最新Delphi/RAD Studioでこれが楽しい!
旧バージョンからの移行 "マインド“ 入門

第33回 エンバカデロ・デベロッパーキャンプ
【A2】Delphi/C++テクニカルセッション

フリーランス
富永 英明 (a.k.a. DEKO)

embarcadero
DEVELOPER CAMP

はじめに

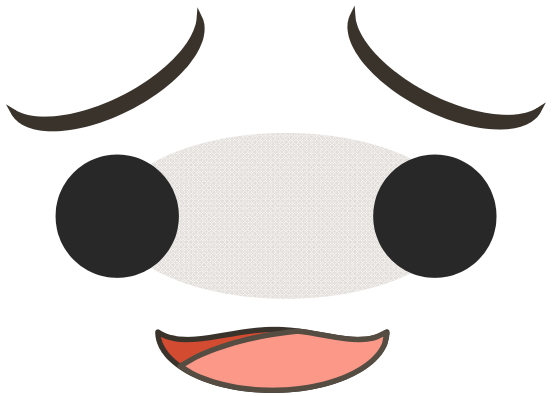
アジェンダ

- どこまでご存じなのです？
- 今時の Delphi (IDE 編)
- 今時の Delphi (言語＆ライブラリ編)
- 移行に関する雑多な話
- 他

#dcamp_jp

カル○スウォーター程度に
薄い内容かもしれませんがご了承ください

- どこまでご存じなのです？



あなたはどのバージョンのユーザー？

- 1996: Delphi 2
- 1999: Delphi 5
- 2002: Delphi 7
- 2006: Turbo Delphi
- 2010: Delphi XE
- 2013: Delphi XE5
- 2016: Delphi 10.1 Berlin

新しい機能を知るには…

- 概要は Wikipedia で。
  https://ja.wikipedia.org/wiki/Delphi
- 仕様は DocWiki で。
  http://docwiki.embarcadero.com/RADStudio/Berlin/ja/
- 動向はエンバカデロブログやニュースレターで
  https://community.embarcadero.com/blogs/blog-menu?view=tags&layout=tag&id=1894
- …でもね。

知識ばっかじゃ
頭でっかちになるで。

祝！無償版！！

- Delphi / C++Builder の Starter Edition が無償化されました！
  https://www.embarcadero.com/products/delphi/starter/promotional-download
  https://www.embarcadero.com/products/cbuilder/starter/promotional-download

トライアルと無償版の使い分け

- 最新機能の習得には無償版。
  やりたい時にすぐ使えるのはやはりメリットが大きい。

- 技術的に可能かどうかを調べるのにはトライアル。
  期限がきた後の後片付けが少々メンドイ。

- (Appmethod がそうだったように)
  トライアルの有効期限が切れたら自動的に無償版に
  なってくれると、とても便利だと思うのよエンバカさん。

- 古い製品しか持っていないのであれば、
  無償版を入れておくべき。共存可能なので！

無償版の注意点

- 10.1 Berlin Professional 以上のライセンスを所持している場合には 10.1 Berlin Starter をインストールしてはいけません。(10.1 Berlin Professional がインストールされる)

- 10.1 Berlin Professional のライセンスを所持していない場合で、10 Seattle Professional がインストールされている環境に 10.1 Berlin Starter をインストールするのは可能です。

- 複数のアカウントのライセンスを同一 PC 内で混在させるとトラブルの元になります。

- アップデートサブスクリプションに加入しているのであれば無償版は別 PC か VM で。

10.1 Berlin Starter のインストール可否

ライセンス / インストール状況	10.1 Berlin Starter Edition
10.1 Belrin Professional 以上のライセンスを所持している	VM や別 PC へ
古い Delphi をインストールしている (Professional 以上の SKU)	共存可能
古い Delphi をインストールしていない (Professional 以上の SKU)	インストール可能

今 “Delphi” で Google 検索をすると？

- 2016/12 現在、Delphi に関する情報はかなり少ない。
  何故かというと、無料HPサービスが軒並み終了したから。

  - 2009/09/30: ODN “まいぺーじ” 終了
  - 2009/12/31: 米 Yahoo! “GeoCities” 終了
  - 2010/11/01: Infoseek “Infowebライト” 終了
  - 2012/05/21: Infoseek “Infowebベーシック” 終了
    2014/06/09: Yahoo! “ジオログ” 終了
  - 2015/02/28: OCN “Page ON” 終了
  - **2016/11/10: @nifty “@homepage” 終了**

- かなり壊滅的です。

んじゃ、どこで情報を得るの？

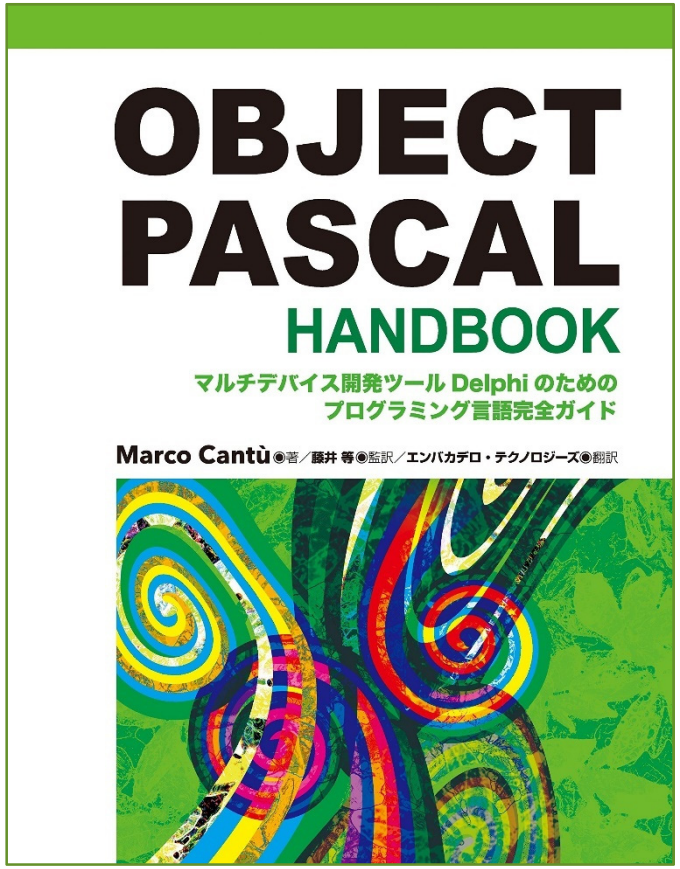
- **公式フォーラム**
  たまに落ちるのが…
  https://forums.embarcadero.com/category.jspa?categoryID=3

