【A2】Delphi / C++Builderテクニカルセッション

「VCLから Excelにアクセス」

Excelクライアントアプリケーションの作成方法

CodeGear
エヴァンジェリスト
高橋智宏

Where Developers Matter
Integrated Development Environments for Windows, Java, and Web Developers
DEVELOPER CAMP

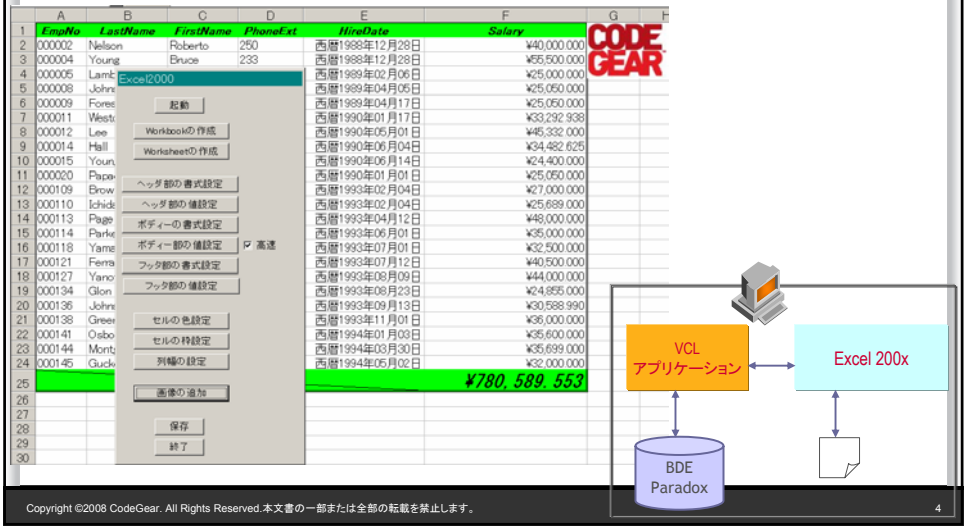
アジェンダ

- 今回作成するアプリケーションの概要
- Officeコンポーネントを使ってみよう
- セルへのアクセスの高速化
- Excel 2007 への対応方法

今回作成するアプリケーションの概要

Delphi6/C++Builder6 ～ Delphi2007/C++Builder2007
Excel 2000 ～ Excel 2007

Excelファイルを作成する



用意するもの

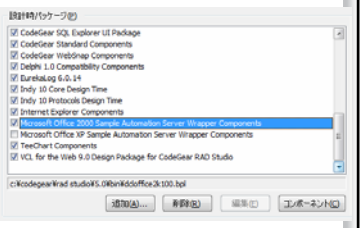
- Delphi 6 ～ 2007
- C++Builder 6 ～ C++Builder2007
  - 今回はDelphi言語版を作成します。C++Builderでは同様のコードを記述します
- Excel 2000 ～ Excel 2007
  - Office 2000 用のサーバーコンポーネントを利用します

CODE GEAR FROM Borland
Where Developers Matter
Integrated Development Environments for Windows, Java, and Web Developers
DEVELOPER CAMP

Officeコンポーネントを使ってみよう

Officeコンポーネントの確認
実装コード

Officeコンポーネントの確認



- プロジェクトにパッケージを追加
  - Excel 2000用を使う
    - 2000, XP, 2003, 2007 用に利用可能
  - TExcelApplication – Excelに接続する
  - TExcelWorkbook – ワークブックに接続する
  - TExcelWorksheet – ワークシートに接続する

Excelの起動からワークシートへの接続まで

```
procedure TForm1.ConnectExcelClick(Sender: TObject);
begin
  // エクセルに接続
  ExcelApplication1.Connect;
  ExcelApplication1.Visible[0] := True;
end;

procedure TForm1.CreateWorkbookClick(Sender: TObject);
var
  wb: ExcelWorkbook;
begin
  // ワークブックを作成して接続
  wb := ExcelApplication1.Workbooks.Add(EmptyParam, 0);
  ExcelWorkbook1.ConnectTo(wb);
end;

procedure TForm1.CreateWorksheetClick(Sender: TObject);
var
  ws: ExcelWorksheet;
begin
  // ワークシートを作成して接続
  ws := ExcelWorkbook1.Worksheets.Add(EmptyParam, EmptyParam, 1, xlWorksheet, 0) as ExcelWorksheet;
  ExcelWorksheet1.ConnectTo(ws);
end;
```

