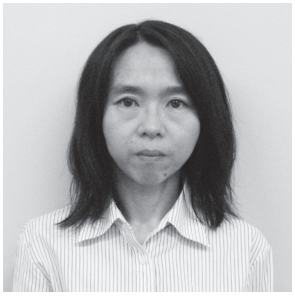
中嶋 祥子

株式会社ミガロ

RAD事業部 技術支援課

Delphi/400とExcelとの連携

本稿は
Delphi/400 アプリケーションと Excel をさらに使いこなすための
リファレンスである。
OLE オートメーションを使った Excel の基本的な出力方法から
Excel に対する各種テクニックをまとめた。



略歴
1968 年 2 月 23 日 生
1990 年奈良女子大学大学家政学卒
2002 年株式会社ミガロ入社
2002 年 11 月 RAD 事業部配属

現在の仕事内容
お客様からの Delphi/400 に関する技術的なご質問やお問い合わせに携わっている。また、メールマガジン「Migaro News」やホームページで、Tips など、開発に役立つ情報を担当していることもある。

1 Delphi/400アプリケーションとExcel

Delphi/400 でアプリケーションを作る場合、Excel の出力機能を利用しようとすると、帳票機能と同じぐらい組み込みを必要とされることが多い。

Excel を扱ううえで、専用のツール（例えばVB-Report3.0 など）を使わないケースでは、Office コンポーネントまたは OLE オートメーションを利用することになる。

Delphi/400 からの Excel の操作は、プログラムソースを見ると難しく感じるかもしれない。だが、よく使う Excel 操作は決まっているので、一度習得してしまえば非常に簡単に扱うことができる。

ここでは、Delphi/400 アプリケーションからの、OLE オートメーションによる Excel の基本的な出力方法や、Excel に対する各種テクニックを紹介しよう。

2 基本的なExcel出力

OLE オートメーションを使った基本的な Excel の出力は、手順として Excel と Book、Sheet を生成して、Cell に値を書き込んでいくという手順となる。

ソースコードを用意したので、参考にしてほしい。【図 1】

出力する Excel の全てを Delphi 側から操作するのは、プログラム的にもパフォーマンス的にも手間となる。そのため、Excel のテンプレートを用意すると、効率的な Excel 出力が行えるようになる。

具体的には、テンプレートとなる Excel をファイルコピーしたうえで、その Excel の中から必要な Cell 値の編集だけを行う。Cell の指定は、次のように指定する。

Cells［行番号 , 列番号］

テンプレートを利用する場合は、Excel 起動部分を、ソースのように処理することで指定の Excel を扱うことができる。【図 2】

3 Excel操作テクニック

ここから、Delphi/400 上で Excel を操作する際によく使われる動作を取り上げる。具体的なプログラムソースを交えて説明する。

①文字のフォント / フォントサイズを設定

フォントは、Font.Name で設定することができる。

フォントサイズは、Font.Size で設定することができる。【図 3】

②行の高さと列の幅を設定

行の高さは、RowHeight で設定することができる。

列の幅は、ColumnWidth で設定することができる。【図 4】

③オートフィットを設定

オートフィットは、AutoFit を使い、

図1

```
procedure TForm1.Button1Click(Sender: TObject);
var
  MsExcel      : Variant;
  MsApplication: Variant;
  WBook        : Variant;
  WSheet       : Variant;
begin
  //Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.WorkBooks.Add ;
  WSheet :=WBook.ActiveSheet;

  //ExcelのCellに値を書き込む
  WSheet.Cells[1,1].Value := 'データ1';
  WSheet.Cells[1,2].Value := 'データ2';
  WSheet.Cells[1,3].Value := 'データ3';

  //保存の確認を行う
  WBook.Saved  :=  False;
  //Excel終了
  MsApplication.WorkBooks.Close;
  MsExcel.Quit;
end;
```

図2

```
  // テンプレートファイルよりコピーして作業ファイルを作成
  CopyFile(PChar('コピー元ファイル'), PChar('コピー先ファイル'), False);
  //Excel起動
  MsExcel := CreateOleObject('Excel.Application');
  MsApplication := MsExcel.Application;
  MsApplication.Visible := True;
  WBook  := MsApplication.Workbooks.Open('コピー先ファイル');
  WSheet :=WBook.ActiveSheet;
```