- **Twitter**
  #delphi_jp とか #dcamp_jp で見つかった人を
  フォローして芋づる式にフォローすればいいです。

- **Facebook**
  “Delphi” で検索すればいいです。
  海外のものも含めていくつもグループが引っかかります。

とりあえずコレを！

- OBJECT PASCAL HANDBOOK (ISBN-13: 978-4877834012)
  Delphi の “言語” に関してはとりあえずこれだけで事足りる！



無償版のお供に！

今時の Delphi (IDE 編)

embarcadero
DEVELOPER CAMP

最近の IDE の機能

- Delphi 2007 までの IDE の新機能は
  http://edn.embarcadero.com/jp/article/34361
  これ以降は以下の通り:

- クラスエクスプローラ
- 旧 IDE っぽくできる
- 言語切り替え
- ソースコードフォーマッタ
- IDE インサイト
- ビルドグループ
- 設定移行ツール
- マルチデバイスプレビュー
- カスタムマニフェスト
- GetIt パッケージマネージャ
- Castalia 拡張
- 非ビジュアルコンポーネントの表示/非表示
- IDE 使用可能メモリ増加
- Konopka 拡張
- etc...

今時の Delphi (言語＆ライブラリ編)

最近の言語＆ライブラリの機能

- Delphi 2007 までの言語の新機能は
  http://edn.embarcadero.com/jp/article/34409
  これ以降は以下の通り:

- Unicode 対応
- ジェネリクス
- 無名メソッド
- タッチ＆ジェスチャ
- RTTI 拡張
- 正規表現ライブラリ
- VCL / FMX スタイル
- ZIP ライブラリ

- FireMonkey
- ロケーション＆センサー
- 組み込み型のレコードヘルパー
- Bluetooth / BLE
- 並列プログラミングライブラリ
- Box2D
- System 名前空間のライブラリ
- etc...

System 名前空間

- System 名前空間のメソッド / クラスは基本的に、
  FireMonkey アプリ、VCL アプリ、コンソールアプリの
  いずれからでも利用可能。

- System.Win 以外は OS に依存しません。
  (但し、Windows 以外では中身が実装されていない事もあります ToT)

- 便利な機能があったりするので、確認する事をオススメします。
  http://docwiki.embarcadero.com/Libraries/Berlin/ja/System

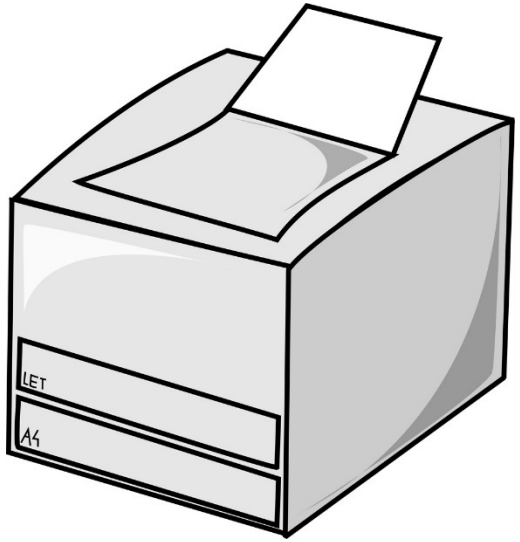
- 移行に関する雑多な情報

新しいバージョンへ移行できますか？

- ANSI 版 Delphi のプロジェクトは、
  まず **Delphi 2007** へ移行して一区切りつけるべきです。
- Delphi 2007 から Unicode 化するのなら、
  一旦 **2009** or **2010** or **XE** で動作するようにしておくといいです。
- ちょっと面白くなってくるのが **2010**、
  かなり面白くなってくるのが **XE4** です。
- Starter は途中のバージョンが貰えないので、
  Delphi 6 Personal や **Turbo Delphi Explorer** のコードを
  動かすのは大変かもしれません (10 年空いていますので...)。

帳票を移行できない！

- **QuickReport** は 10.1 Berlin 対応のものが販売されています。
- **Rave Reports** は 10.1 Berlin 対応のものが販売されています（！）。
- **Fast Report** は 10.1 Berlin に同梱されていますし、
  上位バージョンを購入する事もできます。



- …え？買いたくない？稟議下りない？
  移行できないのは帳票だけのため？

では、帳票を除いて移行しましょう。

- 単一 EXE にこだわらなければいいのです。

- 帳票部だけ古い Delphi で作ればいいのです。
- QuickReport / Rave Reports は Delphi 2007 に付属しています。
- Unicode 文字で苦労し始めたら、帳票部を FastReport に移行しましょう。移行できたら単一 EXE に戻せます (w

BDE を移行できない！

- XE7 以降、BDE は別インストールです。
- **FireDAC** に移行しましょう。
- Professional の場合には
  FireDAC アドオンパックを購入できます。

FireDAC

- …え？BDE を別インストールしてまで使うの？
  昔の BDE インストーラ持ってるの？
  デフォルトでは dBASE と Paradox のドライバしか入らないよ？

BDE は捨てろと何度も言っているのに...

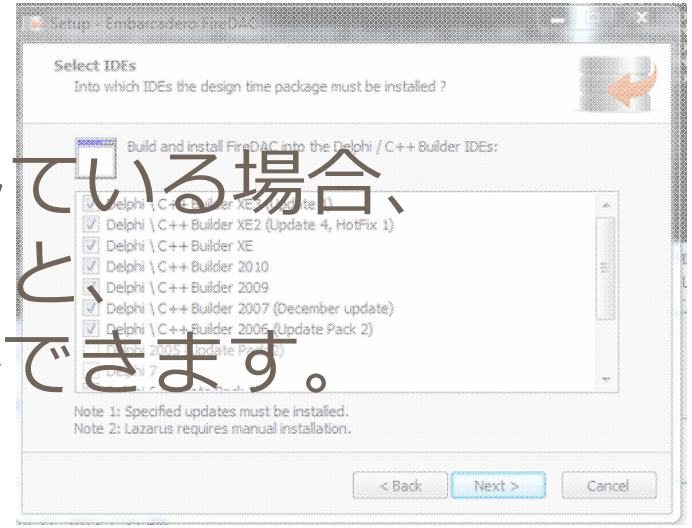
- マルチコア / マルチプロセッサの PC で
  エラーになる事があります (今どきの PC は殆どそうです)。
- Vista 以降だと実行に管理者権限が必要な事があります。
- 64bit アプリではそもそも BDE が使えません。
- 64bit OS の場合、BDE はマトモにインストールできません。
- BDE 本体も SQLLink (BDE 用 DB ドライバ) も
  既に更新されていません。動作しなくなるのは時間の問題です。