ユーティリティ関数

```
const ColTitle: array[1..26] of Char = ('A','B','C','D','E','F','G','H','I','J',
                                        'K','L','M','N','O','P','Q','R','S','T',
                                        'U','V','W','X','Y','Z');

// 列名(A～ZZ)を取得する
function TForm1.GetColTitle(idx: Integer): string;
var
  t: string;
begin
  if idx <= 26 then
  begin;
    t := ColTitle[idx];
  end
  else
  begin
    t := ColTitle[(idx-1) div 26];
    t := t + ColTitle[((idx-1) mod 26) + 1];
  end;
  Result := t;
end;
```

‘A1:AB5’

実装する前の注意点

- Delphi 6／C++Builder 6
  - ExcelRange型ではなくRange型に書き換える
- Excelのマクロ記録機能を活用しよう
  - マクロの記録開始
  - 編集作業
  - マクロの記録終了
  - マクロの内容を確認
- 参考文献を活用しよう
  - 「できる大事典 Excel VBA 2000/2002/2003対応」インプレス
  - 「Excel VBA ポケットリファレンスー Excel97/2000/2002/2003対応」前田智美著 技術評論社

ヘッダー部のフォーマットの設定

```
// ヘッダー部の書式 - Rangeの書式その１
procedure TForm1.SetHeaderFormatClick(Sender: TObject);
var
  col_count: Integer;
  rs: string;
  er: ExcelRange;
begin
  col_count := Table1.FieldDefs.Count;
  rs := Format('%s%d:%s%d',
              [GetColTitle(1), header_count, GetColTitle(col_count), header_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat        := 'G/標準';       // 標準
  er.HorizontalAlignment := xlHAlignCenter; // 中央
  er.Font.Bold           := True;           // 太字
  er.Font.Italic         := True;           // 斜体
end;
```

	A	B	C	D	E	F
1	EmpNo	LastName	FirstName	PhoneExt	HireDate	Salary

‘A1:F1’

ヘッダー部の値の設定

```
// ヘッダー部の値(カラム名)設定 - Rangeの書式その２
procedure TForm1.SetHeaderValueClick(Sender: TObject);
var
  col_count: Integer;
  rs1: string;
  rs2: string;
  er: ExcelRange;
  i: Integer;
begin
  col_count := Table1.FieldDefs.Count;
  rs1 := Format('%s%d', [GetColTitle(1), header_count]);
  rs2 := Format('%s%d', [GetColTitle(col_count), header_count]);
  er := ExcelWorksheet1.Range[rs1, rs2];
  for i := 1 to Table1.FieldDefs.Count do
  begin
    er.Cells.Item[1, i] := Table1.FieldDefs.Items[i-1].Name;
  end;
end;
```

‘A1’～‘F1’

ボディー部のフォーマットの設定

```
// ボディー部の書式
procedure TForm1.SetBodyFormatClick(Sender: TObject);
var
  row_count: Integer;
  rs: string;
  er: ExcelRange;
begin
  Table1.Last;
  row_count := Table1.RecNo;
  // 数値(0を追加), 左寄せ
  rs := Format('%s%d:%s%d', [GetColTitle(1), header_count+1, GetColTitle(1), header_count+row_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat       := '000000';
  er.HorizontalAlignment := xlHAlignLeft;
  // 標準, 左寄せ
  rs := Format('%s%d:%s%d', [GetColTitle(2), header_count+1, GetColTitle(4), header_count+row_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat       := 'G/標準';
  er.HorizontalAlignment := xlHAlignLeft;
  // 日付, 右寄せ
  rs := Format('%s%d:%s%d', [GetColTitle(5), header_count+1, GetColTitle(5), header_count+row_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat       := '"西暦"yyyy"年"mm"月"dd"日"';
  er.HorizontalAlignment := xlHAlignRight;
  // 金額, 右寄せ
  rs := Format('%s%d:%s%d', [GetColTitle(6), header_count+1, GetColTitle(6), header_count+row_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat       := '¥#,##0.000;¥-#,##0.000';
  er.HorizontalAlignment := xlHAlignRight;
end;
```

8	000012	Lee	Terri	256	西暦1990年05月01日	¥45,332.000
9	000014	Hall	Stewart	227	西暦1990年06月04日	¥34,482.625

ボディー部の値の設定 ― 低速版

```
// ボディー部の値設定 - 低速版
procedure TForm1.SetBodyValuesSlow;
var
  row_count: Integer;
  i: Integer;
  rs: string;
  er: ExcelRange;
begin
  Table1.First;
  row_count := 0;
  while not Table1.Eof do
  begin
    Inc(row_count);
    for i := 1 to Table1.FieldDefs.Count do
    begin
      rs := Format('%s%d', [GetColTitle(i), header_count+row_count]);
      er := ExcelWorksheet1.Range[rs, EmptyParam];
      er.Value := Table1.FieldValues[Table1.FieldDefs.Items[i-1].Name];
    end;
    Table1.Next;
  end;
end;
```

