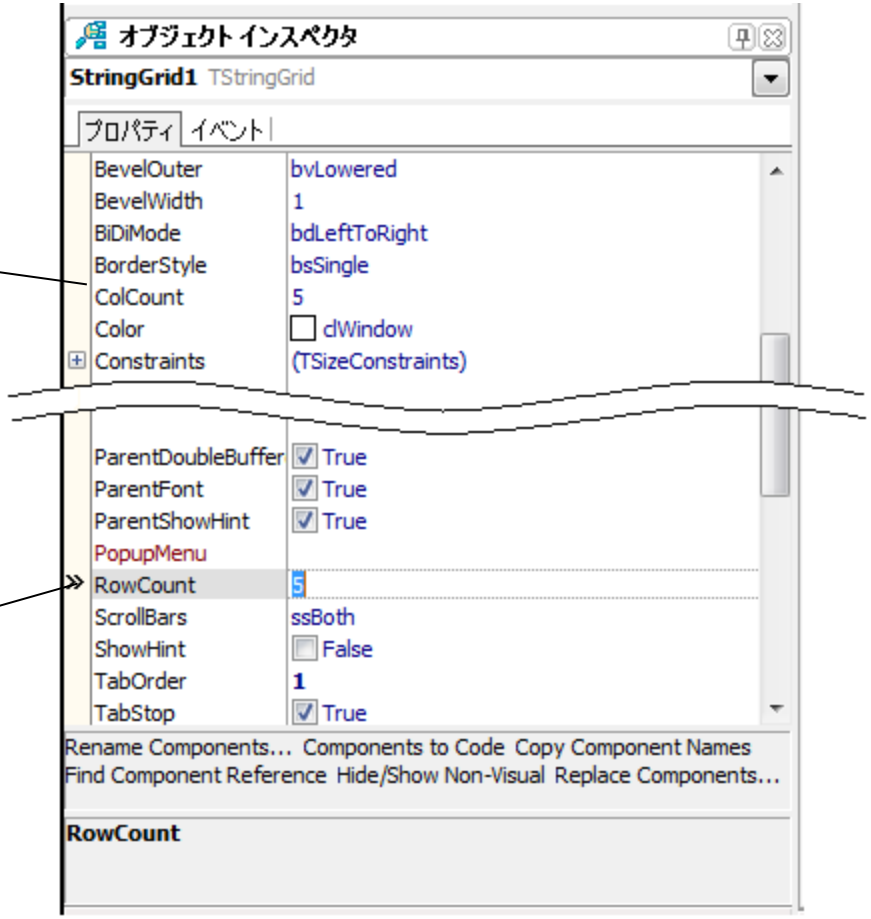
コンポーネント名の
一部を入力すると絞り込
みをしてくれる



コンポーネントのプロパティ

ColCountプロパティ：
グリッドのカラム数

RowCountプロパティ：
グリッドの行数

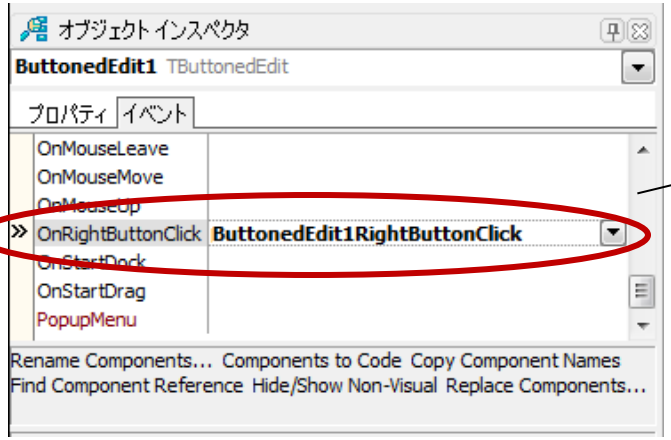


コントロール間の位置合わせ機能

- Paddingプロパティ(TPadding型)
  - コントロールの内側に配置する子コントロールとの間隔
- Marginsプロパティ(TMargins型)
  - コントロールの外側に必要な間隔のサイズ
- AlignWithMarginsプロパティ
  - AlignがalNone以外の場合、コントロールのサイズをMarginsを考慮して決定する

- これらの属性はCSSのPaddingとMarginと同様

イベントプロシージャーの記述



ButtonedEdit1を選択して、
オブジェクトインスペクタの
”OnRightButtonClick”でダブルクリックすると、イベントプロシージャーのひな形が自動生成される

```
//---------------------------------------------------------------------
void __fastcall TForm1::ButtonedEdit1RightButtonClick(TObject *Sender)
{
    if (FileOpenDialog1->Execute()) {
        ButtonedEdit1->Text = FileOpenDialog1->FileName;
        // CSVファイルのパースをする
        LoadCSV1(FileOpenDialog1->FileName);
    }
}
//---------------------------------------------------------------------
```