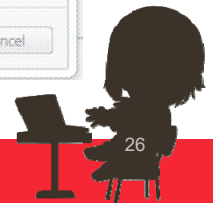
BDE 代替としては…

- それでも BDE を動かしたいのなら、もう止めはしません。

- 予算の都合で Professional の場合には、
  ADO 経由か、IBX で Firebird を使うか
  ZeosDBO (https://sourceforge.net/projects/zeoslib/)
  を使って (できれば) BDE を捨ててください。

- XE4 用の FireDAC (アドオンパック) を所持している場合、
  インストーラに **/SHOWIDE** を付けて実行すると、
  Delphi 6～XE3 にも FireDAC をインストールできます。
  (移行がとても捗ります)





IBX で Firebird を扱えるようにするための修正 (1)

IBX.IBSQL.pas

```
function TIBXSQLVAR.GetCharsetSize: Integer;
begin
  case SqlVar.SqlSubtype and $FF of
     5, // SJIS_0208
     6, // EUCJ_0208
     8, // UNICODE_BE / UCS2BE (InterBase)
    44, // KSC_5601
    56, // BIG_5
    57, // GB_2312
    64, // UNICODE_LE / UCS2LE (InterBase)
    67, // GBK (Firebird)
    68: // CP943C (Firebird)
      result := 2;
    3: // UNICODE_FSS
      begin
        // System Tables incorrectly state they are in Unicode_Fss character set but they are not
        if SqlVar.RelName.StartsWith('RDB$') or (SqlVar.SqlLen mod 3 <> 0) then
          result := 1
        else
          result := 3;
      end;
     4, // UTF8 (Firebird)
    59, // UTF8 / UTF_8 (InterBase)
    69: // GB18030 (Firebird)
      result := 4;
  else
    result := 1;
  end;
end;
```

※ 関数を置き換えます。

IBX で Firebird を扱えるようにするための修正 (2)

IBX.IBDatabase.pas

```
procedure BuildDPBConstants;
begin

  ...

  CodePages.Add('DOS775', 775);       {do not localize} // 15
  CodePages.Add('DOS858', 858);       {do not localize} // 16
  CodePages.Add('DOS862', 862);       {do not localize} // 17
  CodePages.Add('DOS864', 864);       {do not localize} // 18
  CodePages.Add('ISO8859_2', 28592);  {do not localize} // 22
  CodePages.Add('ISO8859_8', 1255);   {do not localize} // 38
  CodePages.Add('DOS866', 866);       {do not localize} // 48
  CodePages.Add('DOS869', 1255);      {do not localize} // 49
  CodePages.Add('BIG_5', 950);        {do not localize} // 56
  CodePages.Add('WIN1255', 1255);     {do not localize} // 58
  CodePages.Add('WIN1256', 1256);     {do not localize} // 59
  CodePages.Add('WIN1257', 1257);     {do not localize} // 60
  CodePages.Add('KOI8R', 20866);      {do not localize} // 63
  CodePages.Add('WIN1258', 1258);     {do not localize} // 65
  CodePages.Add('GBK', 936);          {do not localize} // 67
  CodePages.Add('TIS620', 874);       {do not localize} // 66
  CodePages.Add('CP943C', 932);       {do not localize} // 68
  CodePages.Add('GB18030', 54936);    {do not localize} // 69
end;
```

※ 手続きの最後に追記します。

IBX で Firebird を扱えるようにするための修正 (3)

IBX.IBExtract.pas

```
function TIBExtract.ExtractDDL(Flag: Boolean; TableName: String) : Boolean;
begin

  ...

  if TableName <> '' then
  begin
    if not ExtractListTable(TableName, '', true) then
      Result := false;
  end
  else
  begin
    ListCreateDb;
//    ListEUAUsers;
//    ListEncryptions;
//    if ConnectAsOwner then
//      BuildConnectString;
    ListFilters;
    ListFunctions;
    ListDomains;
    ListAllTables(flag);
    ListIndex;

  ...

end;
```

※ ４行コメントアウトします。

ZeosDBO を 10.1 Berlin へインストールできるようにする

Zeos.inc

```
// Compilation directives for Delphi XE8
{$IFDEF VER290}
  {$DEFINE VER200BELOW} // Used in code
  {$DEFINE DELPHI12_UP} // Used in code
  {$DEFINE DELPHI14_UP} // used in tests only
  {$DEFINE DELPHI15_UP} // Used in zeos.inc only
  {$DEFINE DELPHI16_UP} // Used in code
  {$DEFINE DELPHI17_UP} // Used in zeos.inc only
  {$DEFINE DELPHI18_UP} // Used in zeos.inc only
  {$DEFINE DELPHI19_UP} // Used in zeos.inc only
  {$DEFINE DELPHI20_UP} // Used in zeos.inc only
  {$DEFINE DELPHI21_UP} // Used in zeos.inc only
  {$DEFINE DELPHI22_UP} // Used in zeos.inc only
  {$DEFINE BDS4_UP} // Used in code
  {$DEFINE BDS5_UP} // Used in code
{$ENDIF}

// Compilation directives for Delphi 10 Seatle
{$IFDEF VER300}
  {$DEFINE VER200BELOW} // Used in code
  {$DEFINE DELPHI12_UP} // Used in code
  {$DEFINE DELPHI14_UP} // used in tests only
  {$DEFINE DELPHI15_UP} // Used in zeos.inc only
  {$DEFINE DELPHI16_UP} // Used in code
  {$DEFINE DELPHI17_UP} // Used in zeos.inc only
  {$DEFINE DELPHI18_UP} // Used in zeos.inc only
  {$DEFINE DELPHI19_UP} // Used in zeos.inc only
  {$DEFINE DELPHI20_UP} // Used in zeos.inc only
  {$DEFINE DELPHI21_UP} // Used in zeos.inc only
  {$DEFINE DELPHI22_UP} // Used in zeos.inc only
  {$DEFINE DELPHI23_UP} // Used in zeos.inc only
  {$DEFINE BDS4_UP} // Used in code
  {$DEFINE BDS5_UP} // Used in code
{$ENDIF}
```

```
// Compilation directives for Delphi 10.1 Berlin
 {$IFDEF VER310}
  {$DEFINE VER200BELOW} // Used in code
  {$DEFINE DELPHI12_UP} // Used in code
  {$DEFINE DELPHI14_UP} // used in tests only
  {$DEFINE DELPHI15_UP} // Used in zeos.inc only
  {$DEFINE DELPHI16_UP} // Used in code
  {$DEFINE DELPHI17_UP} // Used in zeos.inc only
  {$DEFINE DELPHI18_UP} // Used in zeos.inc only
  {$DEFINE DELPHI19_UP} // Used in zeos.inc only
  {$DEFINE DELPHI20_UP} // Used in zeos.inc only
  {$DEFINE DELPHI21_UP} // Used in zeos.inc only
  {$DEFINE DELPHI22_UP} // Used in zeos.inc only
  {$DEFINE DELPHI23_UP} // Used in zeos.inc only
  {$DEFINE DELPHI24_UP} // Used in zeos.inc only
  {$DEFINE BDS4_UP} // Used in code
  {$DEFINE BDS5_UP} // Used in code
{$ENDIF}
// END of per Delphi version defines
```