セルに個別にアクセス

フッター部のフォーマットの設定

```
// フッター部の書式
procedure TForm1.SetFooterFormatClick(Sender: TObject);
var
  row_count: Integer;
  footer_row: Integer;
  rs: string;
  er: ExcelRange;
begin
  Table1.Last;
  row_count := Table1.RecNo;
  footer_row := header_count+row_count+1;

  // 複数セルのマージ
  rs := Format('%s%d:%s%d', [GetColTitle(1), footer_row, GetColTitle(5), footer_row]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.MergeCells := True;

  // 合計金額セルの書式
  rs := Format('%s%d', [GetColTitle(6), footer_row]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.NumberFormat        := '¥#,##0.000;¥-#,##0.000';
  er.HorizontalAlignment := xlHAlignRight;
  er.Font.Name           := 'ＭＳ明朝';
  er.Font.Size           := 16;
  er.Font.Bold           := True;
  er.Font.Italic         := True;
end;
```

43	0001145	Guckenheimer	Mark	221	西暦1994年05月02日	¥32,000,000
44						¥1,386,202.293

フッター部の値 － SUMによる自動計算

```
// 合計金額セルの値設定 - SUMマクロの利用
procedure TForm1.SetFooterValueClick(Sender: TObject);
var
  row_count: Integer;
  footer_row: Integer;
  rs: string;
  er: ExcelRange;
begin
  Table1.Last;
  row_count := Table1.RecNo;
  footer_row := header_count+row_count+1;
  // SUMマクロをセルに指定
  rs := Format('%s%d', [GetColTitle(6), footer_row]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.Value := Format('=SUM(%s%d:%s%d)',
                     [GetColTitle(6), header_count+1, GetColTitle(6), header_count+row_count]);
end;
```

ヘッダー&フッター部の背景色を同時に設定

```
// セルの背景色の指定
procedure TForm1.SetCellColorClick(Sender: TObject);
var
  row_count: Integer;
  footer_row: Integer;
  rs1: string;
  rs2: string;
  er: ExcelRange;
begin
  Table1.Last;
  row_count := Table1.RecNo;
  footer_row := header_count+row_count+1;
  // セルの範囲指定にUnionメソッド(RangeC=RangeA+RangeB)を使ってみた
  rs1 := Format('%s%d:%s%d',
              [GetColTitle(1), header_count, GetColTitle(6), header_count]);
  rs2 := Format('%s%d:%s%d',
              [GetColTitle(1), footer_row, GetColTitle(6), footer_row]);
  er := ExcelApplication1.Union(ExcelWorksheet1.Range[rs1, EmptyParam], ExcelWorksheet1.Range[rs2, EmptyParam]);
  // InteriorのColorを利用
  ColorDialog1.Color := er.Interior.Color;
  if ColorDialog1.Execute then
  begin
    er.Interior.Color := ColorDialog1.Color;
  end;
end;
```

‘A1:F1’ + ‘A44:F44’

Union

特定の領域に枠を付ける

```
// セルの枠の指定
procedure TForm1.SetCellBorderClick(Sender: TObject);
var
  row_count: Integer;
  footer_row: Integer;
  rs: string;
  er: ExcelRange;
begin
  Table1.Last;
  row_count := Table1.RecNo;
  footer_row := header_count+row_count+1;
  // 右下がりの斜線＼(xlDiagonalDown)を引く
  rs := Format('%s%d:%s%d',
              [GetColTitle(1), footer_row, GetColTitle(5), footer_row]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.Borders.Item[xlDiagonalDown].LineStyle := xlContinuous; // 直線
  er.Borders.Item[xlDiagonalDown].Weight    := xlThin;       // 細
  er.Borders.Item[xlDiagonalDown].Color     := clBlack;      // 黒
  // 領域全体を二重線(xlDouble)で囲む
  rs := Format('%s%d:%s%d',
              [GetColTitle(1), header_count, GetColTitle(6), footer_row]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.BorderAround(xlDouble, xlThick, xlAutomatic, clBlack);
end;
```

Excelとの接続を切る

```
procedure TForm1.DisconnectExcelClick(Sender: TObject);
begin
  // エクセルから切断
  ExcelWorksheet1.Disconnect;
  ExcelWorkbook1.Disconnect;
  ExcelApplication1.Disconnect;

  //try
  //  ExcelApplication1.Quit;
  //except
  //  on E: Exception do ShowMessage(E.Message);
  //end;

  Application.Terminate; // アプリケーションを終了して閉じる
end;
```

