embarcadero

17th Embarcadero Developer Camp

【A4】Delphi/C++チュートリアル セッション

「SJIS から Unicode へ！ マイグレーションテクニック」

有限会社エイブル
富永 英明

embarcadero

17th Embarcadero
Developer Camp

はじめに

「またですか？」「いえいえ…」

- 11th A6「Delphi 2009 ではじめる Unicode アプリケーション」
  - 既存コード移行のポイント -
  Unicode メインのお話でした。

- 16th 1E「Delphi での文字コードのハンドリングについて」
  文字コード全般のお話でした。

- そして今回は？
  ANSI 版 Delphi (Shift-JIS) からUnicode 版 Delphi (Unicode) へ
  移行するためのお話です。第11回の内容を掘り下げた話になります。

embarcadero

17th Embarcadero Developer Camp

ステップ アップグレード

ステップアップグレードのススメ

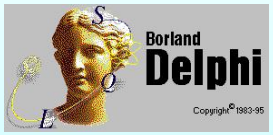
- 古いプロジェクトをUnicode 版 Delphi でコンパイルしようとすると
  様々な問題に直面する 。

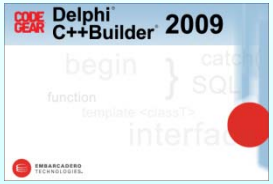
- 段階的にプロジェクトをアップグレードすると
  問題の切り分けが容易になるので、
  結果的には短期間でのマイグレーションが可能となる。

- 一足飛びでプロジェクトをアップグレードすると
  様々な要因が絡み合い、バグなのか仕様なのかを特定するのでさえ困難になってしまう。

- Shift-JIS → Unicode のマイグレーションは、
  即ち、Delphi のバージョンアップを意味する。

Delphi 1 からのステップアップグレード





半分ネタですが、初代 Delphi。

16bit アプリケーション用なので、
ANSI 最終版 である Delphi 2007 へ移行後に Unicode 版 Delphi へ移行するか、
意匠だけを参考に Unicode 版 Delphi で作り直すといいと思います。

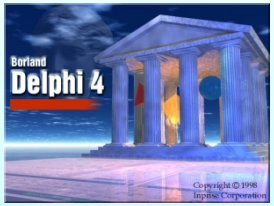
Delphi 1 からのステップアップグレード

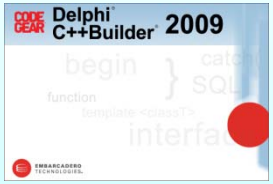
- アップグレードの選択肢
  1. → Delphi 7 → Delphi 2007 → Delphi 最新版
  2. → Delphi 2007 → Delphi 最新版
  3. → Delphi 最新版 (オススメ)

- 理由
  - 保守ならば、Delphi 1 をそのまま使うべき。
  - アップデータの入手が事実上不可能。
  - 最新版 Delphi とは何もかもが違う。

- 移行のポイント
  - 意匠とアルゴリズムを参考に、一から作り直す方が早い。
  - 16bit リソースファイルはそのままでは使えない。

Delphi 2 ～ 5 からのステップアップグレード





Delphi 2 ～ 5 は Win9x のコードが残っている事が多いので、Win9x 対応最終版である Delphi 7 へ移行後、場合によっては更に ANSI 最終版である Delphi 2007 へ移行し、Unicode 版 Delphi へ移行するのが確実です。

Indy 10 対応は Unicode 版 Delphi 移行前に済ませておきます。

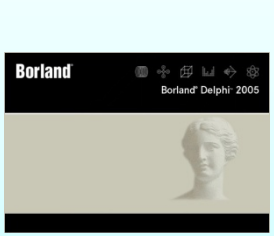
Delphi 2 ～ 5 からのステップアップグレード

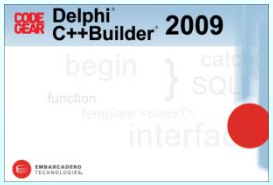
- アップグレードの選択肢
  1. → Delphi 7 → Delphi 2007 → Delphi 最新版 (オススメ)
  2. → Delphi 2007 → Delphi 最新版

- 理由
  - コンポーネントの構造が Delphi 5 / 6 で変更されており、dsgnintf の対応を行わなくてはならない。
  - NetManage コンポーネントを Indy で置き換える必要がある。
  - アップデータの入手が事実上不可能。

- 移行のポイント
  - BDE を使っている場合、代替 DB への移行を行う。
  - QuickReport を使っている場合、RaveReports へ移行するのか、通常版 QuickReport を購入するのか、はたまた帳票ツールを自作するのかを決断しなくてはならない。

Delphi 6～2006 から...





ANSI 版 アプリケーションを保険で作っておく場合には、Delphi 2007 への移行を行う必要があります。

Delphi 2007 は Vista に対応していますので、Alt キーを押下した際の問題も発生しません (“Vista Alt Fix” を参照)。

Indy 10 対応は Unicode 版 Delphi 移行前に済ませておきます。

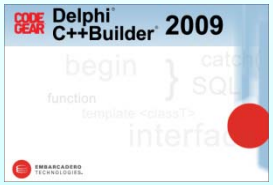
Delphi 6～2006 から…

- アップグレードの選択肢
  1. → Delphi 2007 → Delphi 最新版 (オススメ)
  2. → Delphi 最新版

- 理由
  - Vista 対応を行わなくてはならない。
    (Alt キー問題や、UAC への対応など)

- 移行のポイント
  - Indy 10 への移行を行う。
  - QuickReport を使っている場合、RaveReports へ移行するのか、
    通常版 QuickReport を購入するのか、
    はたまた帳票ツールを自作するのかを決断しなくてはならない。

Delphi 2007 から...



Delphi 2007 → Unicode 版 Delphi への移行は、基本的に文字コード絡みだけ考えれば済みます。

無償版 QuickReport は付属しませんので、場合によっては 製品版を購入します。

ANSI 版 から Unicode 版 への移行は
このパターンが一番楽です (後述)。

Delphi 2007 から…

- アップグレードの選択肢
  1. → Delphi 最新版
- 理由
  - Unicode への対応。Web との連携強化。
  - Windows Vista / 7 への対応。
- 移行のポイント
  - XPMan (.manifest) / VistaAltFix (/ FastMM) の除去。
  - Unicode 対応コードの記述。
  - QuickReport を使っている場合、RaveReports へ移行するのか、通常版 QuickReport を購入するのか、はたまた帳票ツールを自作するのかを決断しなくてはならない。

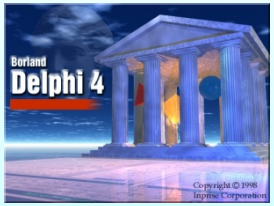
何故、Delphi 7 と Delphi 2007 なのか？

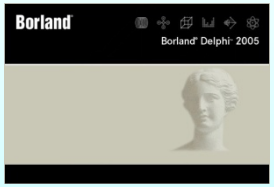
- Delphi 7
  - Win9x 対応最終版なので。
  - Win9x にインストール可能なリモートデバッガが付属している (Pro も)。
  - IDE が Win9x にインストール可能
  - Windows XP 以降のテーマに対応している
  - Win9x 対応が不要ならば必須ではない。
- Delphi 2007 /R2
  - ANSI 対応最終版なので。
  - Vista に対応しており、Alt キー問題も発生しない
  - Unicode 版 Delphi に限りなく近いため、Delphi 2007 からのマイグレーションだと基本的に文字コード絡みの問題しか起きない。そういった意味で問題発生時の原因の切り分けが非常に楽。

Delphi と Windows の発売年の関係

年	バージョン	コンパイラ	対応OS	Windows 発売		
				9x系	NT系	Server系
1995	Delphi	16bit (ANSI)	Windows 3.1	Windows 95	Windows NT3.51	
1996	Delphi 2	32bit (ANSI)	Windows 3.1, 95, NT3.51		Windows NT4.0	
1997	Delphi 3 / 3.1		Windows 3.1, 95, NT3.51(SP5), NT4.0			
1998	Delphi 4		Windows 95, 98, NT4.0 (SP3)	Windows 98		
1999	Delphi 5		Windows 95, 98/SE, NT4.0 (SP3)	Windows 98 SE		
2000				Windows Me	Windows 2000	
2001	Delphi 6		Windows 98/SE, Me, NT4.0 (SP5) , 2000	–	Windows XP	
2002	Delphi 7		Windows 98/SE, Me, 2000, XP			
2003	(Delphi 8)					Server 2003
2004	Delphi 2005		Windows 2000 (SP2), XP, Server 2003			
2005	BDS 2006		Windows 2000 (SP4), XP (SP2), Server 2003 (SP1)			
2006	Turbo Delphi 2006		Windows 2000 (SP4), XP (SP2), Server 2003 (SP1)			Server 2003 R2
2007	Delphi 2007 / R2		Windows 2000 (SP4), XP (SP2), Server 2003 (SP1), Vista		Windows Vista	
2008	Delphi 2009	32bit (Unicode)	Windows 2000 (SP4), XP (SP3), Server 2003 (SP1), Server 2008, Vista			Server 2008
2009	Delphi 2010		Windows XP (SP3), Server 2003 (SP1), Server 2008, Vista (SP1), 7		Windows 7	Server 2008 R2
2010	Delphi XE		Windows XP (SP3), Server 2003 (SP1), Server 2008, Vista (SP1), 7			

ブランチ





Win9x 対応

ANSI 最終版

Unicode 版

<!>一つのソースで全対応させようと思わない事 <!>

Delphi 2007 はもう入手できないのでは？

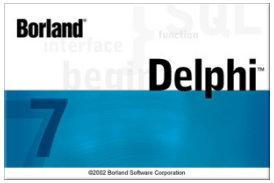
- RAD Studio 2007 R2 及び、Delphi 2007 for Win32 R2 はアップグレード版こそないものの、通常版は販売継続対象製品となっている。
- ToolCloud でも Delphi 2007 を入手可能。
- それと…

いくらなんでも Delphi 7 はもう入手できないのでは？

- そこで、Delphi XE ですよ。

- Delphi XE を購入すると…
  - Delphi 7 / 2007 / 2009 / 2010 が追加費用なしに利用可能です 。
    (アカデミック版は除く)

- RAD Studio XE なら加えて以下の開発環境が利用可能。
  - C++ Builder 6 / 2007 / 2009 / 2010
  - Delphi Prism 2009 / 2010 / 2011
  - RadPHP (旧 Delphi for PHP)



仮想化ソフトウェア (1)

- 古い OS でアプリケーションを検証したり、デバッグしたい場合には、仮想 PC を利用するといい。
  - [Microsoft Virtual PC 2007 SP1]
    http://www.microsoft.com/downloads/details.aspx?FamilyId=28C97D22-6EB8-4A09-A7F7-F6C7A1F000B5&displaylang=ja
    XP / Vista でも利用可能。7 の “Windows Virtual PC” とは排他利用。
  - [Windows Virtual PC]
    http://www.microsoft.com/downloads/details.aspx?familyid=2B6D5C18-1441-47EA-8309-2545B08E11DD&displaylang=ja
    Windows 7 用。VT 対応 CPU が必要。XP モードは必須ではない。
  - [Oracle VM VirtualBox]
    http://www.virtualbox.org/wiki/Downloads
    USB が使える他、DirectX にも対応している。
    Open Source Edition もあるが、こちらは自前でビルドしなくてはならない。

仮想化ソフトウェア (2)

- 仮想化ソフトウェアを使うことのメリット
  - そもそも、最近の PC には古い OS がインストールできない。
    - ドライバ類が揃わない事が多い。
    - 下手をすると、今となっては割高な PATA-HDD を購入しなくてはならない。
  - PC を何台も置くスペースが不要。
    - KVM 切り替え器も不要。
  - スナップショット機能を利用すれば、設定した時点の環境に復元する事ができる。
    - クリーンインストール環境を作るために何度も OS をインストールしなくていい。
    - Delphi を何度もインストールして、開発環境を整備しなくて済む。
- ステップアップグレードのお供に。

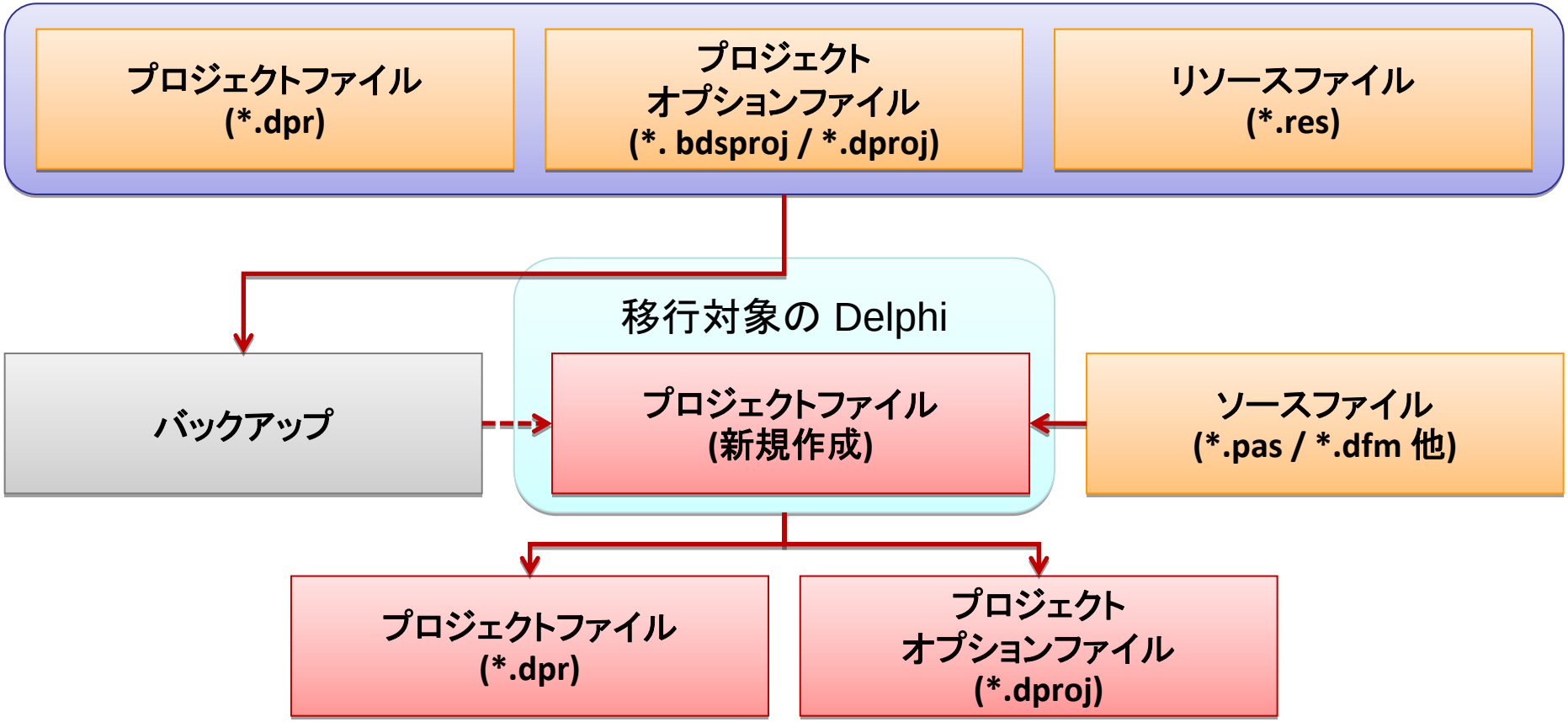
embarcadero

17th Embarcadero Developer Camp

プロジェクトファイル

古いプロジェクトファイルの移行

- *.dpr は捨てて再生成。
  自動アップグレードでうまくいく事もあるが、いかない事も多い。



Delphi 7 への移行でやるべき事

- Win9x と XP 以降を動作対象にするなら…
  *.manifest ファイルがあるなら破棄し、XPMan を uses する。

- Vista 以降も動作対象にするなら…
  Vista Alt Fix (http://cc.embarcadero.com/item/24282) を uses する。