※ 追記します。

ものすごく小規模なスタンドアロン DB アプリなら…

- Professional でも FireDAC + IBLite が使えますよ。
  http://support.embarcadero.com/jp/article/43726

- Windows / macOS / iOS / Android で使える組み込み DB。
- セッション数 1
- DB サイズ 100MB 以下
- 配布に制限はない (無制限配布可能)

- DB サイズの制限がキツいのなら、
  有償の Interbase ToGo か、
  Firebird (Embedded) を使いましょう。

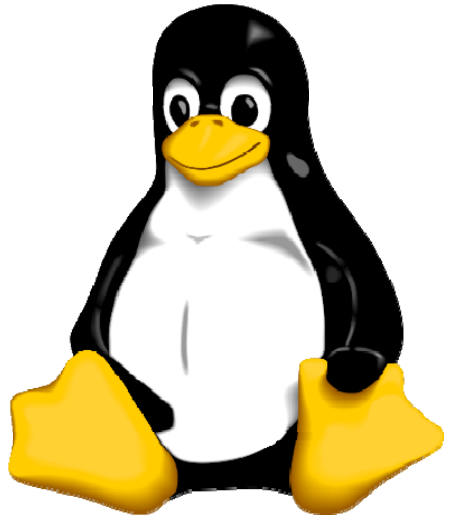
移行と同時に３層アプリにしたいけど...

- DataSnap + FireDAC でやれればそれが一番いいかと。
(DataSnap は Enterprise 以上の SKU の機能です)

- ３層の目的が外部からのアクセスかつ、
外部アクセスの頻度が少ないのなら、**WebBroker** で
外部アクセス用のアプリをポン付けするのも手かと。

- この後のセッションで詳しくやると思いますが、
Delphi アプリが Linux で動作するようになる上に、
WebBroker に対応するようです。

- http://docwiki.embarcadero.com/RADStudio/ja/WebBroker_アプリケーションの作成



コンポーネントを移行できない！

- 購入できるものは購入。
  ソースコード付きで $100 以下なら安いものです。
  (GetIt で買えればいいのですけどねぇ…)
- 更新が止まっている＆ソースコードがない →
  捨てましょう。代替コンポーネントを探すべきです。
  (たまにエンバカさんがくれるコンポーネントに依存してはいけません)
- 更新が止まっている＆ソースコードがある→
  誰かに新しいバージョン用を作ってもらいましょう。
- (たとえそれがフリーウェアだったとしても)
  仕事として依頼すればコンポーネント改修を
  請け負ってくれるヒトはいると思います。

- 楽しく移行

ある程度最近の Delphi に移行した方がいい理由

- 言語の新機能によってコードが楽に書ける場合がある。
(ジェネリクスとか無名メソッドとか)
- 新しいライブラリによって、コードが楽に書ける場合がある。
(正規表現とか ZIP ライブラリとか)
- 今まで困難だった処理を書ける事がある。
(拡張 RTTI とか)
- FireMonkey のライブラリの幾つかが System 名前空間に
移動されている。(VCL からも使える)

お小言はここまで。

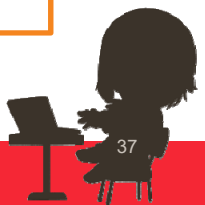
- Delphi の新しい機能を使って “楽しく簡潔に” コードを書いてみましょう。
- **楽するための努力をしてみましょう (^q^)**
- 移行だけの話ではなく、ともすれば退屈になりがちな業務アプリのメンテが退屈ではなくなるかもしれません。

高度な文字列検索と抽出

GetComponentNames.dpr

```
program GetComponentNames;

uses
  Classes, ClipBrd, RegularExpressions;

const
  Exp = 'object +(?<Component>.+):';
var
  RegEx: TRegEx;
  Match: TMatch;
  SL: TStringList;
begin
  RegEx := TRegEx.Create(Exp);
  Match := RegEx.Match(Clipboard.AsText);
  SL := TStringList.Create;
  try
    while Match.Success do
      begin
        SL.Add(Match.Groups.Item['Component'].Value);
        Match := Match.NextMatch;
      end;
    Clipboard.AsText := SL.Text;
  finally
    SL.Free;
  end;
end.
```

正規表現ライブラリ



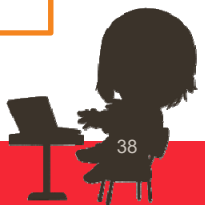
Swift のような for 文 (Stride)

uStride.pas

```
uses
 …, uStride;


var
  i: Integer;
begin
  // 0..10
  for i in Stride(0, 10) do
    Writeln(i);

  // -10..10
  for i in Stride(-10, 10) do
    Writeln(i);

  // 0..100 Step 3
  for i in Stride(0, 100, 3) do
    Writeln(i);

  // 10..0 (downto)
  for i in Stride(10, 0, -1) do
    Writeln(i);
end;
```

for in do

高度なレコード型

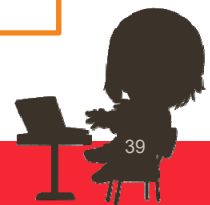


プリミティブ型のヘルパー

```
var
  Value: Integer;
begin
  Value := 100;
  Edit1.Text := Value.ToString;
end;
```