Excelの終了は基本的に不要？

セルへのアクセスの高速化

ExcelのGUI更新を一時的に止める
Variantの2次元配列で一度にセット

ExcelのGUI更新を一時的に止める

- セルの値を更新中、ワークシートに触れてしまう
  - ExcelのGUIを触れないようにしよう
- 結果として、無駄なGUIの更新が無くなり、処理が高速化

```
procedure TForm1.SetBodyValueClick(Sender: TObject);
begin
  if FastMode.Checked then // 高速版?
  begin
    ExcelApplication1.ScreenUpdating[0] := False;  // エクセルのGUIを停止
    try
      SetBodyValuesFast;
    finally
      ExcelApplication1.ScreenUpdating[0] := True; // エクセルのGUIを復活
    end;
  end
  else
  begin
    SetBodyValuesSlow;
  end;
end;
```


Variantの2次元配列で一度にセット

```
// ボディー部の値設定 - 高速版
procedure TForm1.SetBodyValuesFast;
var
  ar: Variant; // 2次元のVariant配列
  fv: Variant;
  row_count: Integer;
  i: Integer;
  rs: string;
  er: ExcelRange;
begin
  // 2次元のVariant配列を作成
  ar := VarArrayCreate([0, Table1.RecordCount-1, 0, Table1.FieldDefs.Count-1], varVariant);
  Table1.First;
  row_count := 0;
  while not Table1.Eof do
  begin
    for i := 0 to Table1.FieldDefs.Count-1 do
    begin
      fv := Table1.FieldValues[Table1.FieldDefs.Items[i].Name];
      VarArrayPut(ar, fv, [row_count, i]); // Variant配列にフィールドの値をセット
    end;
    Table1.Next;
    Inc(row_count);
  end;

  // エクセルの広い範囲に一気にVariant配列を書き込む!!
  rs := Format('%s%d:%s%d', [GetColTitle(1), header_count+1, GetColTitle(Table1.FieldDefs.Count), header_count+row_count]);
  er := ExcelWorksheet1.Range[rs, EmptyParam];
  er.Value2 := ar;
end;
```

Excel 2007 への対応方法

画像の追加 ― 挿入位置
ファイルの保存 ― xlFileFormat列挙子

CODE GEAR FROM Borland
Where Developers Matter
Integrated Development Environments for Windows, Java, and Web Developers
DEVELOPER CAMP

画像の追加

- Excel 2007 では、画像を挿入後、明示的に位置を指定し直す必要がある

```
// 画像の追加
procedure TForm1.AddImageClick(Sender: TObject);
var
  er: ExcelRange;
  ps: Pictures;
  p : Picture;
begin
  ExcelApplication1.ScreenUpdating[0] := False; // エクセルのGUIを停止
  try
    // 画像を追加するセルを選択状態にする
    er := ExcelWorksheet1.Range[Format('%s%d', ['G', 1]), EmptyParam];
    er.Select;
    // Pictureのインスタンスを生成して、画像を読み込んで追加
    ps := ExcelWorksheet1.Pictures as Pictures;
    p := ps.Insert('c:¥tmp¥codegear.jpg', EmptyParam);
    // Excel2007では位置の指定が必要!!
    p.Top  := er.Top;
    p.Left := er.Left;
  finally
    ExcelApplication1.ScreenUpdating[0] := True; // エクセルのGUIを復活
  end;
end;
```

ファイルの保存

- Excel 2007 では、従来のファイルフォーマットで保存しようとすると、エラーが発生する
  - 従来のxlFileFormat列挙子 – xlExcel9597 = 43
- Excel 2007 では、数値 56 を渡す必要がある
  - Excelのマクロ記録機能とヘルプで、保存時の値(56)を確認する
  - 定義が無いので、自前で定義する必要あり
  - const xlExcel8 = 56;
- Excel の各バージョン番号を確認する必要がある
  - TExcelApplicationのVersionプロパティ(文字列)
  - Office 2007 – '12.0'
  - Office 2003 – '11.0'
  - Office XP – '10.0'
  - Office 2000 – '9.0'
  - Office 97 – '8.0'または'8.0a'

ファイルの保存(続き)

```
// Excel2007のxlFileFormat列挙子, Excel2003までは xlExcel9597 = 43
const xlExcel8 = 56;
// XML形式では無い従来のフォーマット(.xls)でファイルを保存
procedure TForm1.SaveWorkbookClick(Sender: TObject);
var
  ver: WideString;
  format: Integer;
begin
  ver := ExcelApplication1.Version[0]; // エクセルのバージョン取得
  if ver = '12.0' then
    format := xlExcel8     // Excel2007
  else
    format := xlExcel9795; // Excel95～Excel2003

  if SaveDialog1.Execute then
  begin
    ExcelWorkbook1.SaveAs(SaveDialog1.FileName, format, EmptyParam, EmptyParam,
                          False, False, xlNoChange, xlUserResolution, False,
                          EmptyParam, EmptyParam, 0);
    ExcelWorkbook1.Close;
  end;
end;
```