- Dsgnintf に対する対応を行う (コンポーネント)

```
// [旧]
uses
  ..., DsgnIntf;

// [新]
uses
  ..., DesignIntf, DesignEditors, VCLEditors, RTLConsts;
```

- フォームファイルを テキスト形式 DFM へコンバートする
  convert.exe -i -s -t *.dfm

Delphi 2007 への移行でやるべき事

- XPMan を削除
  プロジェクトオプションでテーマを有効にする事で同等となる。
- FastMM を削除
  BDS2006 以降で同等のメモリマネージャが採用されている。
- Vista Alt Fix を削除
  Delphi 2007 以降では不要になっている。
- Application.MainFormOnTaskbar := True; を指定
  メインフォームをタスクバーへ格納する事が可能となる。
- ReportMemoryLeaksOnShutdown := True; を指定
  アプリケーション終了時にメモリリークをレポートしてくれる。
- (あれば) CLX コードの除去。

Unicode 版 Delphi への移行でやるべき事

- 下記のコンパイラ指令を有効にする
  - 文字欠損の検索を行う
    ```
    {$WARN IMPLICIT_STRING_CAST ERROR}
    {$WARN IMPLICIT_STRING_CAST_LOSS ERROR}
    {$WARN WIDECHAR_REDUCED ERROR}
    ```
    暗黙の文字コード変換をエラーに昇格させる。

  - 文字列形式の代入時にチェックしない (XE では常に OFF)
    ```
    {$STRINGCHECKS OFF}
    ```
    C++Builder との連携がない、或いは 連携していても AnsiString への代入を行わない場合は OFF で構わない。
    生成される EXE のサイズが多少小さくなり、実行速度も速くなる。

  - RTTI を可能な限りオフ (Delphi 2010 以降)
    ```
    {$RTTI EXPLICIT METHODS([]) FIELDS([]) PROPERTIES([])}
    ```
    RTTI の機能を積極的に利用する事がない場合に。
    生成される EXE のサイズが小さくなる。

embarcadero

17th Embarcadero Developer Camp

コンポーネント

ネットワーク関連のコンポーネント

- NEWT Intranet / NetManage / FastNet
  - Indy とは完全に互換性がない。
  - Indy への移行をお早めに。
- TServerSocket / TClientSocket
  - [コンポーネント | パッケージのインストール] で “$(BDS)¥bin¥dclsocketsxxx.bpl” をインストールすれば利用可能。

- Indy 10
  - ftp://indy.fulgan.com/ZIP/ から最新版を DL 可能
  - ANSI 最終版 (10.2.3) は “Indy10.zip*”
  - Unicode 版は "IndyTiburon.zip"

データベース関連のコンポーネント

- BDE (Borland Database Engine)
  - Delphi 6 付属の 5.2 が最新
  - 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る
  - 他のデータベースコンポーネントへの移行をオススメ
- IBX (InteBase Express)
  - Delphi 7 用の最新版は 7.11 で、Delphi 2007 付属のものと同等。
    http://cc.embarcadero.com/Item/24267
- dbGo (ADO Express)
  - BDE で ODBC 接続しているのなら、ADO の ODBC 接続へ
- DBX (db Express)
  - Delphi 6 から付属。Oracle とかならコレ。

帳票系のコンポーネント

- QuickReport
  - Delphi 7 用 QuickReport 3.5.1 Standard
    http://www.quickreport.co.uk/WebInfoPage.aspx?WebInfoID=620
  - Delphi 2007 用 QuickReport 4.0.6 Standard
    http://cc.embarcadero.com/Item/25002
  - Unicode 版 Delphi には無償で使える Standard 版は付属しない。
  - 有償である Professional 版の購入を検討する or
    RaveReports への変更を検討する or
    印刷コンポーネント (またはクラス) を自作する。
- RaveReports
  - QuickReport とは互換性がない。
  - Delphi 7 から最新版 Delphi までに付属する。

その他のコンポーネント

- サードパーティ製
  - DCU で提供されているものや、ActiveX / OCX は、
    対象となる Delphi に対応しているか確認する。

- フリーソフト系
  1. 対象となる Delphi に対応するよう、作者にお願いしてみる。
  2. 作者が対象となる Delphi を所持していないのなら、
     Delphi をプレゼントしてみる。
  3. 作者と連絡がつかないのであれば、代替コンポーネントを探す。

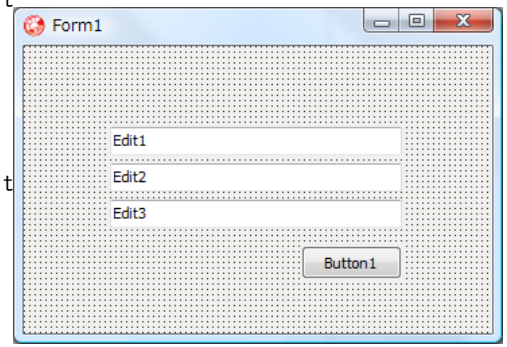
- Project JEDI (JCL / JVCL)
  - http://jvcl.delphi-jedi.org/
    膨大な数のコンポーネントライブラリ集。
    代替となるコンポーネントがあるかも？

(ビジュアル) コンポーネントの置換

- サポートされないコンポーネントは代替コンポーネントで置換
  - 代替コンポーネントもないのにプロジェクトを開くと、
    フォームが開けず、全体的なレイアウトすら把握できない。
  - 代替コンポーネントには系統の似たコンポーネントを選択する。
  - 「あー、でも代替コンポーネントを貼り直すのが面倒くさいなぁ…」
  - 代替コンポーネントを利用する場合には
    ソースファイルを編集して、コンポーネントの置換を行う。
- では、例として、TEdit を TComboBox へ置換してみる。

0.サンプル


```
object Form1: TForm1
  Left = 353
  Top = 154
  Caption = 'Form1'
  ClientHeight = 213
  ClientWidth = 348
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  OldCreateOrder = False
  PixelsPerInch = 96
  TextHeight = 13
  object Edit1: TEdit
    Left = 64
    Top = 60
    Width = 217
    Height = 21
    TabOrder = 0
    Text = 'Edit1'
  end
  object Edit2: TEdit
    Left = 64
    Top = 87
    Width = 217
    Height = 21
    TabOrder = 1
    Text = 'Edit2'
  end
  object Edit3: TEdit
    Left = 64
    Top = 114
    Width = 217
    Height = 21
    TabOrder = 2
    Text = 'Edit3'
  end
  object Button1: TButton
    Left = 206
    Top = 148
    Width = 75
    Height = 25
    Caption = 'Button1'
    TabOrder = 3
    OnClick = Button1Click
  end
end
```




```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls,
  Forms,  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    Private 宣言
  public
    Public 宣言
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit3.Text := IntToStr(StrToInt(Edit1.Text) +
StrToInt(Edit2.Text));
end;
end.
```

1.フォームファイル (*.dfm) の編集

1. 事前に テキスト形式 DFM へ変換しておく。
2. *.dfm を テキストエディタで開く。
3. “object <コントロール名>: TEdit” になっている所を検索し、TEdit を TComboBox へ置換する。

```
object Edit1: TEdit
  Left = 64
  Top = 60
  Width = 217
  Height = 21
  TabOrder = 0
  Text = 'Edit1'
end
object Edit2: TEdit
  Left = 64
  Top = 87
  Width = 217
  Height = 21
  TabOrder = 1
  Text = 'Edit2'
end
object Edit3: TEdit
  …
```

```
object Edit1: TComboBox
  Left = 64
  Top = 60
  Width = 217
  Height = 21
  TabOrder = 0
  Text = 'Edit1'
end
object Edit2: TComboBox
  Left = 64
  Top = 87
  Width = 217
  Height = 21
  TabOrder = 1
  Text = 'Edit2'
end
object Edit3: TComboBox
  …
```

2.ソースファイル (*.pas) の編集

1. *.pas を テキストエディタで開く。
2. フォームクラス宣言部にある TEdit を TComboBox へ置換する。
3. 代替コンポーネントに存在しないイベントハンドラが事前に判るのなら、
   とりあえずコメントアウトして潰しておく。

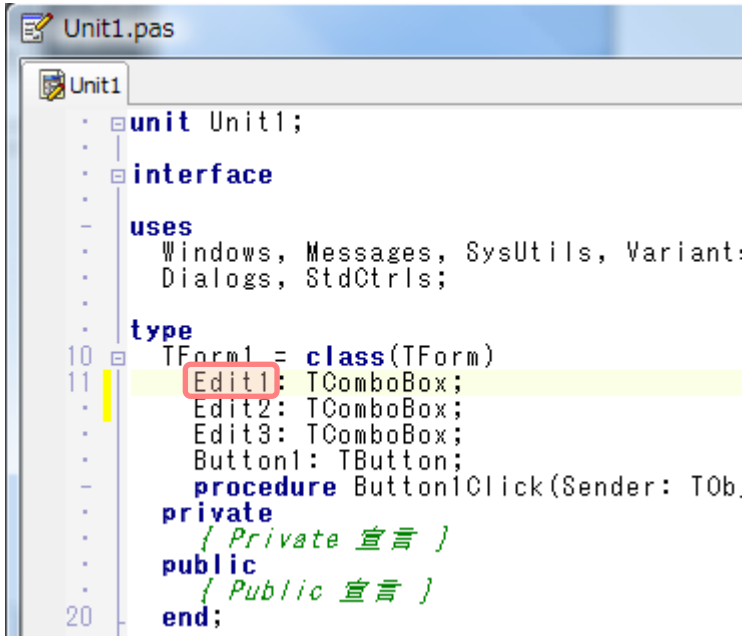
```
type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
      { Private 宣言 }
  public
      { Public 宣言 }
  end;
```

```
type
  TForm1 = class(TForm)
    Edit1: TComboBox;
    Edit2: TComboBox;
    Edit3: TComboox;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
      { Private 宣言 }
  public
      { Public 宣言 }
  end;
```

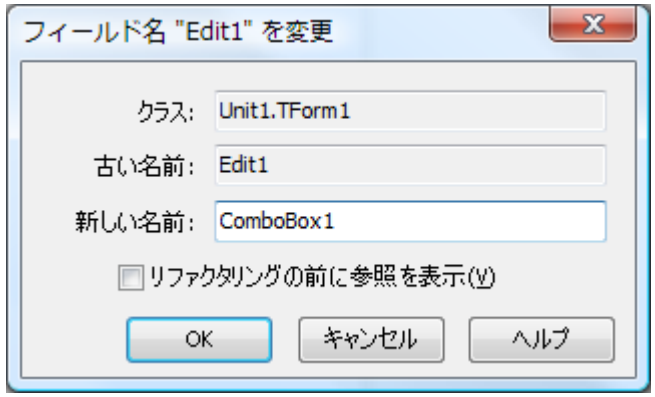
3.仕上げ (その1)

- フォームが表示できないようなら、再度 *.dfm をテキストエディタで開き、存在しないプロパティ / イベントハンドラを削除する。

- コントロールの名前を変更する必要があれば、リファクタリングで名前を変更する (同期編集ではフォーム側の名前が変更されないので駄目)。

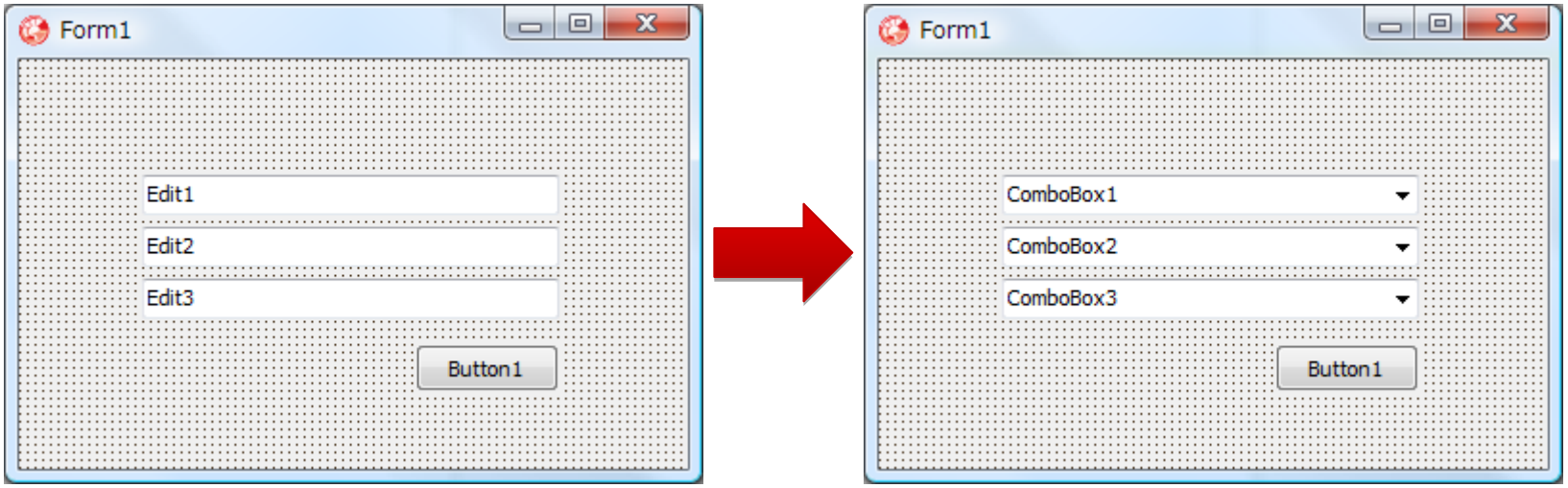


Edit1 にカーソルを置いた状態で
Ctrl + Shift + E
でコントロール名を変更できる。



4.仕上げ (その2)