CSVファイルのパース

```
void __fastcall TForm1::LoadCSV1(UnicodeString CSVFile)
{
    std::unique_ptr<TStreamReader> pStreamReader(
        new TStreamReader(CSVFile, TEncoding::Default, false, 1024));

    std::unique_ptr<TStringList> pStringList(new TStringList());

    pStringList->Delimiter = L','; // カンマ区切り

    int row = 1;
    while (pStreamReader->EndOfStream == false) {
    pStringList->DelimitedText = pStreamReader->ReadLine();
        for (int i = 0; i < pStringList->Count; ++i) {
            if (i >= StringGrid1->ColCount) break;
            StringGrid1->Cells[i + 1][row] = pStringList->Strings[i];
        }
        ++row;
        if (row >= StringGrid1->RowCount) break;
    }
    pStreamReader->Close();
}
```

VCLについての「お約束」

- VCLクラスを動的に使用する場合はnew演算子で構築
  - E2459が発生

```
void foo(void)
{
  TObject o1;      // エラー
  TObject *o2 = new TObject();
}
```

  - C++Builderではスマートポインタ（自動的にdeleteしてくれるポインタ）の使用を推奨。

- VCLクラスを継承してメソッドを追加する場合（イベントハンドラを含む）は、必ず__fastcall 修飾子をつける。

CSVファイルのパース（Boost版）

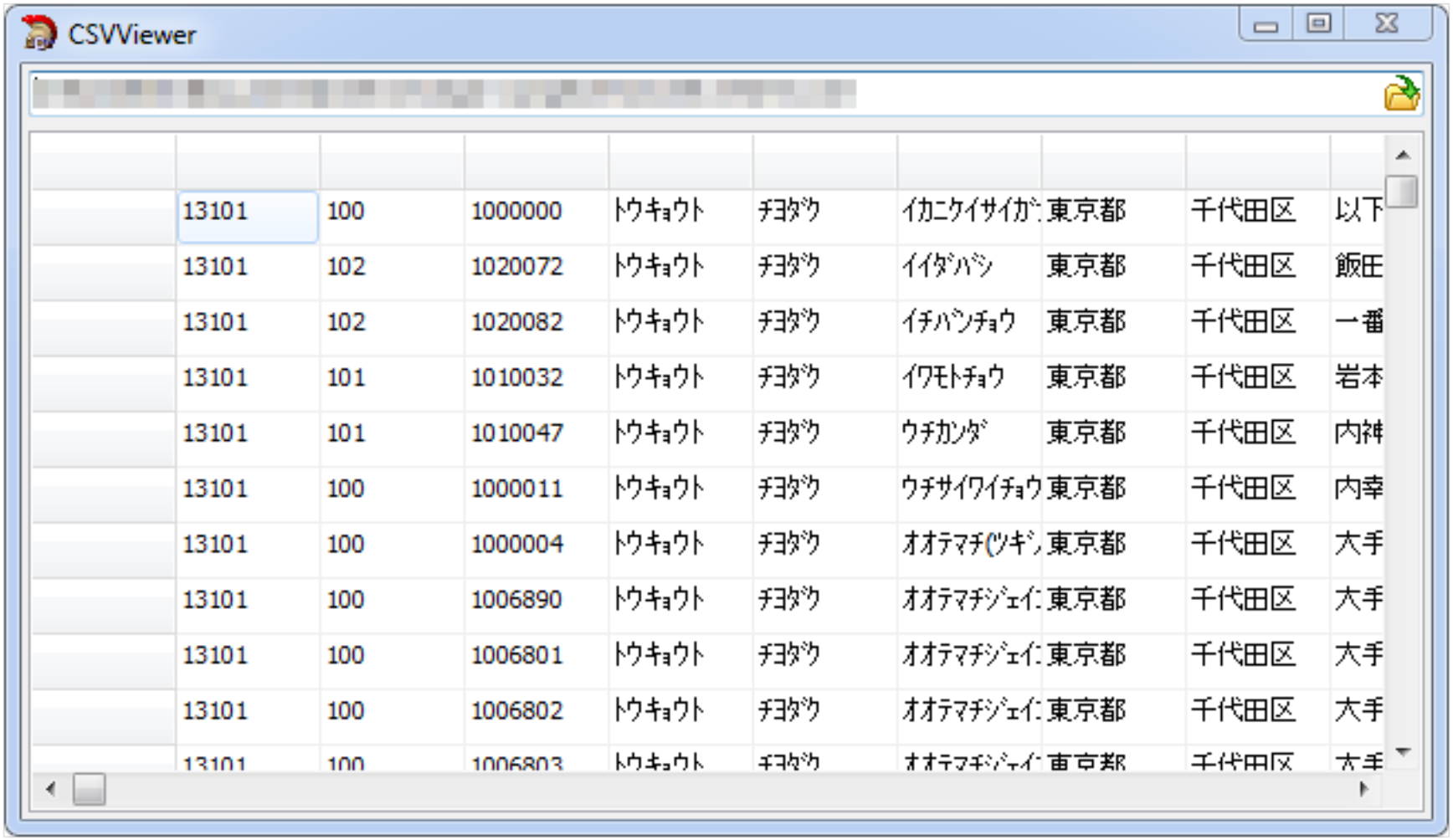
```
void __fastcall TForm1::LoadCSV2(UnicodeString CSVFile)
{
    typedef boost::char_separator<char> char_separator;
    typedef boost::tokenizer<char_separator> csv_tokenizer;
    char_separator separator(",", "", boost::keep_empty_tokens);

    std::ifstream ifs(CSVFile.t_str(), ios::in);
    std::string buffer;
    int row = 1;

    while (ifs && std::getline(ifs, buffer)) {
        int col = 1;
        csv_tokenizer tokens(buffer, separator);    // トークン分割
        for (csv_tokenizer::iterator iter = tokens.begin(); iter != tokens.end(); ++iter) {
            std::string s = *iter;
            boost::trim(s); // 空白を除去
            boost::trim_if(s, boost::is_any_of("¥""));  // 二重引用符を除去
            StringGrid1->Cells[col][row] = UnicodeString(s.c_str());

            ++col;
            if (col > StringGrid1->ColCount) break;
        }
        ++row;
    }
}
```