標準のヘルパーが
使いにくければ
自分でヘルパー記述して
機能を上書きしましょう

プリミティブ型ヘルパー

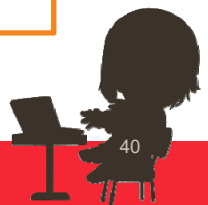


型変換のコードを減らす

uCustomeditHelper.pas

```
type
 { TCustomEditExtention }
 TCustomEditExtention = class helper for TCustomEdit
 private
    ...
 published
  // Properties
  property AsInteger: Integer read GetInteger write SetInteger;
end;


procedure TForm1.Button1Click(Sender: TObject);
var
  Value: Integer;
begin
  Value := 100;
  Edit1.AsInteger := Value;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  Value: Integer;
begin
  Value := Edit1.AsInteger;
  ShowMessage(Value.ToString);
end;
```

クラスヘルパー



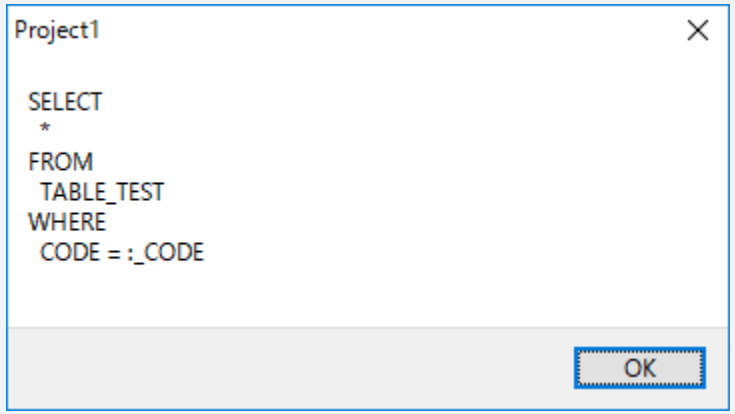
SQL をベタ書きしたくない

- 何故ベタ書きしたくないのか。
- **実行時でないとバグが発見できないから。**
  (SQL は単なる文字列なので)
- LINQ は Delphi では使えない (Oxygene にはあります)。
- そもそも LINQ (to SQL 等) では複雑な SQL に対応できない。
- でも、そこそこ簡単にやりたい。

SQL をベタ書きしたくない (1)

uSQLBuilder.pas

```
uses
  ..., uSQLBuilder;

procedure TForm1.Button1Click(Sender: TObject);
var
  Dmy: string;
  SQL: TChainSelectSQL;
begin
  Dmy := SQL.SELECT('*').FROM('TABLE_TEST').WHERE('CODE = :_CODE').Term();
  ShowMessage(Dmy);
end;
```



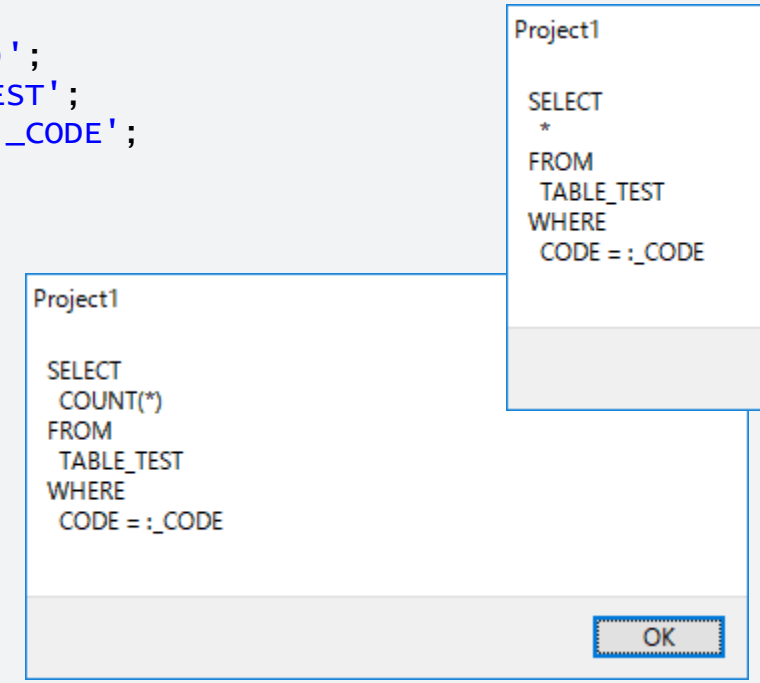
メソッドチェーン

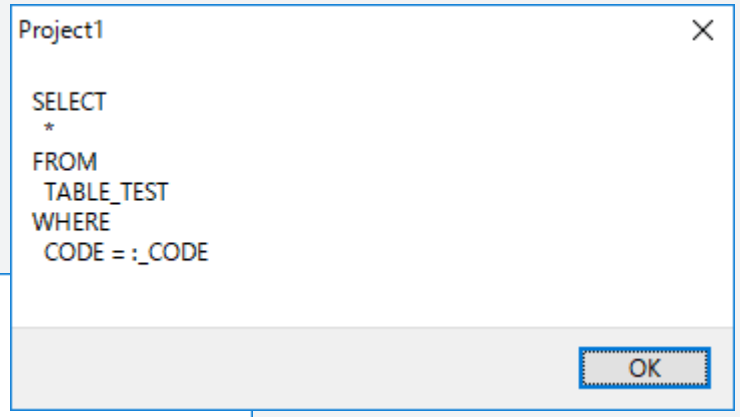
高度なレコード型

SQL をベタ書きしたくない (2)

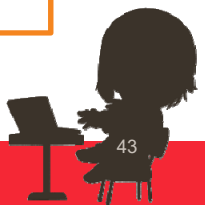
uSQLBuilder.pas

```
uses
  ..., uSQLBuilder;

procedure TForm1.Button1Click(Sender: TObject);
var
  Dmy: string;
  SQL: TSelectSQL;
begin
  // 1st
  SQL.SELECT := 'COUNT(*)';
  SQL.FROM   := 'TABLE_TEST';
  SQL.WHERE  := 'CODE = :_CODE';
  Dmy := SQL.Build;
  ShowMessage(Dmy);

  // 2nd
  SQL.SELECT := '*';
  Dmy := SQL.Build;
  ShowMessage(Dmy);
end;
```