- コンパイルしてみてエラーになる箇所はとりあえずコメントアウトし、対処方法は後で考える。
- この方法だと、コントロールのサイズやタブオーダー、イベントハンドラがそのままなので、比較的楽に代替コンポーネントに置き換えられる。
- 正規表現による置換が行えるエディタを使うと *.dfm の編集が楽。



embarcadero

17th Embarcadero
Developer Camp

最近の Delphi で知ってるとちょっと便利な事

知っておくと便利なツール類

- XN Resource Editor
  http://cc.embarcadero.com/Item/25783
  *.dcr にも対応しているリソースエディタ。

- $(BDS)¥bin¥convert.exe / $(Delphi)¥bin¥convert.exe
  フォームファイルをテキスト形式 / バイナリ形式に変換する。

- $(BDS)¥bin¥ CGRC.EXE (2009 以降)
  Unicode に対応したリソースコンパイラ / バインダ。BRCC32.EXE の代替に。

- $(BDS)¥bin¥ BDSSetLang.exe (2010 以降)
  IDE の UI / ライブラリのリソース文字列を “日英独仏 “ へ切り替える。

- $(BDS)¥bin¥Formatter.exe (XE 以降)
  ソースコードフォーマッタ (コード整形) のスタンドアロン版。

知っておくと便利な for in do (1)

- 列挙イロイロ

```
const
  MIA: array [0..2] of string =
    ('Forest Speyer', 'Brad Vickers', 'Rebecca Chambers');
var
  c: Char;
  i: Integer;
  s: String;
begin
  Memo1.Lines.Clear;
  // 集合型
  for c in ['A', 'C'..'F', 'Z'] do
    Memo1.Lines.Add(c);
  // 静的配列
  for s in MIA do
    Memo1.Lines.Add(s);
  // 数値ジェネリック配列 (2010 以降)
  for i in TArray<Integer>.Create(0131, 0513, 4011, 4312) do
    Memo1.Lines.Add(IntToStr(i));
  // 文字ジェネリック配列 (2010 以降)
  for s in TArray<string>.Create('AQUACURE','SAFSPRIN', 'ADRAVIL') do
    Memo1.Lines.Add(s);
end;
```

※ for in do は Delphi 2005 から利用可能。Generics は Delphi 2009 から。

知っておくと便利な for in do (2)

- サブディレクトリを含めたファイルの列挙 (2010 以降)

```
uses
  ..., IOUtils;

var
  FileName: String;
begin
  for FileName in TDirectory.GetFiles('C:¥TEST', '*.pas',
TSearchOption.soAllDirectories) do
    Memo1.Lines.Add(FileName);
end;
```

- 正規表現による一致文字列の抜き出し (XE 以降)

```
uses
  ..., RegularExpressions;

var
  Match: TMatch;
begin
  for Match in TRegEx.Matches('ABCABCABC', 'C') do
    Memo1.Lines.Add(Format('%s (%d, %d)', [Match.Value, Match.Index, Match.Length]));
end;
```

embarcadero

17th Embarcadero
Developer Camp

文字コード関連

“3(4)回目”なので簡単に？

- 文字コードそのものに対する話は (さすがに) 省略。
- 本項以降は Delphi 2007 への移行が終わったものとして話を進める。
  - Delphi 2007 → Unicode 版 Delphi への移行においては基本的に文字コード絡みの問題しか起きないため、問題発生時の原因の切り分けが非常に楽になる。
  - 大まかな Delphi 2007 → Unicode 版 Delphi への移行の手法は、11th A6 「Delphi 2009 ではじめる Unicode アプリケーション - 既存コード移行のポイント」の資料を参考に。
  - その他の文字コードに関する情報は 16th 1E「Delphi での文字コードのハンドリングについて」の資料を参考に。

約束事

- ANSI ベースのアプリケーションは作らない
  ANSI ベースのアプリケーションを作るのなら Delphi 2007 で。

- {$IFDEF UNICODE} による条件コンパイルを行わない。
  ANSI には ANSI の、Unicode には Unicode の問題点がある。

- 旧来使っていた文字コード変換ライブラリは極力使わない。
  - 代入による暗黙の文字コード変換が可能
  - TEncoding クラスや文字コード変換関数が存在する
  - Indy にも文字コード絡みのクラス / 関数が存在する。

- Indy は最新版を基本とする。

- データベースは Unicode に対応しているものを基本とする。

embarcadero

17th Embarcadero Developer Camp

Indy

Indy – バージョンについて

- 本項のソースコードで使われている Indy のバージョンは 10.5.7.0
  (XE は 10.5.8)
- 本資料執筆時点での最新版 r4358 (2010/08/24) を
  Delphi 2010 へインストールしている。
- 新旧バージョンの問題が混在する事を避けるため、
  本資料では一貫して r4358 を利用している。
- 今日時点の最新版では不具合が解消している可能性がある。
- 不具合があったら、まずは最新版をインストールし、
  それでも不具合が出るようであれば、フォーラムや QC で報告。
  http://forums2.atozed.com/viewforum.php?f=6

Indy (FTP)

- 文字コードはまず関係しない。
  関係したとしても、Unicode 版 Delphi に限った話ではない。

- テキストファイルであっても Binary モードで転送。
  ASCII モードによる改行コード変換機能を使わず、
  改行コードは転送元であらかじめ変換しておき、
  Binary モードで転送したほうがいい。

  - UTF-16 ファイルの改行コードが変換されたら？
  - バイト列 “0x0D 0x00 0x0A 0x00” はどう解釈されて変換されるのか？

- FTP で日本語ファイル名を使うべきではない。
  FTP サーバ (ホスト) 側でファイル名に使われる文字コードと
  クライアント側でファイル名に使われる文字コードが一致しない場合は
  ファイル名に日本語等を用いるとファイル名が文字化けする。

Indy (FTP) – 日本語ファイル名

- Indy (TIdFTP) では、ファイル名を バイト列で処理する。
  - ‘ABC’ は Unicode で U+0041 U+0042 U+0043 だが、
    これは 0x41 0x42 0x43 として処理される。
  - ‘あいう’ は Unicode で U+3042 U+3044 U+3046 なので、
    0x42 0x44 0x46 となってしまう。
  - ANSI 版の時は String も Byte だったので問題はなかった。
  - ‘あいう’ は 0x82 0xA0 0x82 0xA2 0x82 0xA4 なのでそのまま通る。
- 日本語ファイル名を通すには、ファイル名を
  サーバ側の文字コードでエンコーディングしたものを、
  Byte列→ Word 列変換し TIdFTP のファイル名に渡す。

Indy (FTP) – 日本語ファイル名のアップロード


```
uses
  ..., IdFTP, IdFTPCommon, MECSUtils;

procedure TForm1.Button1Click(Sender: TObject);
var
  FTP: TIdFTP;
  U8: UTF8String;                          // ホストでの文字コード (UTF-8)
  Src, Dst: UnicodeString;
begin
  Src := 'C:¥Document and Settings¥John¥Desktop¥ある研究員の手紙.eml';
  U8  := UTF8String(ExtractFileName(Src)); // ファイル名だけを抽出し、UTF-8 に。
  Dst := MecsStretchElement(U8);           // UTF-8 バイト列をワード列に押し込む。
  FTP := TIdFTP.Create;
  try
    with FTP do
      begin
        Host     := 'ftp.umbrella.com';
        Username := 'john';
        Password := 'ada';
        Connect;
        if Connected then
          begin
            TransferType := ftBinary;      // Binary モード
            ChangeDir('/docs/');
            Put(Src, Dst, False);
            ShowMessage('Done.');
            Disconnect;
          end;
      end;
  finally
    FTP.Free;
  end;
end;
```

ホストでの文字コードが UTF-8 の場合、Shift_JIS ファイル名でアップロードすると文字化けしてしまいます。

FTP クライアントで確認すると文字化けしていないように見えるものも、ホスト側から見れば文字化けしています。

例えば、 のように html からリンクを張っても、普通にアップロードした場合には "404 not found" になってしまい、html からは参照できません。

Indy (FTP) – 日本語ファイル名

FTP クライアント
(Shift-JIS)

“ある研究員の手紙.eml “
アップロード!!

FTP サーバ
(UTF-8)

0x82 0xA0 0x82 0xE9 0x8C 0xA4
0x8B 0x86 0x88 0xF5 0x82 0xCC
0x8E 0xE8 0x8E 0x86 .eml
なんじゃこりゃ？

Indy (FTP) – 日本語ファイル名

FTP クライアント
(Shift-JIS)

“ある研究員の手紙.eml “
ってある？

FTP サーバ
(UTF-8)

0x82 0xA0 0x82 0xE9 0x8C
0xA4
0x8B 0x86 0x88 0xF5 0x82
0xCC
0x8E 0xE8 0x8E 0x86 .eml
は、あるよ。

Indy (FTP) – 日本語ファイル名

クライアント
(Web ブラウザ)

<a href = “ある研究員の手紙.eml “ > クリック！
404 Not found?
なんで？

HTTP サーバ

“ある研究員の手紙.eml “？
そんな名前のファイルはない。

Indy (FTP) – 日本語ファイル名

SSH クライアント

ls –all
あれ？
“ある研究員の手紙.eml” が
ないぞ？

サーバ

“ある研究員の手紙.eml “？
そんな名前のファイルはない。

Indy (FTP) –日本語ファイル名のダウンロード


```
uses
  ..., IdFTP, IdFTPCommon, MECSUtils;

procedure TForm1.Button1Click(Sender: TObject);
var
  FTP: TIdFTP;
  U8: UTF8String;                          // ホストでの文字コード (UTF-8)
  Src, Dst: UnicodeString;
begin
  U8  := UTF8String('ある研究員の手紙.eml');
  Src := MecsStretchElement(U8);     // UTF-8 バイト列をワード列に押し込む。
  Dst := 'C:¥Document and Settings¥Ada¥Desktop¥' + UnicodeString(U8);
  FTP := TIdFTP.Create;
  try
    with FTP do
      begin
        Host     := 'ftp.umbrella.com';
        Username := 'john';
        Password := 'ada';
        Connect;
        if Connected then
          begin
            TransferType := ftBinary; // Binary モード
            ChangeDir('/docs/');
            Get(Src, Dst, True, True);
            ShowMessage('Done.');
            Disconnect;
          end;
      end;
  finally
    FTP.Free;
  end;
end;
```

ダウンロードもアップロード時と同様となります。

…しかし、ホストの文字コードが UTF-8 である場合はまだいいのですが(“UnicodeString = UTF-16” とはロスレス変換なので)、EUC-JPだったり、Shift_JIS だったりすると話はさらにややこしくなってきます。

理由は事項で。

Indy (FTP) – ホストが UTF-8 でないと？

- ホストの文字コードが EUC-JP や Shift_JIS の場合、先程のコードに

```
type
  CP20932 = type AnsiString(20932); // EUC-JP
  CP932   = type AnsiString(932);   // Shift_JIS
var
  EUCJP: CP20932;
  SJIS: CP932;
begin
  …
```

このように追記し EUCJP や SJIS で置き換えればいい…のだが。

- EUC-JP や Shift_JIS に存在しない Unicode 文字は化けてしまう。
  正確には、代替文字 “?” で置換されてしまう。
  ファイル名において “?” は “1 文字のワイルドカード” なので、
  正しく処理されない (詳細は後述)。

- ファイル名をわざとラウンドトリップさせた後に、代替文字 “?” があれば
  “代替文字を除去する / エラーとして Abort する” 等の処置が必要となる。

Indy (HTTP) - Get() その1

- 引数が一つしかない TIdHttp.Get() を使う場合

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  IdHTTP1.Response.CharSet := 'utf-8';
  Memo1.Lines.Add(IdHTTP1.Get('http://www.umbrella.com'));
end;
```

  - 最初からサイトの文字エンコーディングが判明している場合に使う。
  - 戻り値は String (UnicodeString) なので、文字コードの変換が行われる。
    つまり、本来の文字ではない文字になっている可能性がある。
    (サイトの文字エンコーディングが UTF-8 ならロスレス変換)
  - ファイルとしてダウンロードする用途には使えない。

Indy (HTTP) - Get() その2

- TIdHttp.Get() のオーバーロードされたメソッドを使う場合

```
procedure TForm1.Button1Click(Sender: TObject);
var
  HTTP: TIdHttp;
  MS: TMemoryStream;
  URL: String;
  FileName: String;
begin
  HTTP := TIdHttp.Create;
  MS := TMemoryStream.Create;
  try
    URL := 'http://www.umbrella.com/';
    HTTP.Get(URL, MS);                                  // 第 2 引数が TStream
    MS.Position := 0;
    // ファイルとして DL
    FileName :=  'index.html';
    MS.SaveToFile(FileName);
    // Memo に読み込み
    Memo1.Lines.LoadFromStream(MS, TEncoding.UTF8); // サイトの文字コードは UTF-8
  finally
    MS.Free;
    HTTP.Free;
  end;
end;
```

- こちらは応用が利いて使い勝手がいい。

Indy (HTTP) – 日本語ファイル名

- html ファイル名が日本語の場合。
  「ファイル名に日本語はやめなさい」と言いたいトコロですが…

```
uses
  ..., IdURI;

var
  URL: string;
begin
  URL := TIdURI.URLEncode('http://www.umbrella.com/D.I.J.の日記.html');
  ...
```

  - 少なくとも FTP の時のような無茶はしなくて済む。
  - TIdFTP のファイル名に対して URLEncode() を用いる事はできない。