実行結果


CSVViewer
13101 100 1000000 トウキョウト チヨダク イカニケイサイカ 東京都 千代田区 以下
13101 102 1020072 トウキョウト チヨダク イイダバシ 東京都 千代田区 飯田
13101 102 1020082 トウキョウト チヨダク イチバンチョウ 東京都 千代田区 一番
13101 101 1010032 トウキョウト チヨダク イワモトチョウ 東京都 千代田区 岩本
13101 101 1010047 トウキョウト チヨダク ウチカンダ 東京都 千代田区 内神
13101 100 1000011 トウキョウト チヨダク ウチサイワイチョウ 東京都 千代田区 内幸
13101 100 1000004 トウキョウト チヨダク オオテマチ 東京都 千代田区 大手
13101 100 1006890 トウキョウト チヨダク オオテマチジェイ 東京都 千代田区 大手
13101 100 1006801 トウキョウト チヨダク オオテマチジェイ 東京都 千代田区 大手
13101 100 1006802 トウキョウト チヨダク オオテマチジェイ 東京都 千代田区 大手

C++Builder 2010への移行ポイント

UnicodeStringとAnsiStringの違い

- 原則として、UnicodeStringとAnsiString間の暗黙的な型変換が行われる
- ただし、状況によっては暗黙の型変換が発生せずにコンパイルエラーが発生する

AnsiStringからUnicodeStringへ

- ケース1：Ansi～系関数で”未定義の関数’～’を呼び出した”が発生
  - AnsiStrings.hppをインクルードする
  - SysUtils.hppで定義されている文字列ユーティリティ関数はUnicodeString用
  - AnsiStringを使った文字列ユーティリティ関数はAnsiStrings.hppに移動

AnsiStringからUnicodeStringへ

- ケース2：UnicodeStringを使用する、もしくは、明示的にキャストする

例1）三項演算子

```
// AsStringプロパティはUnicodeString
AnsiString Hinmoku = Code ?
    pDataset->FieldByName(“HINMOKUCODE”)->AsString : AnsiString();
```

例2）関数の引数にAnsiStringの参照がある

```
AnsiString DestPath;
// C++Builder 2010ではコンパイルエラー(E2285)が発生する
if (SelectDirectory("Select save folder.", "¥¥", DestPath,
    TSelectDirExtOpts() << sdNewUI << sdNewFolder)) {
    ShowMessage(DestPath);
}
```

AnsiStringからUnicodeStringへ

- ケース3：”const char*” / “char*” な引数に文字列を渡す
  - UnicodeString::t_strメソッドは元のデータが破壊されてしまうことを予測していないことがあるので、その場合は一度AnsiStringオブジェクトを用意して、それを渡す必要がある。

```
void Foo(char* str);

UnicodeString us;
Foo(us.t_str()); //文字列が破壊される場合がある

UnicodeString us;
AnsiString as(us);
Foo(as.c_str());
us = as;
```

TDBGridのブックマーク関連

- TBookmarkList::Itemsの型がAnsiStringからTByteDynArrayに変更
- TDataSet::GotoBookmarkの引数もvoid*からTByteDynArrayに変更

```
// C++Builder6の場合
AnsiString Mark;
Mark = DBGrid1->SelectedRows->Items[i];
pDataSet->GotoBookmark((void *)Mark.c_str());

// C++Builder2010の場合
TByteDynArray Mark;
Mark = DBGrid1->SelectedRows->Items[i];
pDataSet->GotoBookmark(Mark);
```

リンクエラーに対する対処

- "[リンカ 致命的エラー] Fatal: ファイル 'STLPMT.LIB' を開けません"
  - C++Builder 6で作成したスタティックライブラリをBDS 2006以降でリンクしたときに発生する場合がある。
  - スタティックライブラリの内部でSTLを使用している場合。
  - 対処はスタティックライブラリを再ビルドする以外方法はない。
  - STLPortがC++Builderを既にサポートしていないため。
- コンパイルオプションからNO_STRICTを外す

デモ

Q & A

最後に・・・

ご静聴ありがとうございました！！