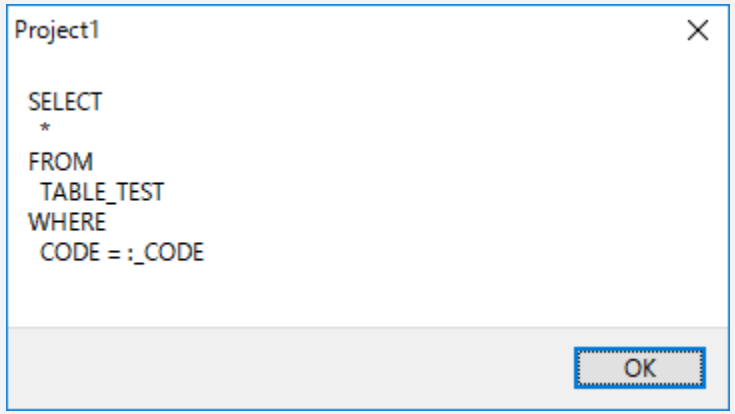
高度なレコード型



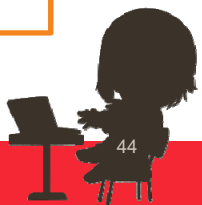
SQL をベタ書きしたくない (3)

uSQLBuilder.pas

```
uses
  ..., uSQLBuilder;

procedure TForm1.Button1Click(Sender: TObject);
const
  SQL: TSelectSQL = (
    SELECT: '*';
    FROM:   'TABLE_TEST';
    WHERE:  'CODE = :_CODE';
  );
begin
  ShowMessage(SQL.Build);
end;
```



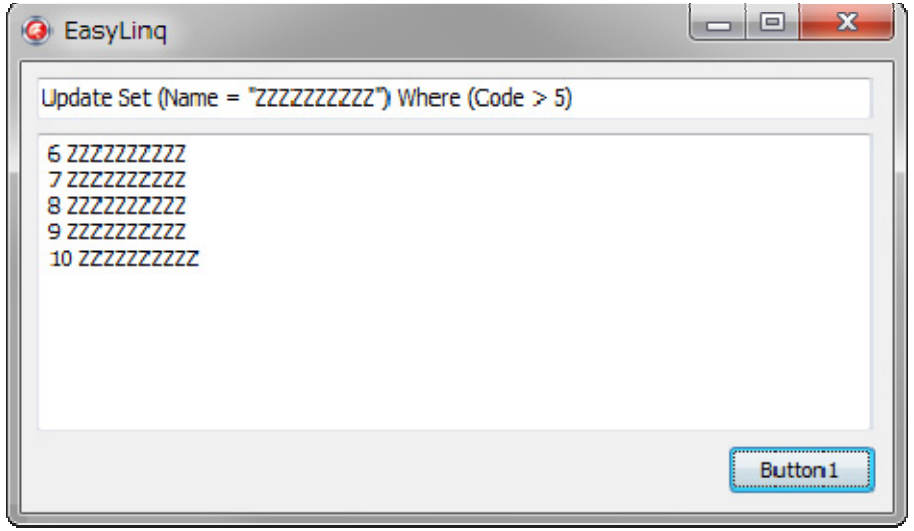
高度なレコード型



ちなみに

- LINQ (to Object) っぽい実装は Delphi にもあります。

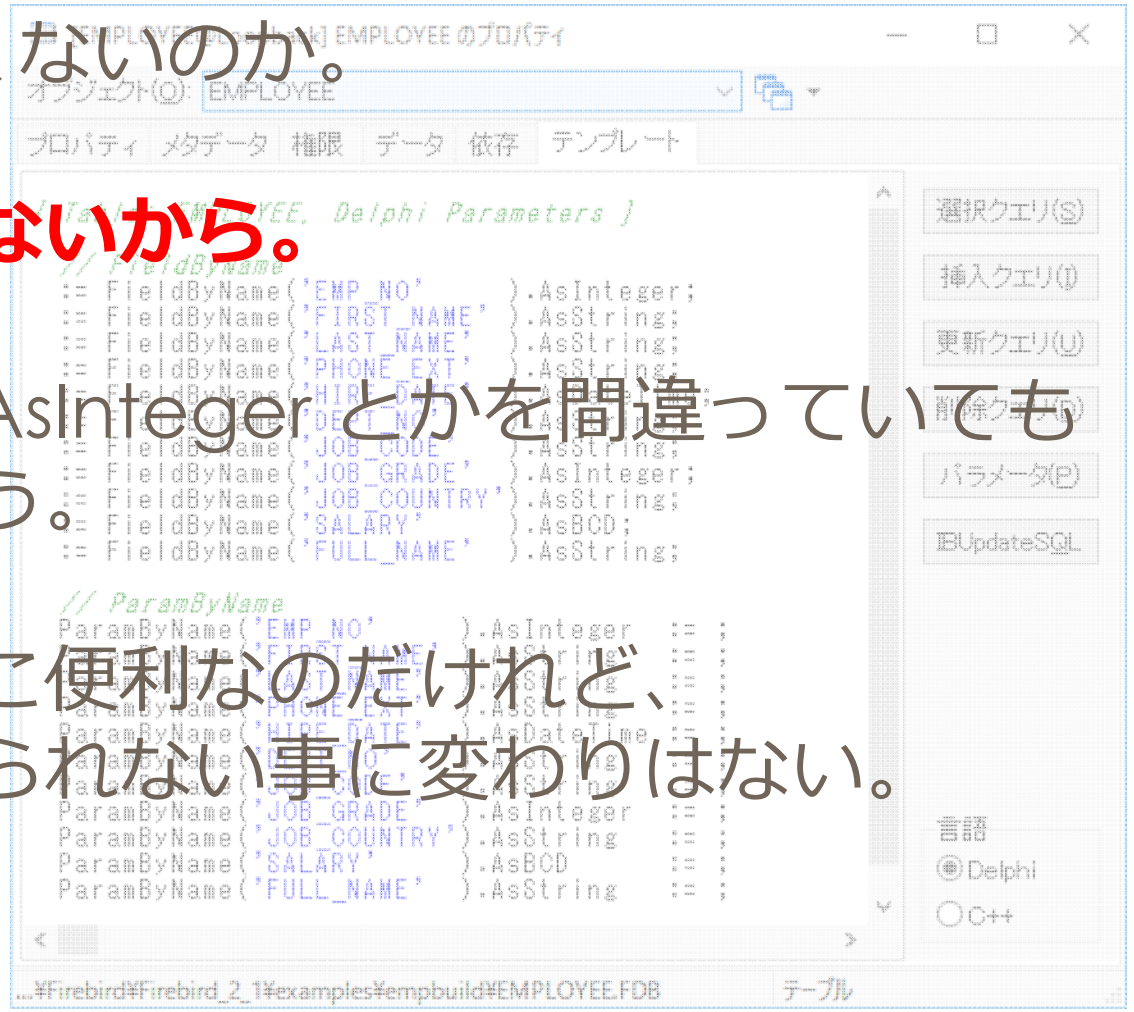
- EasyLINQ for Delphi
  https://github.com/DanielaSe/Delphi-EasyLINQ