```
var
  URL: string;
  Enc: TEncoding;
begin
  Enc := TEncoding.GetEncoding(20932); // EUC-JP
  URL := TIdURI.URLEncode('http://www.umbrella.com/D.I.J.の日記.html', Enc);
  ...
  Enc.Free;
  ...
```

  - URL エンコードに使う文字エンコーディングを指定する事も可能。

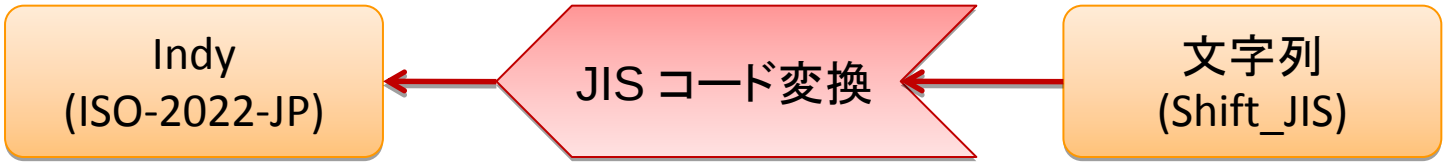
embarcadero

17th Embarcadero Developer Camp

休憩

Indy (Mail 関連)

- もはや文字コードを自前で変換する必要はない。
  古いソースコードに文字コード変換が存在するのなら除去する。
- 古い Indy の場合



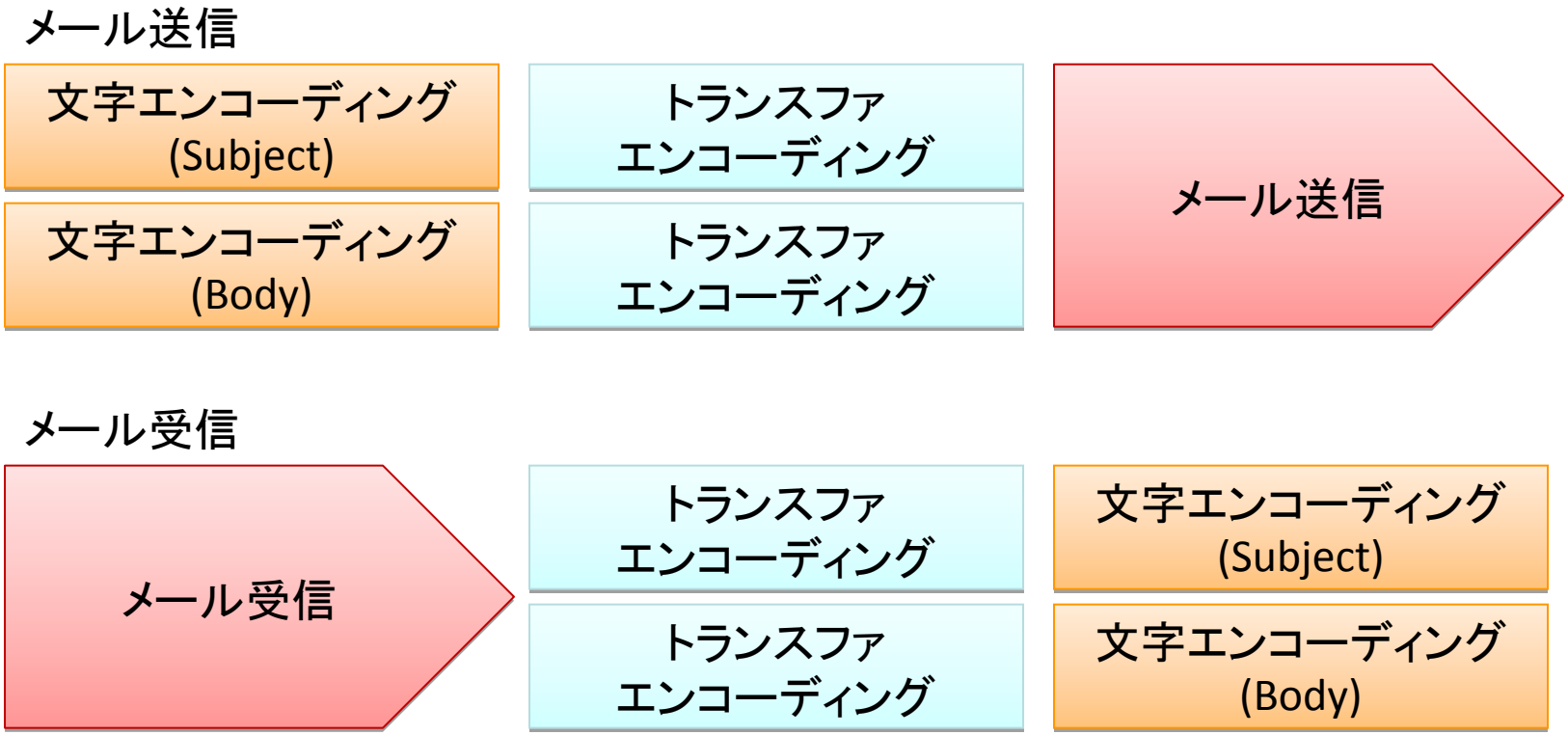
- 新しい Indy の場合 (ANSI 版 Delphi)

- 新しい Indy の場合 (Unicode 版 Delphi)

Indy (Mail 関連)

- QuickSend() は使わない。
  - 日本語環境の場合、Body の文字エンコーディングが何であろうと Subject 部は “必ず ISO-2022-JP になってしまう” ので、場合によっては文字化けする (Subject と Body のエンコーディングが異なるため)。
  - Send() をラッピングして同等の関数を作ったほうがいい。
- 文字コードが何であれ、Subject も Body も BASE64 化する事 (SMTP)。
  - BASE64 化すれば、7bit エンコーディングになるのでトラブルが少ない。
  - 受け取る側 (メーラ等) で文字化けを起こす可能性が低下する。
- UTF-8 ならロスレスなので、可能な限り UTF-8 を利用し、ISO-2022-JP のメールは利用しない事 (SMTP)。
  - Unicode → ISO-2022-JP では文字欠損が起きる。
  - 所謂、”波ダッシュ問題” が発生する。

Indy (Mail 関連)



トランスファエンコーディングには 8 bit でそのまま送る “8bit”、7 bit で送る “7bit”、文字エンコーディングを 7 bit 化する “BASE64” / “Quoted-Printable” があります。Subject と Body の文字エンコーディングは同じにして、“BASE64” または “Quoted-Printable” で 7 bit 化して送信すればトラブルに遭遇する確率が下がります。

Indy (SMTP) – ISO-2022-JP


```
procedure TForm1.IdMessage_InitializeISO(var VHeaderEncoding: Char; var VCharSet: string);
begin
  VHeaderEncoding := 'B'; // 念のために BASE64 エンコーディング
  VCharSet := 'ISO-2022-JP';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  SMTP: TIdSMTP;
  Msg : TIdMessage;
begin
  SMTP := TIdSMTP.Create(nil);
  try
    SMTP.Host      := 'smtp.rpd-stars.com';
    SMTP.Port      := 25;
    SMTP.Connect;
    Msg := TIdMessage.Create(SMTP);
    try
      Msg.OnInitializeISO           := IdMessage_InitializeISO;
      Msg.ContentType               := 'text/plain';
      Msg.CharSet                   := 'ISO-2022-JP';
      Msg.ContentTransferEncoding   := 'BASE64';                          // 念のために BASE64 エンコーディング
      Msg.From.Address              := 'albert.wesker@rpd-stars.com';     // 送信者のメールアドレス
      Msg.Recipients.EMailAddresses := 'richard.keel@racoon-city.com';    // 送信先のメールアドレス
      Msg.Subject                   := MecsMappingFix_UnicodeToJISX0208('例の洋館について');      // 件名
      Msg.Body.Text                 := MecsMappingFix_UnicodeToJISX0208('黄道特急事件の件は…'); // 本文
      SMTP.Send(Msg);
    finally
      Msg.Free;
    end;
    SMTP.Disconnect;
  finally
    SMTP.Free;
  end;
end;
```

文字コード変換は不要ですが、
Unicode → ISO-2022-JP 変換なので、
所謂 “波ダッシュ問題” が発生します。

Indy (SMTP) – UTF-8


```
procedure TForm1.IdMessage_InitializeISO(var VHeaderEncoding: Char; var VCharSet: string);
begin
  VHeaderEncoding := 'B'; // BASE64 エンコーディング (7 bit 化)
  VCharSet :=  'UTF-8';
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  SMTP: TIdSMTP;
  Msg : TIdMessage;
begin
  SMTP := TIdSMTP.Create(nil);
  try
    SMTP.Host      := 'smtp.rpd-stars.com';
    SMTP.Port      := 25;
    SMTP.Connect;
    Msg := TIdMessage.Create(SMTP);
    try
      Msg.OnInitializeISO           := IdMessage_InitializeISO;
      Msg.ContentType               := 'text/plain';
      Msg.CharSet                   := 'UTF-8';
      Msg.ContentTransferEncoding   := 'BASE64';                    // BASE64 エンコーディング (7 bit 化)
      Msg.From.Address              := 'albert.wesker@rpd-stars.com'; // 送信者のメールアドレス
      Msg.Recipients.EMailAddresses := 'richard.keel@racoon-city.com'; // 送信先のメールアドレス
      Msg.Subject                   := '例の洋館について';            // 件名
      Msg.Body.Text                 := '黄道特急事件の件は…';         // 本文
      SMTP.Send(Msg);
    finally
      Msg.Free;
    end;
    SMTP.Disconnect;
  finally
    SMTP.Free;
  end;
end;
```

UTF-8 と Unicode なのでロスレス変換になります。文字のマッピングを変更する必要はありません。

Indy (POP3)


```
procedure TForm1.Button2Click(Sender: TObject);
var
  POP3: TIdPOP3;
  Msg : TIdMessage;
  i, MsgCnt: Integer;
begin
  Memo1.Clear;
  POP3 := TIdPOP3.Create(nil);
  try
    POP3.Host     := 'pop.racoon-city.com';
    POP3.Port     := 110;
    POP3.Username := 'racooncity'; // UserName
    POP3.Password := 'keel';       // Password
    POP3.Connect;
    Msg := TIdMessage.Create(nil);
    try
      MsgCnt := POP3.CheckMessages;
      for i:=1 to MsgCnt do
        begin
          POP3.RetrieveHeader(i, Msg);
          Msg.NoDecode :=
            (LowerCase(Msg.ContentTransferEncoding) <> 'base64') and
            (LowerCase(Msg.ContentTransferEncoding) <> 'quoted-printable');
          POP3.Retrieve(i, Msg);
          Memo1.Lines.Add('Subject: ' + Msg.Subject);
          Memo1.Lines.Add('Body:');
          Memo1.Lines.Add(Msg.Body.Text);
        end;
    finally
      Msg.Free;
    end;
    POP3.Disconnect;
  finally
    POP3.Free;
  end;
end;
```

ISO-2022-JP / UTF-8 共通です。

BASE64 化、または
Quoted-Printable 化
されたメールを受け取る場合には、
何の問題もありません。

7 bit / 8bit の Raw データが
送信されて来た場合には、
TIdMessage.NoDecode を
明示的に False に設定します。

日本語メールの場合、
Tranfer Encoding は、
Quoted-Printable よりも BASE64
の方が効率がいいです。

Indy – 雑多な情報

- CharSet 名からコードページを得るには？
  - IdCharSets.CharsetToCodepage() を使う。
  - MecsUtils.MecsCharsetToCodepage() も使える。
- コードページから CharSet 名を得るには？
  - 何故か CharsetToCodepage の逆を行う関数は Indy に存在しない (ハズ)。
  - MecsUtils.MecsCodepageToCharset () を使う。
- Indy は頻繁に更新されるので、
  過去の解決策よりもスマートな解決策が見つかる事がある。
- 最新の IndyTiburon.zip は Delphi 5 ～ Delphi XE すべてに
  インストール可能。

embarcadero

17th Embarcadero Developer Camp

Database

BDE はダメなのか？

- Unicode 版 Delphi でも使えない事はない。
  - 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る
  - BDE を配布するのが面倒 (古いインストーラは UAC に対応していない)。
- 但し、BDE は Delphi 6 (2001 年発売) 付属の 5.2 が最新。

Database	Version
InterBase	6.0
MS SQL Server	7.0
Oracle	8.1.6
Sybase	11.0.3
Informix	7.2 /9.11
DB2	2.11 / 5.0

- 実質、ネイティブドライバ以外には使えない。
  - このうち、Access / ASCII は ADO で置換可能。
  - dBASE / Paradox についても、ODBC 経由の ADO で接続可能なので、今更 BDE を使う理由はない。

BDE からの移行先は？

- InterBase
  IBX または DBX4
- MS SQL server
  dbGo (ADO Express) または DBX4
- Oracle / Sybase / Informix / DB2
  DBX4
- MS Access / ASCII
  dbGo (ADO Express)
- Paradox / dBASE / MS FoxPro
  ODBC 経由で dbGo (ADO Express)
  または、BlackfishSQL や InterBase (To-Go)、Firebird (Embedded) へ移行。

BDE からの移行手順は？

- Borland Database EngineアプリケーションのdbExpressへの移行
  http://edn.embarcadero.com/jp/article/33547
- ADOを利用したデータベースアプリケーション開発 (C++Builder)
  http://edn.embarcadero.com/jp/article/38241
- dBASEからInterBaseへの移行
  http://edn.embarcadero.com/jp/article/36558
- ParadoxからInterBaseへの移行
  http://edn.embarcadero.com/jp/article/36548
- デベロッパーキャンプ資料
  http://conferences.embarcadero.com/jp
  http://ht-deko.minim.ne.jp/tech044.html

4バイト UTF-8 に対応した DB を使う

- DB の制限を調べる
  - InterBase
    UNICODE-FSS では 3 バイトの UTF-8 にしか対応していない。
  - Firebird
    2.0 以前の UNICODE-FSS では 3 バイトの UTF-8 にしか対応していない
  - MySQL
    現状では 3 バイトの UTF-8 にしか対応していない。
  - MS SQL Server 7.0 / 2000
    サロゲートペアを正しく照合しない。
  - Oracle
    AL32UTF8 は 4 バイト、UTF8 は 3 バイトの UTF-8 。
- DB 格納前に正規化を行う
  - 文字の見た目は同じでも、抽出条件に引っ掛からない場合がある。

3バイト UTF-8 の DB ではダメなのか？

- 用途によっては 3 バイト UTF-8 の DB でもどうにかなる。
  - 4 バイト文字 (= サロゲートペア) が通らないのだから、
    サロゲートペアを何らかの形でエスケープして DB にストアしてやればいい。
  - 一番簡単なのは、サロゲートペアを HTML の
    数値文字参照 (&#xXXXX) 形式で扱う方法。
  - だがしかし…例によって、HTTPApp.EncodeHTML() / DecodeHTML() は、
    問題を抱えている (QC#78903 / QC#87435)。

```
uses
  ..., MecsUtils;

  // HTML エンコードされた文字列をパラメータにセット
  ParamByName('BOW').AsString := MecsHTMLEncode(Dmy);

  // HTML エンコードされた文字列をデコードして取り出す
  Dmy := MecsHTMLDecode(FieldByName('BOW').AsString);
```

  - 但し、読み書きの際のオーバーヘッドや、Order By による並び順が
    OS や DB が定めているロケール順にならない事があるのに注意。
  - Varchar 等のサイズを大きめに取る必要がある。

embarcadero

17th Embarcadero Developer Camp

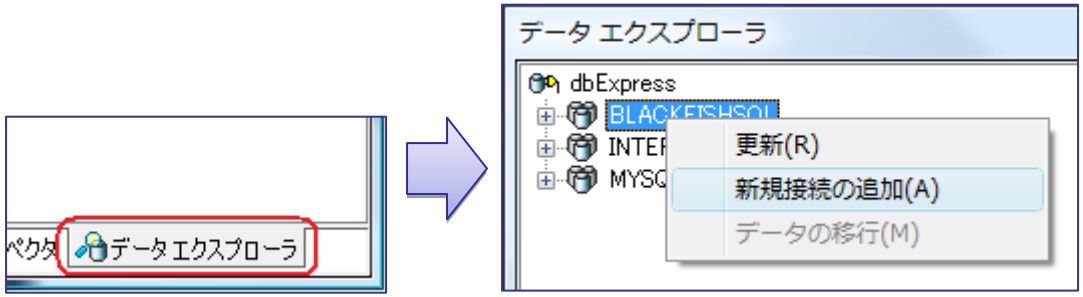
Blackfish™ SQL with DBX4 (BDE 代替)

Blackfish SQL with DBX4

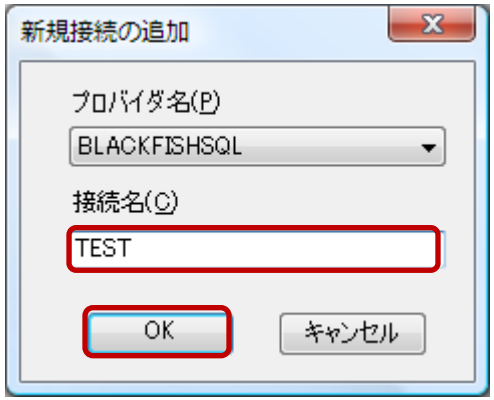
- BDE (Paradox / dBASE) の代替としてローカル接続を行う例。
- Unicode 版 Delphi には標準添付 (Delphi 2007 R2 ～ 2010)。
- ローカル接続なら Pro 版でも大丈夫。
- Blackfish SQL の動作環境には .NET 2.0 が必要。
- ライセンスファイルの取得と配置
  ライセンスファイルの取得と配置方法はインストールフォルダの
  “deploy.htm” / “deploy_ja.htm” に記述がある。

データベースの作成

1. “データエクスプローラ” から “BLACKFISHSQL” を右クリックして “新規接続の追加” を選択

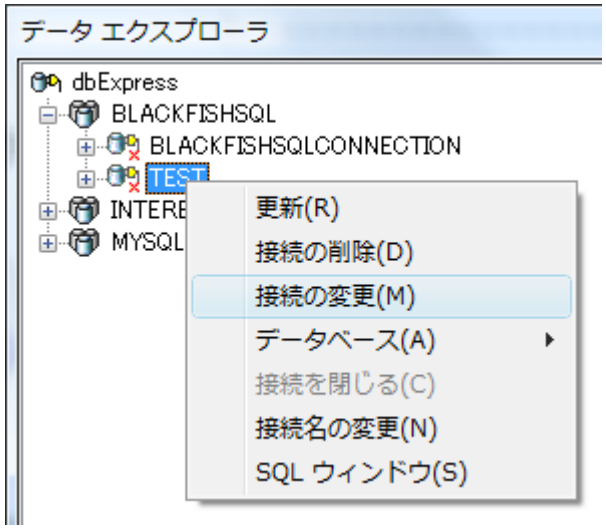


2. 適当な接続名を入力して “OK” ボタンを押下。



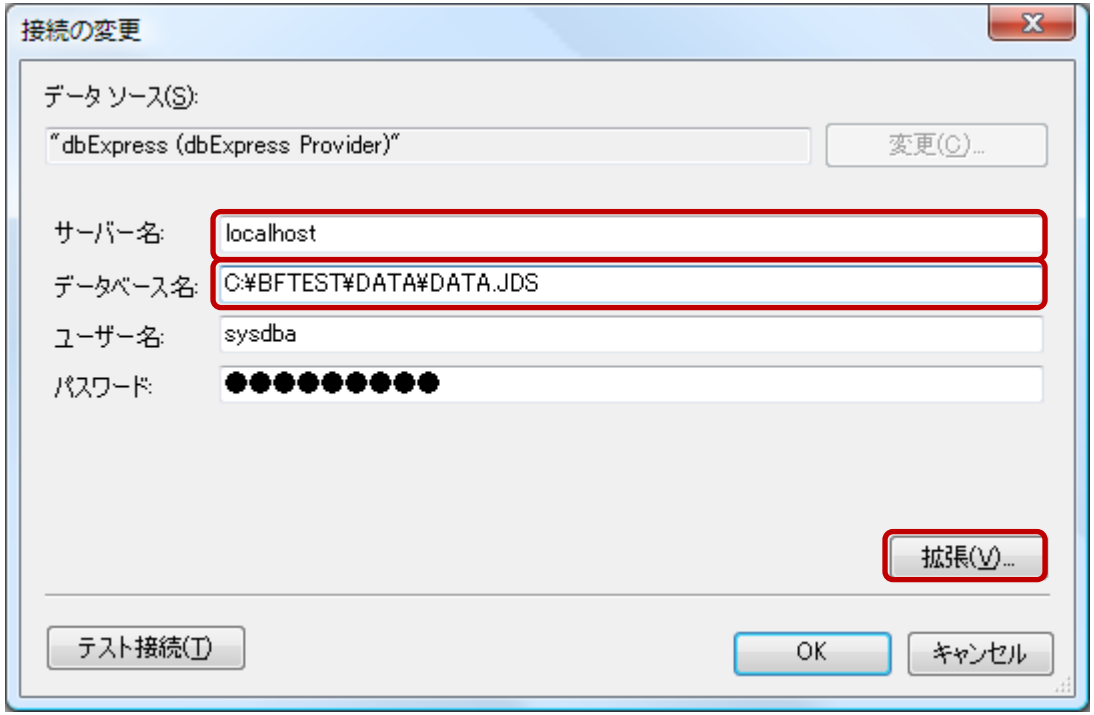
データベースの作成

3. “BLACKFISHSQL” にある “(接続名)” を右クリックし、”接続の変更”。



データベースの作成

4. “サーバ名” に “localhost”、
”データベース名” には作成するデータベース名をフルパスで入力。

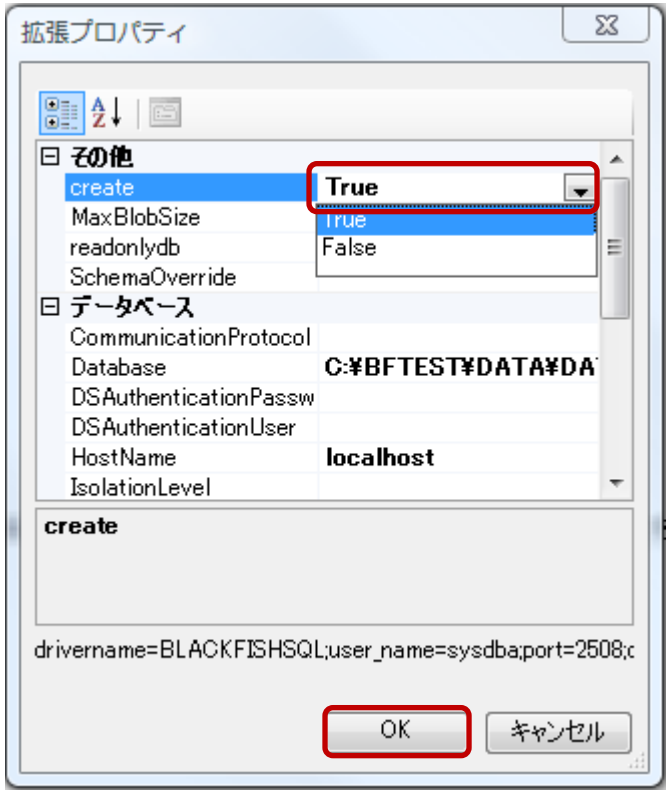


例では “C:¥BFTEST¥DATA¥DATA.JDS” を作成している。

5. “拡張” ボタンを押下。

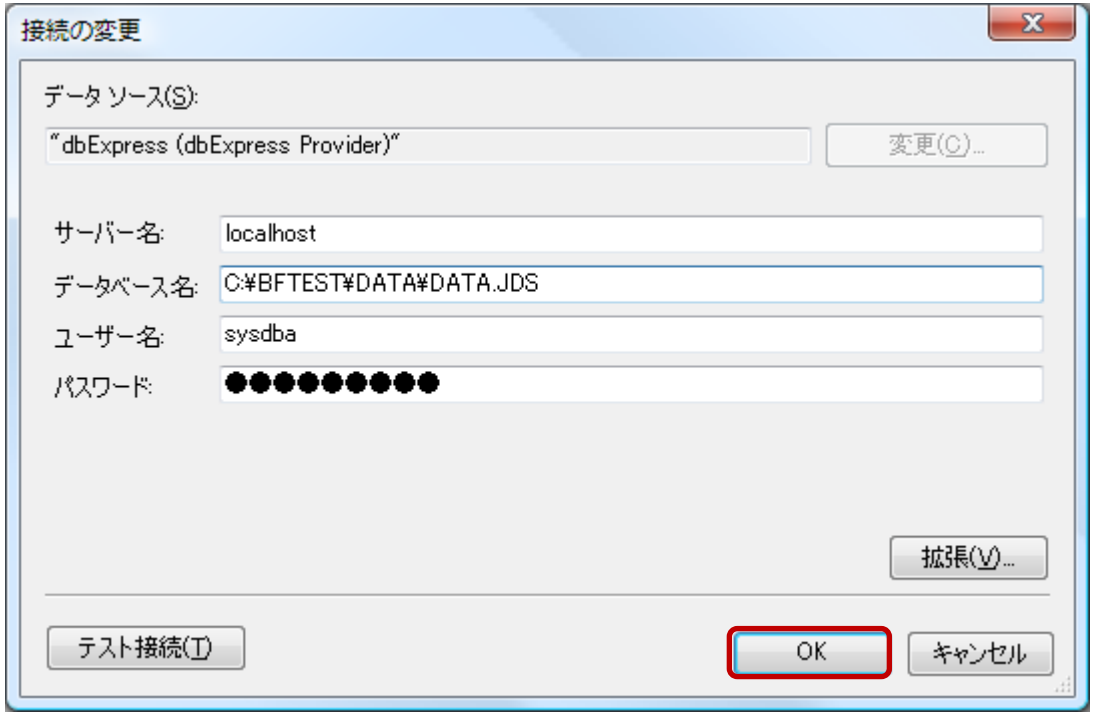
データベースの作成

6. [その他] – **[create] を True** に設定し、“OK” ボタンを押下。



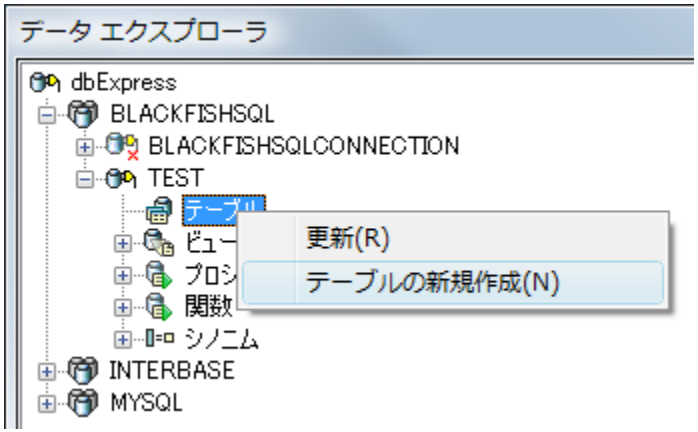
データベースの作成

7. “接続の変更” ダイアログの “OK” ボタンを押下。



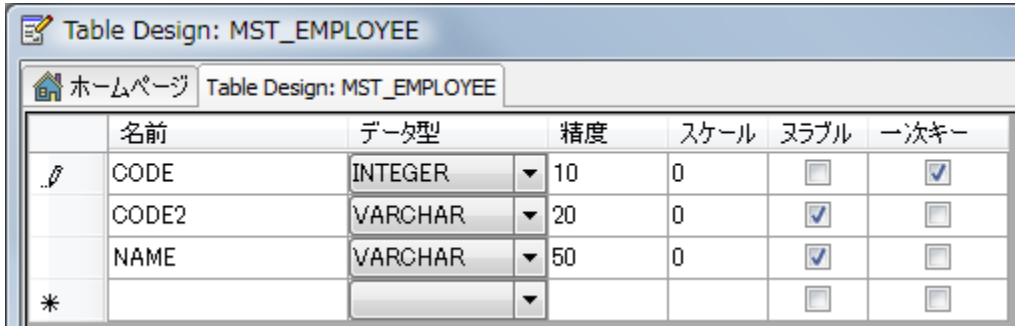
テーブルの作成

1. “データエクスプローラ” から 接続名 “TEST” を選択、
   ”テーブル” を右クリックして “テーブルの新規作成” を選択”



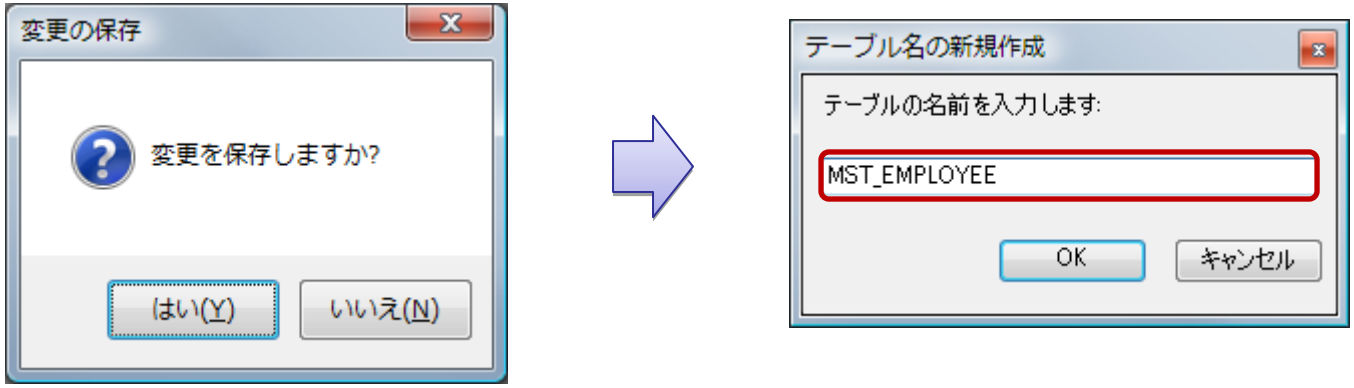
テーブルの作成

2. テーブル設計でテーブルを作成。