拡張 RTTI

FieldByName() を書きたくない

- 何故 FieldByName() を書きたくないのか。
- **実行時でないとバグが発見できないから。**
- フィールド名は文字列なので、AsIntegerとかを間違っていてもやはりコンパイルが通ってしまう。
- コードジェネレータでもそこそこ便利なのだけれど、実行時でないとエラーを見つけられない事に変わりはない。



FieldByName() を書きたくない (1)

```
Edit1.Text := IntToStr(FieldByName('EMP_NO'     ).AsInteger);
Edit2.Text := FieldByName('FIRST_NAME' ).AsString;
Edit3.Text := FieldByName('LAST_NAME'  ).AsString;
Edit4.Text := FieldByName('PHONE_EXT'  ).AsString;
Edit5.Text := DateToStr(FieldByName('HIRE_DATE'  ).AsDateTime);
Edit6.Text := FieldByName('DEPT_NO'    ).AsString;
Edit7.Text := FieldByName('JOB_CODE'   ).AsString;
Edit8.Text := IntToStr(FieldByName('JOB_GRADE'  ).AsInteger);
Edit9.Text := FieldByName('JOB_COUNTRY').AsString;
Edit10.Text := FieldByName('FULL_NAME'  ).AsString;
```

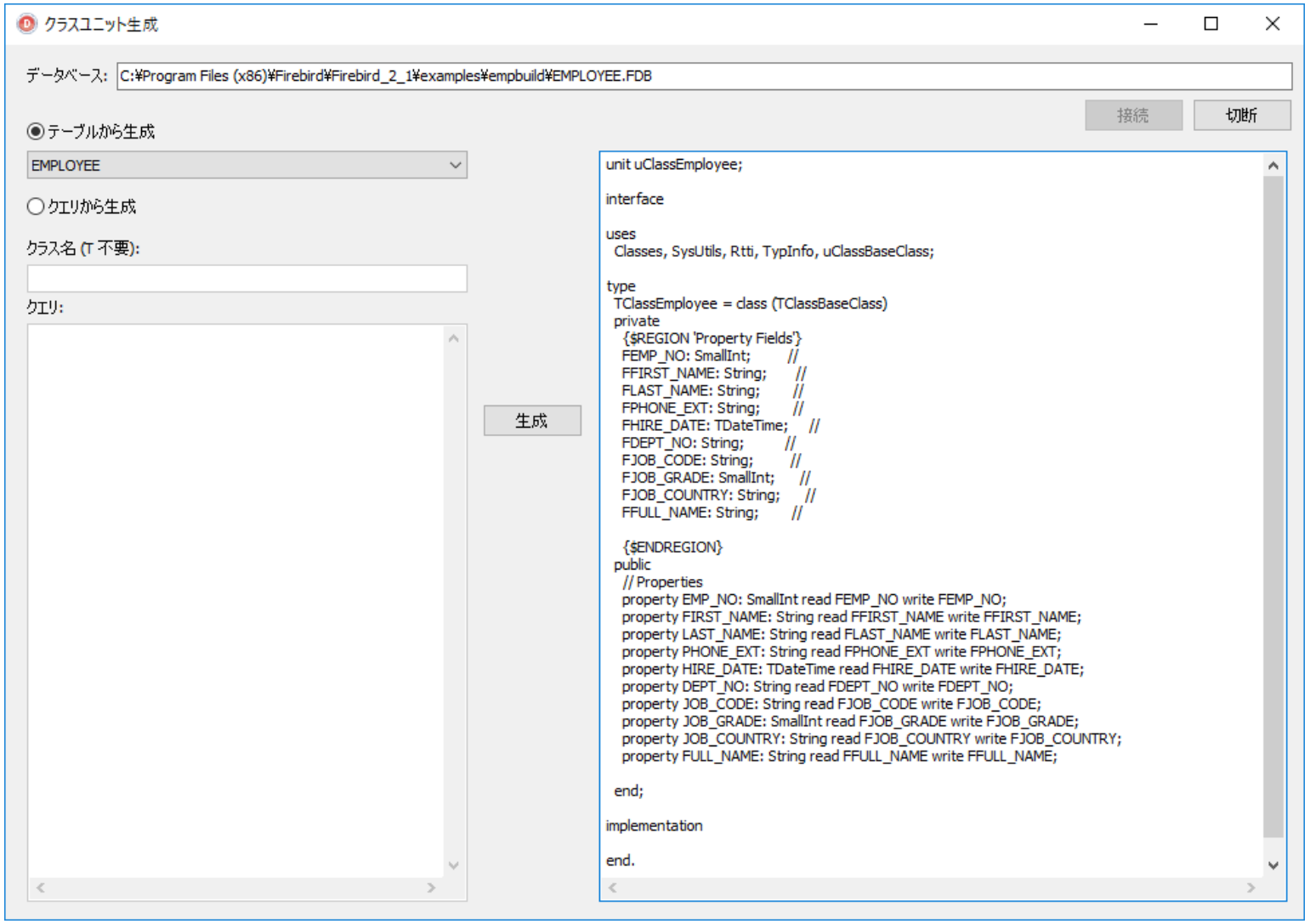
- フィールド名が間違っていても
  実行してみないとわからない。
- 型変換関数のせいで読みにくい。

FieldByName() を書きたくない (2)

```
unit uClassEmployee;

interface

uses
  Classes, SysUtils, Rtti, TypInfo, uClassBaseClass;

type
  TClassEmployee = class (TClassBaseClass)
  private
    FEMP_NO: SmallInt;          //
    FFIRST_NAME: String;        //
    FLAST_NAME: String;         //
    FPHONE_EXT: String;         //
    FHIRE_DATE: TDateTime;      //
    FDEPT_NO: String;           //
    FJOB_CODE: String;          //
    FJOB_GRADE: SmallInt;       //
    FJOB_COUNTRY: String;       //
    FFULL_NAME: String;         //
  public
    property EMP_NO: SmallInt read FEMP_NO write FEMP_NO;
    property FIRST_NAME: String read FFIRST_NAME write FFIRST_NAME;
    property LAST_NAME: String read FLAST_NAME write FLAST_NAME;
    property PHONE_EXT: String read FPHONE_EXT write FPHONE_EXT;
    property HIRE_DATE: TDateTime read FHIRE_DATE write FHIRE_DATE;
    property DEPT_NO: String read FDEPT_NO write FDEPT_NO;
    property JOB_CODE: String read FJOB_CODE write FJOB_CODE;
    property JOB_GRADE: SmallInt read FJOB_GRADE write FJOB_GRADE;
    property JOB_COUNTRY: String read FJOB_COUNTRY write FJOB_COUNTRY;
    property FULL_NAME: String read FFULL_NAME write FFULL_NAME;
  end;

implementation

end.
```