```
CREATE TABLE MST_EMPLOYEE
(
  CODE    INTEGER NOT NULL,
  CODE2   VARCHAR(20),
  NAME    VARCHAR(50),
  PRIMARY KEY (CODE)
)
```

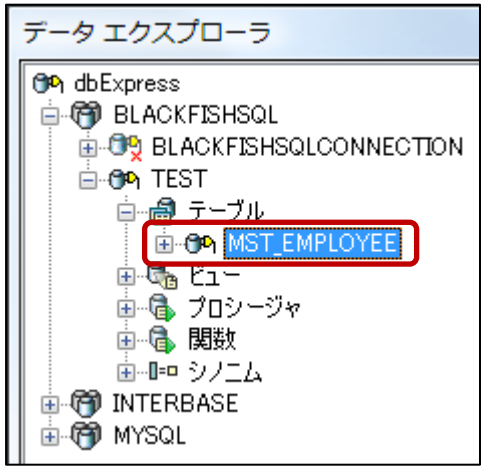
3. タブを閉じようとすると変更を保存するか聞いて来るので “はい” を押下。



4. 続けてテーブル名を尋ねてくるので、“MST_EMPLOYEE” と入力して “OK” を押下。

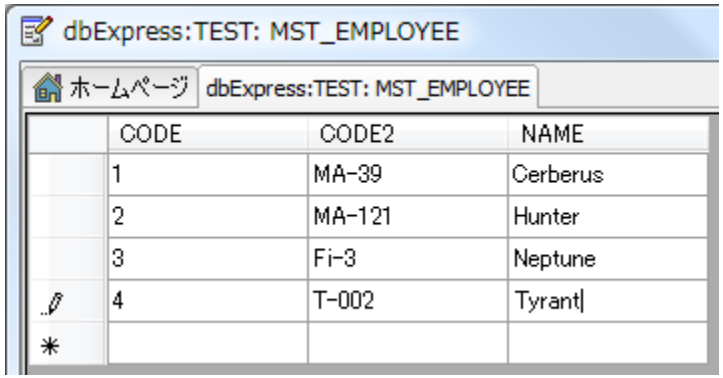
データの閲覧と入力

1. “BLACKFISHSQL” - “TEST” - “テーブル” と辿っていき、
   “MST_EMPLOYEE” をダブルクリックします。

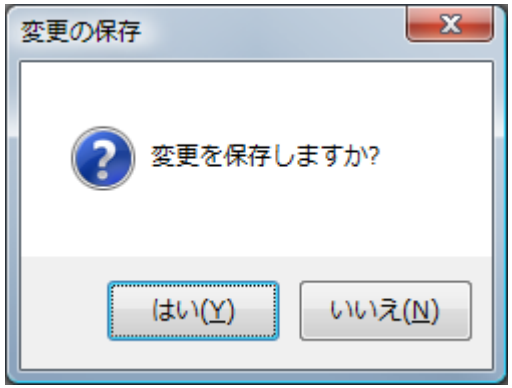


データの閲覧と入力

2. タブでデータグリッドが表示されます。



3. タブを閉じようとすると変更を保存するか聞いて来るので “はい” を押下。



ソースコード

```
uses
  ..., DbxBlackfishSQL; // midas.dll を配布するのが面倒なら、MidasLib を uses に追加

procedure TForm1.Button1Click(Sender: TObject);
var
  CTX: TDBXContext;
  PROP: TDBXBlackfishSQLProperties;
  DBXC: TSQLConnection;
  QRY: TSQLQuery;
  Dmy: String;
begin
  DBXC := TSQLConnection.Create(Self);
  QRY := TSQLQuery.Create(Self);
  try
    DBXC.LoginPrompt := False;
    DBXC.DriverName  := 'BLACKFISHSQL';
    CTX := TDBXContext.Create;
    PROP := TDBXBlackfishSQLProperties.Create(CTX);
    try
      DBXC.Params.Clear;
      DBXC.Params.Add(Format('DriverUnit=%s', [PROP.Values['DriverUnit']]));
      Dmy := StringReplace(PROP.Values['MetaDataPackageLoader'],
                           'TDBXDataStoreMetaDataCommandFactory',
                           'TDBXClientDriverLoader',
                           [rfIgnoreCase]);
      DBXC.Params.Add(Format('DriverPackageLoader=%s', [Dmy]));
      DBXC.Params.Add(Format('MetaDataPackageLoader=%s', [PROP.Values['MetaDataPackageLoader']]));
      DBXC.Params.Add(Format('User_Name=%s',[PROP.UserName])); // User Name
//    DBXC.Params.Add(Format('User_Name=%s',['sysdba']));
      DBXC.Params.Add(Format('Password=%s' ,[PROP.PassWord])); // Password
//    DBXC.Params.Add(Format('password=%s' ,['masterkey']));
```

プロジェクトは C:¥BFTEST に保存します。

ソースコード

```
      Dmy := IncludeTrailingPathDelimiter(ExtractFilePath(ParamStr(0))) +
             'DATA¥DATA.JDS';
      DBXC.Params.Add(Format('DataBase=%s' ,[Dmy]));          // Database
    finally
      PROP.Free;
      CTX.Free;
    end;
    QRY.SQLConnection := DBXC;
    DBXC.Connected := True;
    QRY.SQL.Clear;
    QRY.SQL.Add('Select * From MST_EMPLOYEE');
    QRY.Open;
    Memo1.Lines.Clear;
    while not QRY.EOF do
      begin
        Dmy := QRY.FieldByName('NAME').AsString;
        Memo1.Lines.Add(Dmy);
        QRY.Next;
      end;
    QRY.Close;
    DBXC.Connected := False;
  finally
    QRY.Free;
    DBXC.Free;
  end;
end;
```

特に難しい所はありません。Blackfish SQL で難しいのは動作させるまでと配布時です。
パラメータの設定部分は無理矢理ですが、こうしておくとパッケージのバージョンに依存しない
ソースコードにする事ができます。

配布

- まずは以下のファイルを集める。

```
$(BDS)¥bin
  - BSQLServer.exe
  - BSQLServer.exe.config
  - RAD Studio BlackfishSQL.slip

%ProgramFiles%¥Common Files¥CodeGear Shared¥RAD Studio¥Shared Assemblies¥7.0
  - Borland.Data.BlackfishSQL.LocalClient.dll

%ProgramFiles%¥Common Files¥CodeGear Shared¥RAD Studio¥Shared Assemblies¥7.0¥ja
  - Borland.Data.BlackfishSQL.LocalClient.resources.dll

%SystemRoot%¥System32
  - midas.dll (必要であれば)
```

<!> “7.0” は Delphi 2010 の場合。2009 なら “6.0”、XE なら “8.0” <!>
<!> TClientDataset を利用すると midas.dll が必要になる <!>
<!> サンプルソースコードの範囲では midas.dll は不要 <!>
<!> uses に MidasLib を含めれば midas.dll の配布は不要 <!>

配布

- 以下の様に配置し、配布する。

```
¥APP
|    MyApp.exe（自作アプリケーション）
|    Borland.Data.BlackfishSQL.LocalClient.dll
|    BSQLServer.exe
|    BSQLServer.exe.config
|    RAD Studio BlackfishSQL.slip
|
+---DATA
|        DATA.JDS                （データベース）
|        DATA_LOGA_0000000000 （データベースログファイル）
|        DATA_LOGA_ANCHOR       （不要かも）
|
+---ja
         Borland.Data.BlackfishSQL.LocalClient.resources.dll
         （エラーメッセージ等を日本語で表示したい時）
```

– “BSQLServer.exe -install” を実行すると、Blackfish SQL をサービスとして登録できる。インストーラに指定しておくとよい。
– サービスはデフォルトで自動起動だが、インストール直後は停止状態なので、“net start BlackfishSQL” を実行するか、OS を再起動しなくてはならない。
– サービスの登録解除は “BSQLServer.exe -remove” 。

トラブルシューティング

- データベースエクスプローラに “BLACKFISHSQL” がない。
- データベースを作ろうとするとエラーになる。

“<コモンドキュメントフォルダ>¥ RAD Studio¥dbExpress” 以下にある、
・dbxconnections.ini
・dbxdrivers.ini
が、おかしくなっている可能性がある。

関連して、複数のバージョンの Delphi をインストールしている環境では、
“System.InvalidCastException: 型 ‘Borland.Data.TDBXDynalinkDriverLoader‘ のオブジェクトを型 ’Borland.Data.TDBXDriverLoader’ にキャストできません。”
のようなエラーが出ることがある。

Delphi 2010 の場合、
[HKEY_CURRENT_USER¥Software¥CodeGear¥BDS¥7.0¥DBExpress]
に存在するレジストリエントリのパスに “¥7.0” が含まれていない事がある。

“<コモンドキュメントフォルダ>
2000 / XP: “¥Documents and Settings¥All Users¥Documents”
Vista / 7: “¥Users¥Public¥Documents”

トラブルシューティング

- もう少し詳しく説明。

“DBX の設定ファイルの場所” はレジストリに書かれており、
DBX の設定ファイルは以下の場所にある。

バージョン	DBX 環境設定ファイルの場所
2006 以前	$(BDS)¥dbExpress
2007	<コモンドキュメントフォルダ>¥RAD Studio¥dbExpress
2009	<コモンドキュメントフォルダ>¥RAD Studio¥dbExpress
2010	<コモンドキュメントフォルダ>¥RAD Studio¥dbExpress¥7.0
XE	<コモンドキュメントフォルダ>¥RAD Studio¥dbExpress¥8.0

見ての通り、デフォルトの状態では 2007 / 2009 の DBX は共存できない。
(設定ファイルの場所を別にしてレジストリを書き換えればおそらく共存可能)

2010 は共存できるハズなのだが、
何故かレジストリが 2007 / 2009 と同じになっている事がある (詳細不明)。

embarcadero

17th Embarcadero Developer Camp

Firebird SQL with IBX

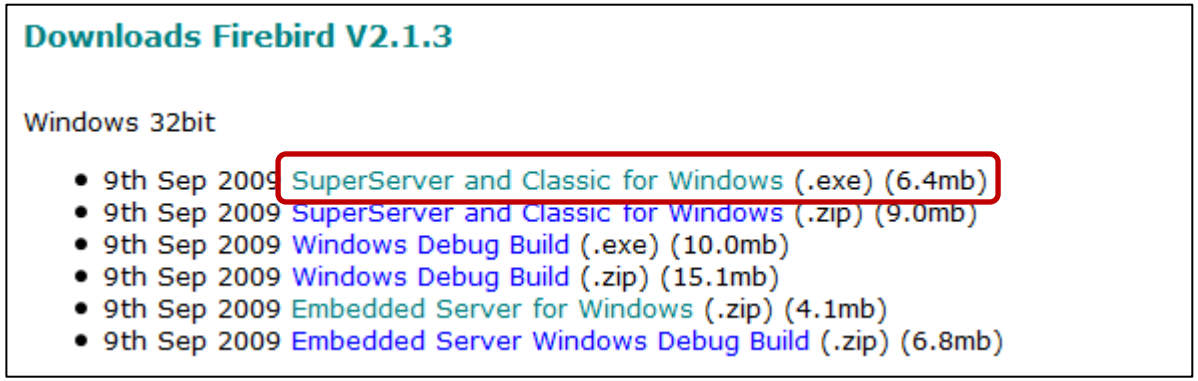
(BDE 代替)

Firebird with IBX

- BDE (Paradox / dBASE) の代替としてローカル接続を行う例。
- C/S 接続、Embedded への対応もほぼ同じコードで可能。
- IBX で Firebird を使う事は正式サポートされていない。
- Ent 版 / Arc 版をお持ちの方は DBX4 での接続を推奨。
  (Delphi 2010 以降で正式にサポートされている。)
- サポートが必要な場合は InterBase® の利用を。

ダウンロードとインストール

- Firebird の ダウンロード
  http://www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp_download_21



“Super Server and Classic for Windows (.exe)” をダウンロード。
ついでに “Embedded Server for Windows (.zip)” もダウンロードしておくといい。

- Firebird のインストール
  インストーラを起動し、ひたすら “Next” を押していけばいい。
  勝手にサービス起動するように設定される。

管理ツールのダウンロードとインストール

- IBConsole 日本語版+α (管理ツール) の ダウンロード
  http://ht-deko.minim.ne.jp/junkbox.html#IBCONSOLE

Product:	・IBConsole for Windows (Unicode Edition) (2000/XP/Vista/7)
Original Version:	1.0.0.337
Version:	rel.13
DownLoad:	ibc_b337_win_ja_unicode_rel.13.zip
Update:	2010/03/04

必ず “Unicode Edition” をダウンロード。

- IBConsole 日本語版+α のインストール
  適当な場所に解凍してお使いください。

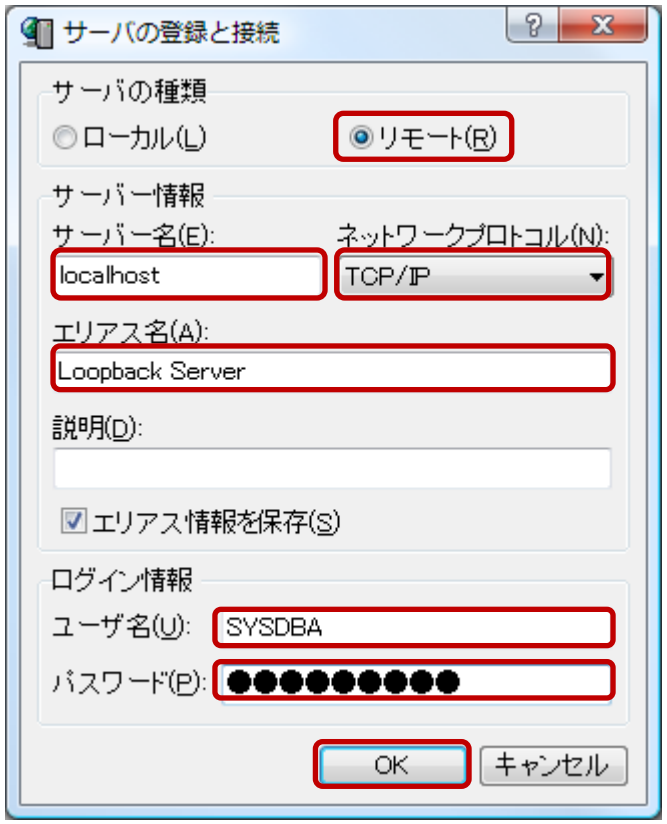
- (Firebird のインストール先)¥bin¥isql.exe という管理ツールもあるが、コマンドラインで操作する必要がある。

IBConsole 日本語版 + αの設定

1. IBConsole.exe を起動したら、
   ”InterBase サーバ” を右クリックして “サーバを登録”

IBConsole 日本語版 + αの設定

2. 各種パラメータを入力し、”OK” ボタンを押下。



サーバの種類:
“リモート”

サーバ名:
“localhost” (または 127.0.0.1)

ネットワークプロトコル:
“TCP/IP”

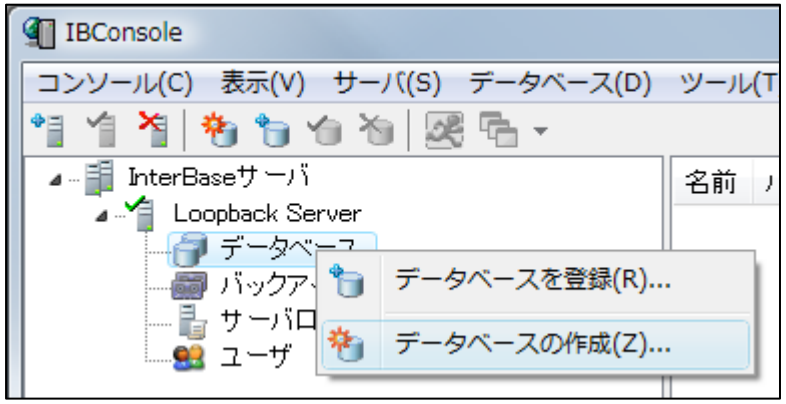
エリアス名:
任意 (ここでは “Loopback server”)

ユーザ名:
“SYSDBA”

パスワード:
“masterke” (“masterkey” でもいい)

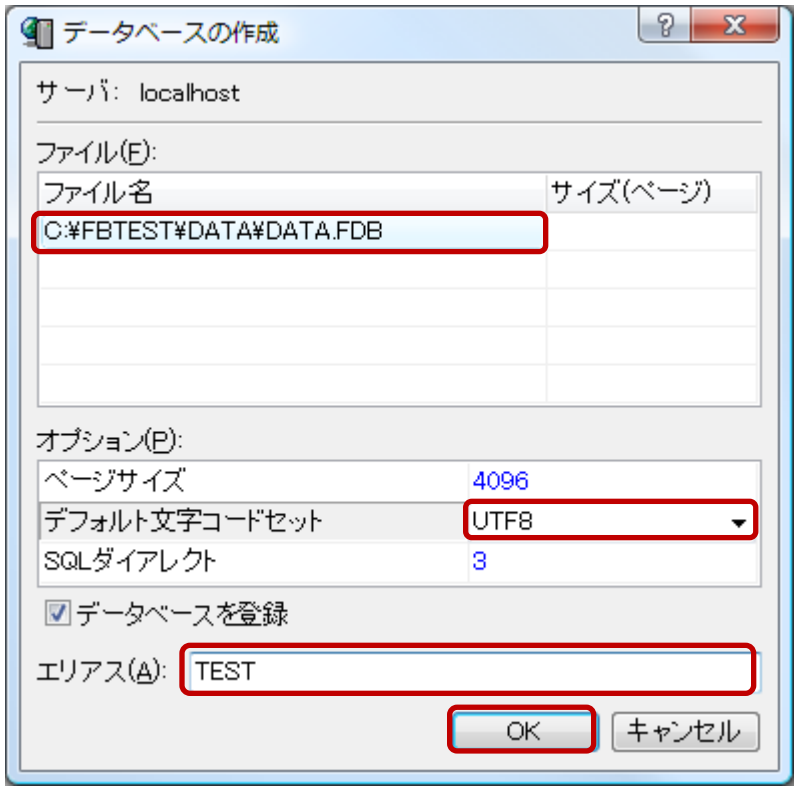
データベースの作成

1. “Loopback Server” の下にある “データベース” を右クリックして “データベースの作成”



データベースの作成

2. 各種パラメータを入力し、”OK” ボタンを押下。



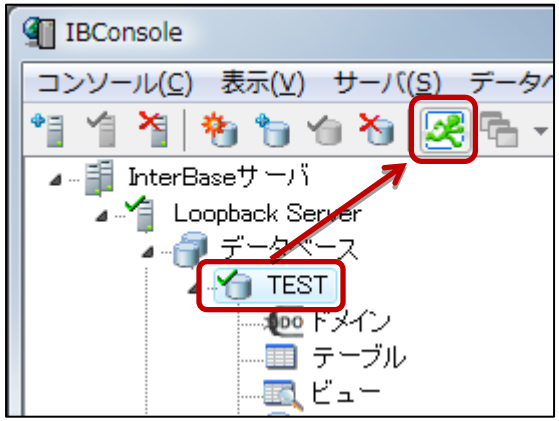
ファイル名:
任意
“C:¥FBTEST¥DATA¥DATA.FDB”

デフォルト文字コードセット:
“UTF8”
(Delphi 2009 では “UNICODE_FSS”)

エリアス:
任意
(ここでは “TEST”)

テーブルの作成

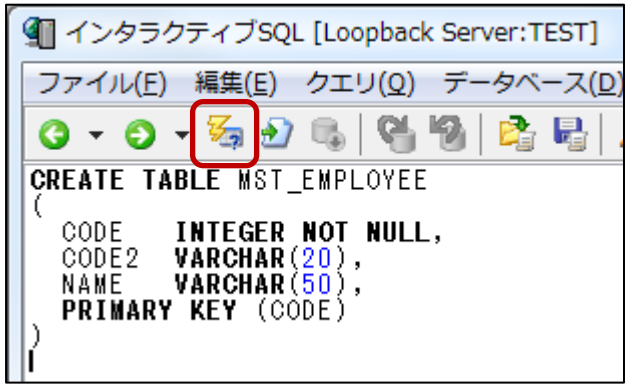
1. エリアス “TEST” を有効にした状態で
   ツールバーにある “インタラクティブ SQL” ボタンを押下。



※ または [ツール | インタラクティブ SQL] をクリック。

テーブルの作成

2. “インタラクティブ SQL” で、CREATE TABLE 文を入力。
   ツールバーにある “実行” ボタンを押下。



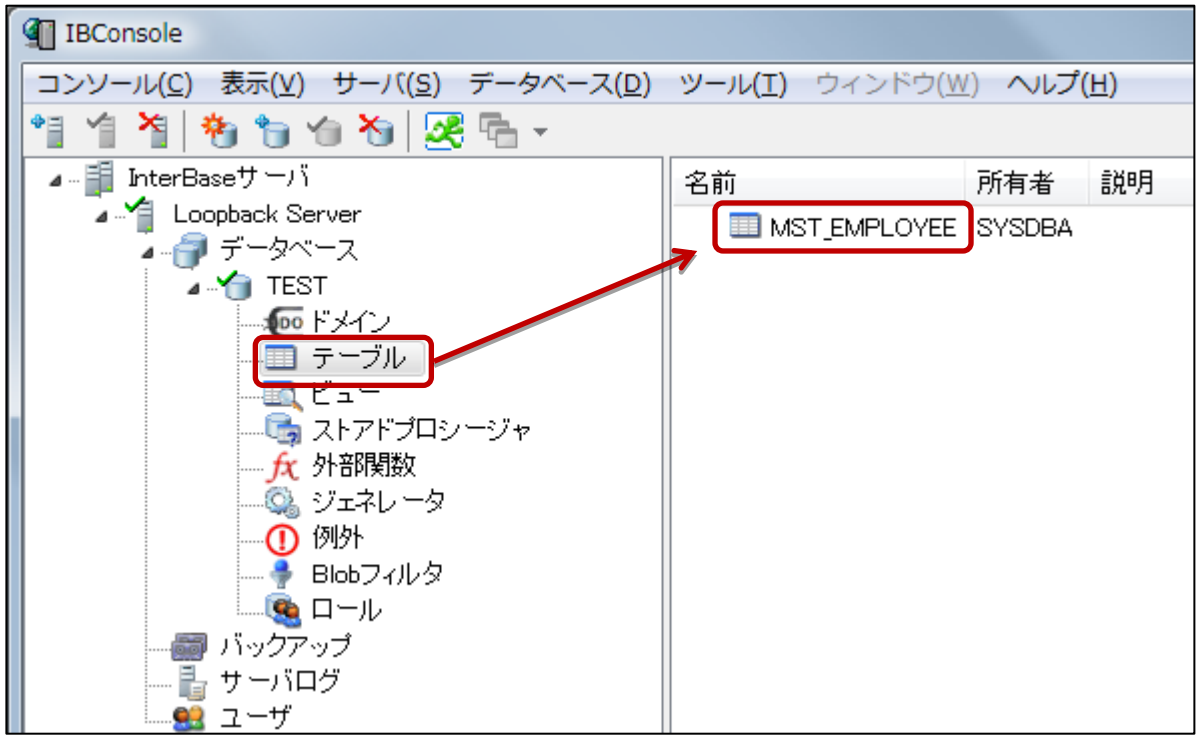

```
CREATE TABLE MST_EMPLOYEE
(
  CODE    INTEGER NOT NULL,
  CODE2   VARCHAR(20),
  NAME    VARCHAR(50),
  PRIMARY KEY (CODE)
)
```

※ または [クエリ | 実行] をクリック。

3. “インタラクティブ SQL” を閉じます。

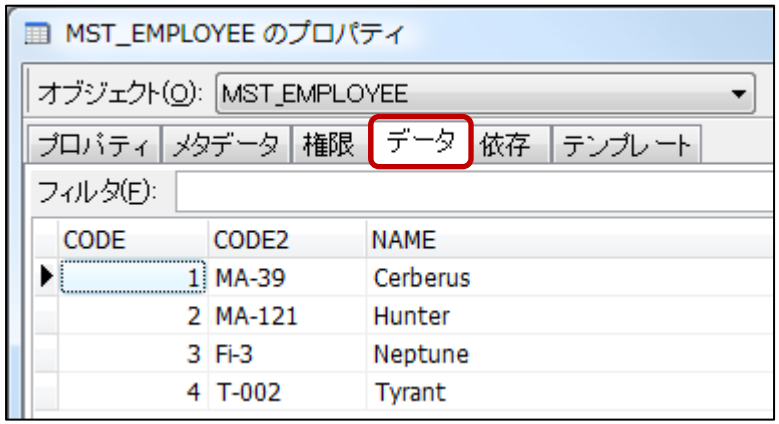
データの閲覧と入力

1. 右ペインで “InterBase サーバ” – “Loopback Server” – “データベース” – “TEST” – “テーブル” と辿っていき、左ペインの “MST_EMPLOYEE” をダブルクリック。

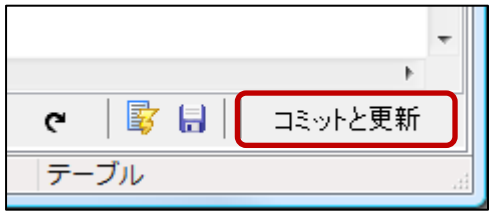


データの閲覧と入力

2. “データ” タブを選択するとデータグリッドが表示されます。



4. 入力が終わったら、右下の “コミットと更新” を押して入力内容を確定させます。



5. “(オブジェクト) のプロパティ” フォームを閉じます。

ソースコード


```
procedure TForm1.Button1Click(Sender: TObject);
var
  IBXC: TIBDatabase;
  Tran: TIBTransaction;
  QRY: TIBQuery;
  Dmy: String;
begin
  IBXC := TIBDatabase.Create(Self);
  Tran := TIBTransaction.Create(Self);
  QRY  := TIBQuery.Create(Self);
  try
    IBXC.LoginPrompt := False;
    IBXC.DefaultTransaction := Tran;
    IBXC.Params.Clear;
    IBXC.Params.Add(Format('user_name=%s',['SYSDBA']));    // UserName
    IBXC.Params.Add(Format('password=%s' ,['masterke']));  // Password
    IBXC.Params.Add(Format('lc_ctype=%s' ,['UTF8']));      // CharSet(2009 ならUNICODE_FSS)
    Dmy := 'localhost:' +                                  // Embedded の場合にはサーバ名不要
           IncludeTrailingPathDelimiter(ExtractFilePath(ParamStr(0))) +
           'DATA¥DATA.FDB';
    IBXC.DataBaseName := Dmy;
    QRY.Database := IBXC;
    IBXC.Connected := True;
    QRY.SQL.Clear;
    QRY.SQL.Add('Select * From MST_EMPLOYEE');
    QRY.Open;
    Memo1.Lines.Clear;
    while not QRY.EOF do
      begin
        Dmy := QRY.FieldByName('NAME').AsString;
        Memo1.Lines.Add(Dmy);
        QRY.Next;
      end;
```

プロジェクトは C:¥FBTEST に保存します。

ソースコード