- 実装部には何もないクラスです
- DB から自動生成します
- 何とか手書きもできるレベルです
- テーブルに変更があれば再生成します

FieldByName() を書きたくない (3)



FieldByName() を書きたくない (4)

```
Employee.Dataset := QRY; // クラスとデータセットを関連付け
Employee.GetFields;      // データを取得

Edit1.Text  := IntToStr(Employee.EMP_NO);
Edit2.Text  := Employee.FIRST_NAME;
Edit3.Text  := Employee.LAST_NAME;
Edit4.Text  := Employee.PHONE_EXT;
Edit5.Text  := DateToStr(Employee.HIRE_DATE);
Edit6.Text  := Employee.DEPT_NO;
Edit7.Text  := Employee.JOB_CODE;
Edit8.Text  := IntToStr(Employee.JOB_GRADE);
Edit9.Text  := Employee.JOB_COUNTRY;
Edit10.Text := Employee.FULL_NAME;
```

- 型変換関数が…

拡張 RTTI

FieldByName() を書きたくない (5)

```
Employee.Dataset := QRY; // クラスとデータセットを関連付け
Employee.GetFields;      // データを取得

Edit1.Text  := Employee.EMP_NO.ToString;
Edit2.Text  := Employee.FIRST_NAME;
Edit3.Text  := Employee.LAST_NAME;
Edit4.Text  := Employee.PHONE_EXT;
Edit5.Text  := DateToStr(Employee.HIRE_DATE);
Edit6.Text  := Employee.DEPT_NO;
Edit7.Text  := Employee.JOB_CODE;
Edit8.Text  := Employee.JOB_GRADE.ToString;
Edit9.Text  := Employee.JOB_COUNTRY;
Edit10.Text := Employee.FULL_NAME;
```

- 惜しい！
- TDateTime 型用のヘルパーを書けばよさそう

プリミティブ型ヘルパー

拡張 RTTI

FieldByName() を書きたくない (6)

```
Employee.Dataset := QRY; // クラスとデータセットを関連付け
Employee.GetFields;      // データを取得

Edit1.AsInteger  := Employee.EMP_NO;
Edit2.AsString   := Employee.FIRST_NAME;
Edit3.AsString   := Employee.LAST_NAME;
Edit4.AsString   := Employee.PHONE_EXT;
Edit5.AsDate     := Employee.HIRE_DATE;
Edit6.AsString   := Employee.DEPT_NO;
Edit7.AsString   := Employee.JOB_CODE;
Edit8.AsInteger  := Employee.JOB_GRADE;
Edit9.AsString   := Employee.JOB_COUNTRY;
Edit10.AsString  := Employee.FULL_NAME;
```

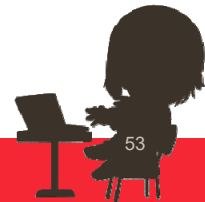
- Edit 側に規則を持たせるのも手

クラスヘルパー

拡張 RTTI

FieldByName() を書きたくない (7)

- テーブルのクラスはテーブルが変更になったら
  再生成するのだから、コード中の変更箇所も判りやすい。
- 例えば ‘FULL_NAME’ フィールドを削除したりリネームした場合
  にはコンパイルした段階で修正しなければならない箇所が判る。
- 追加されたフィールドはコンパイル時には判らないが、
  少なくともエラーにはならない。
- テーブルのクラスではなくレコードで作ってもいいけれど、
  継承ができないのでユニットが冗長になる。



FieldByName() を書きたくない (8)

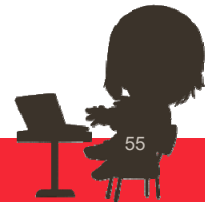
- 先程のテーブル用クラスは Parameter のセットにも使えます。

```
Employee.Dataset := QRY; // クラスとデータセットを関連付け

Employee.EMP_NO      := Edit1.AsInteger;
Employee.FIRST_NAME  := Edit2.AsString;
Employee.LAST_NAME   := Edit3.AsString;
Employee.PHONE_EXT   := Edit4.AsString;
Employee.HIRE_DATE   := Edit5.AsDate;
Employee.DEPT_NO     := Edit6.AsString;
Employee.JOB_CODE    := Edit7.AsString;
Employee.JOB_GRADE   := Edit8.AsInteger;
Employee.JOB_COUNTRY := Edit9.AsString;
Employee.FULL_NAME   := Edit10.AsString;

Employee.SetParams;      // パラメータをセット（SQL に存在するものだけ）
QRY.ExecSQL;
```

右辺と左辺をひっくり返すだけ

まとめ

- 最新版 Delphi を持っていない場合には**無償版でお勉強**。
- 最近の Delphi で何ができるのか、
  DocWiki や **OBJECT PASCAL HANDBOOK** で確認しましょう。
- (特に意味もなく) 古いバージョンに留まっていると、
  面白い書き方や、簡単な書き方がある事に気が付かない事があります。
- 無理矢理、新しい機能で書けとは言いませんが、
  新しいバージョンへ移行しておけば、コードを楽しく簡潔に書く事が可能になります。



THANKS!❤

www.embarcadero.com/jp

第33回 エンバカデロ・デベロッパーキャンプ

- Extras

ライナタン・カクチョウ！

- いつもお世話になっております m(_ _)m
  http://d.hatena.ne.jp/tales/

- 16進数表示プラグイン
- Clipboard-History-Disabler
- Component Initializer
- Component Tray
- DisableStat
- DLight
- dNotes plugin
- Jpfix plugin
- Lock on Load plugin
- New Edit Window for Embedded Designer
- LIB フォルダを読取専用で開く
- Quick Quick Edit
- Sort Projects plugin
- Template Replacer
- UTF8ize plugin
- 凹ンパイルプラグイン
- 痛IDEプラグイン