```
    QRY.Close;
    IBXC.Connected := False;
  finally
    QRY.Free;
    Tran.Free;
    IBXC.Free;
  end;
end;
```

ローカル接続であっても、ループバック接続させるために、
DatabaseName プロパティには “localhost:” または “127.0.0.1:” を付加しておきます。
“lc_ctype” には、文字エンコーディングを指定します。

配布

- とっても簡単。
  - プロジェクトの生成物 (.exe)
  - Firebird のインストーラ

    これらを配布するだけ。
    アプリケーションにインストーラが必要なら、アプリケーションインストール実行後に、Firebird インストーラを起動するように設定する。

- Firebird のインストールさえユーザにやらせたくない場合
  - “Firebird Embedded Server” を使ってスタンドアロンで動作させる。
  - プロジェクトの生成物 (.exe) と共にファイルコピーするだけで動作する。

Firebird Embedded Server での配布

- とっても簡単。
  - Embedded Server の zip を解凍し、
    必要なファイルを以下のように配置するだけ。

```
¥APP
|    MyApp.exe（自作アプリケーション）
|    aliases.conf
|    GDS32.DLL（fbembed.dll をリネーム）
|    firebird.conf
|    firebird.msg
|    ib_util.dll
|    icudt30.dll
|    icuin30.dll
|    icuuc30.dll
|    Microsoft.VC80.CRT.manifest
|    msvcp80.dll
|    msvcr80.dll
|
+---DATA
|        DATA.FDB（データベース）
|
+---intl
|        fbintl.conf
|        fbintl.dll
|
+---udf
         fbudf.dll
         ib_udf.dll
```

配布は単純にフォルダコピーでいい事になります。

面倒な設定は一切不要です。

データベースをリードオンリーに設定すれば、CD / DVD から起動するアプリケーションに DB が使える事になります。

もちろん、DB をリードオンリーにすればデータの追加や更新はできませんが、カタログ等の検索用途には使えます。

トラブルシューティング

- DB に接続できない (通常の Firebird)。
  - 単純に、Firebird がサービスとして起動していない。
  - パスの通った場所に別の GDS32.DLL が存在する。
    (Firebird のインストール先)¥bin¥fbclient.dll を GDS32.DLL にリネームし、アプリケーションと同じ場所に置いてみる。
  - データベース名にサーバ名を含めていない (“localhost:” または “127.0.0.1:”)
- Delphi 2009 だと “UTF8” で接続できない。
  - Delphi 2009 では ”UNICODE_FSS” を使う。UTF-8 (4 バイト文字) は通る。
- DB に接続できない (Embedded Server)。
  - 配布用ファイルが足りない。
  - Fbembed.dll を GDS32.DLL にリネームしていない。
  - データベース名にサーバ名を含めてしまっている。
- IBConsole で DB に接続できない (Embedded Server)。
  - Embedded Server には、サーバ管理の機能が備わっていない。

embarcadero

17th Embarcadero Developer Camp

ファイル名

日本語ファイル名 – その1

- Windows のファイル名
  - NT 系の Windows ではファイル名に Unicode が使える。
  - 8.3 形式のファイル名には Unicode は使えない。
  - 日本語ファイル名は基本的に NFC で正規化されている。
- Mac OSX のファイル名
  - 日本語ファイル名は基本的に NFD で正規化されている (結合文字列)。
- Linux のファイル名
  - 日本語ファイル名は基本的に NFC で正規化されている。
- 各種アーカイバのファイル名
  - 仕様上は Unicode ファイル名に対応しているが、
    機能がオミットされているものがある。
  - アーカイバによっては、内部を ANSI で処理しているものがある

日本語ファイル名 – その2

- ISO9660 のファイル名 (CD / DVD)
  - Joliet　形式では、UCS-2 をサポートする。サロゲートペアは考慮されない。
  - おそらく結合文字列もサポートされない。
    これは、Mac OSX から持ってきたファイルを Windows で焼こうとすると
    エラーになる事を意味する (濁音、半濁音を含む文字等)。

- UDF のファイル名 (Blu-ray 他)
  - ファイル名は Unicode をサポートする。
  - 但し、UDF Bridge を使っている場合には、ISO 9660 の影響を受ける。

- 無用なトラブルを避けるため、どこからか貰ってきたファイルは、
  ファイル名を事前に NFC で正規化してリネームすべき。
  (日本語を含むファイル名の場合)

日本語ファイル名 – 正規化

- ファイル名を正規化する関数

```
uses
 ..., MecsUtils;

procedure FilenameNormalize(aFileName: TFileName; NType: TNORM_FORM);
var
  Src: String;
  Dst: String;
begin
  Src := ExtractFileName(aFileName);
  if MecsNormalize(Src, Dst, NType) then
    if Src <> Dst then
      RenameFile(aFileName,
        IncludeTrailingPathDelimiter(ExtractFilePath(aFileName)) + Dst);
end;
```

```
// ファイル名を NFC で正規化してリネーム
FilenameNormalize('C:¥STARS¥ケンド銃砲店からのFAX.TXT', NormalizationC);
```

FindFirst() / FindNext() と組み合わせて使う場合には、
“フォルダ名が NFD になっていないか” も考慮すること。

日本語ファイル名 – UCS2 ファイル名

- ファイル名が UCS2 かを調べる関数

```
function IsUCS2String(s: String): Boolean;
var
 i: Integer;
begin
  result := True;
  for i:=1 to Length(s) do
    if SysUtils.ByteType(s, i) <> mbSingleByte then
      begin
        result := False;
        Break;
      end;
end;
```

  - “UCS2 = U+0000 ～ U+FFFF”
  - “U+FFFF よりも大きいコードポイント = サロゲートペアで表される”
    …なので、”文字列中に High-Surrogate または Low-Surrogate が含まれていたら UCS2 ではない” という事になる。

日本語ファイル名 – ANSI ファイル名

- Indy (FTP) の項で触れたように、相手によってはファイル名を ANSI にしなければならない事がある。
- だが、Unicode → ANSI 変換で変換できない文字は欠損するのではなく、”?” に置換されてしまう。

飼育係の日誌 9–21 May 1998.txt

飼育係の日誌 9?21 May 1998.txt

- “?” はファイル名には使えないので、そこでエラーになってしまう。
  (注: “–” は U+2013: ENDASH。海外では日付の範囲を表すのに使われる)

日本語ファイル名 – ファイル名チェック

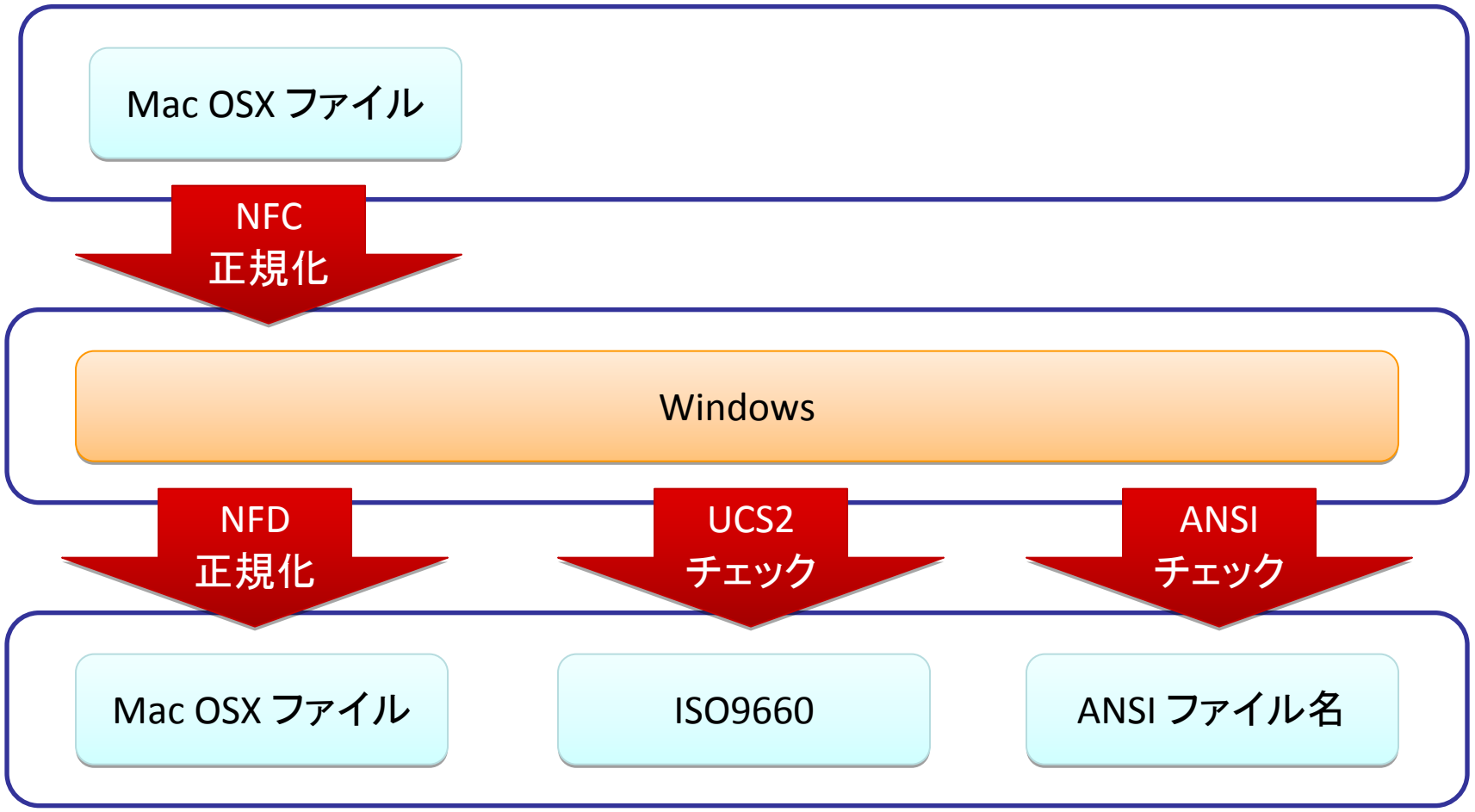

```
uses
  ..., StrUtils, shlwapi;

function IsFileName(S: TFileName): Integer;
const
  RSV : array [0..24] of String = ('CON', 'PRN', 'AUX', 'CLOCK$', 'NUL',
   'COM0', 'COM1', 'COM2', 'COM3', 'COM4', 'COM5', 'COM6', 'COM7', 'COM8', 'COM9',
   'LPT0', 'LPT1', 'LPT2', 'LPT3', 'LPT4', 'LPT5', 'LPT6', 'LPT7', 'LPT8', 'LPT9');
var
  i, ret: Integer;
begin
  result := 0;
  if Trim(S) = '' then
    begin
      result := -2;
      Exit;
    end;
  for i:=1 to Length(S) do
    begin
      ret := PathGetCharType(S[i]);
      if (ret and (GCT_INVALID or GCT_SEPARATOR or GCT_WILD)) > 0 then
        begin
          result := i;
          Break;
        end;
    end;
  if result = 0 then
    if StrUtils.AnsiIndexText(ChangeFileExt(S, ''), RSV) >= 0 then
      result := -1;
end;
```

引数 S に渡された文字列が
ファイル名として有効であれば
戻り値は 0 となり、無効であれば、
不正な文字の位置を返します。

戻り値が負数の場合には、
以下の理由で無効なファイル名です。
-1 = 予約語が使用されている
-2 = 文字列が空

ファイル名が ANSI でなければならない
場合、変換後のファイル名をこの関数
でチェックするとよいでしょう。

日本語ファイル名 – まとめ



embarcadero

17th Embarcadero Developer Camp

まとめ

今回のまとめ。

- 面倒臭がらずにステップアップグレードしましょう。
- フォームファイル (*.dfm) はテキスト形式に変換しましょう。
- マイグレーションには仮想化ソフトウェアを活用しましょう。
- レポートツールは吟味しましょう。
- Indy は最新版を使いましょう。
- IP*Works! の事は後々考えましょう。
- BDE は流石にもうやめときましょう。
- DB を使うなら Delphi XE はEnterprise 版以上を購入しましょう。
- ファイル名には気を付けましょう。
- 文字コード関連の QC には Vote しましょう。

embarcadero

17th Embarcadero Developer Camp

Q & A

embarcadero

17th Embarcadero Developer Camp

Extras

参考資料

- Embarcadero Technologies
  - [Delphi、C++Builder、RAD Studio 2010 - バージョンアップ情報]
    http://www.embarcadero.com/jp/rad-in-action/migration-upgrade-center
  - 第11回デベロッパーキャンプ 【A6】Delphi テクニカルセッション
    [Delphi 2009 ではじめる Unicode アプリケーション] 資料
    http://conferences.embarcadero.com/jp/article/images/39112/a6.pdf
  - デベロッパーキャンプアンコールセッション
    [Delphi での文字コードのハンドリングについて] 資料
    http://conferences.embarcadero.com/article/images/40483/devcamp_encore_20100415_Full.pdf
  - MECSUtils
    http://cc.embarcadero.com/item/26061
    http://ht-deko.minim.ne.jp/tech021.html

参考資料

- 無料で使える DB (ライセンスはしっかりと確認を)
  - Oracle Database 10g Express Edition
    http://www.oracle.com/technology/products/database/xe/index.html
  - Microsoft SQL Server 2005 / 2008 Express Edition
    http://www.microsoft.com/japan/sqlserver/2005/editions/express/default.mspx
  - http://www.microsoft.com/downloads/details.aspx?familyid=7522A683-4CB2-454E-B908-E805E9BD4E28&displaylang=ja
  - IBM DB2 Express-C
    http://www-06.ibm.com/software/jp/data/db2/v9/express-c/
  - Sybase Adaptive Server Enterprise (ASE) Express Edition (Linux)
    http://response.sybase.com/forms/ASE_Linux_Download
  - Firebird / Firebird Embedded Server
    http://www.firebirdsql.org/

参考資料

- 無料で使える DB (ライセンスはしっかりと確認を)
  - MySQL / MySQL Embedded Server
    http://www-jp.mysql.com/
    http://www-jp.mysql.com/oem/
  - PostgreSQL
    http://www.postgresql.jp/

    (以下、開発者向けに無料)
  - Sybase SQL Anywhere Developer Edition
    http://www.sybase.jp/detail?id=1016644
  - IBM Informix Dynamic Server Developer Edition
    http://www-06.ibm.com/software/jp/data/informix/ids/idsde/
  - Embarcadero InterBase SMP 2009 Developer Edition
    http://edn.embarcadero.com/article/39094