図3

```
WSheet.Cells.Font.Name := 'ＭＳ Ｐゴシック';  //フォント名を設定
WSheet.Cells.Font.Size := 12;                 //フォントサイズを設定
```

図4

```
WSheet.Rows[1].RowHeight := 50;              //行の高さを設定
WSheet.Columns[1].ColumnWidth := 50;         //列の幅を設定
```

図5

```
WSheet.Cells.Select;                              //セルを全選択
WSheet.Cells.EntireColumn.AutoFit;                //オートフィットを設定
```

セルの値で幅を設定できる。【図5】

④先頭のシートを選択

ワークシートは、WorkSheets［ワークシート番号］.Selectで選択できる。【図6】

⑤Book名を指定して保存

Bookの保存は、SaveAsで、ファイル名や詳細パラメータを指定することができる。【図7】

⑥範囲を指定

セルの範囲は、Rangeで指定することができる。【図8】

⑦罫線を出力

罫線は、Rangeで範囲を指定し、出力することができる。

Borders.LineStyleで、罫線のスタイルを設定できる。

Borders.Weightで、罫線の太さを設定できる。【図9】

⑧セルを結合

複数セルは、Rangeで指定して、MergeCellsで結合できる。【図10】

⑨シートを印刷

ワークシートは、PrintOutで印刷することができる。【図11】

⑩シートをプレビュー表示

⑪警告メッセージを制御

Excelの警告メッセージは、DisplayAlertで出力制御をすることができる。【図12】

4 Excelのマクロ記録機能を活用

「Excel操作テクニック」では、具体的に、Delphi/400でよく使うプログラム例を挙げてみた。だが、例にないExcelの動作を行いたいときはどうすればいいのだろうか。

Excelでは、マクロで動作を記録して、VBのソースとして調べることができる。VBのソースはもちろんDelphiのソースとは異なる。とはいえ、操作やプロパティは同じように扱えるため、十分にDelphiのプログラムの参考とすることができる。

Excelのマクロ記録

ここでは、前述の⑥の［範囲を指定］を例に見ていこう。

⑴最初に、Excelの［ツール］→［マクロ］から［新しいマクロの記録］を実行する。【図13】【図14】
⑵これで、マクロが記録状態となるので、調べたいExcelの操作を実際に行う。ここでは、A1からC5のセルを選択して、Ctrl + Cでコピーする。
⑶Excelの操作が終わったら、マクロの記録を終了する。【図15】

VBのソースで確認

では、マクロが記録されているので、VBのソースで、この記録されたExcelの操作を確認してみよう。

⑷確認するマクロを選択し、［編集］で開けば、マクロのソースを見ることができる。【図16】

ソースを見てみると、A1からC5のセルの範囲はRangeで指定されて、Selectで選択されていることがわかる。また、コピーは、Copyというメソッドが使用されているのがわかる。【図17】

Delphiのソースに組み替え

このソースを参考に、Delphiのソースに組み替えていく。すると、⑥の［範囲を指定］のようなプログラムを作成することができる。

Excelの操作は、基本的に何でもマクロに記録できる。そのため、参考となるようなソースが見つけられないときには、これらの調査を行って、独自にプログラムを作成することができる。少し応用的な内容ではあるが、Excelのプログラムで詰まった際にはぜひ試してみてほしい。

5 Excel2007への対応

Excel2007のPCで、xlsx形式を指定してファイル保存をしたいことがある。その場合、実行PCのExcelのバージョンがExcel2007かどうかを判断する必要がある。

ExcelApplicationのVersionで、Excelのバージョンを取得/判断することができる。

各Excelのバージョンは、以下の通りである。

・2007 - ‘12.0‘
・2003 - ‘11.0‘
・XP　- ‘10.0‘
・2000 - ‘ 9.0‘
・97　- ‘ 8.0‘または ‘8.0a‘

例えば、バージョンを判断してファイル名（拡張子）を設定する場合のソースを作成してみた。【図18】

応用して組み込めば、これまでの作成済のExcel処理も、バージョンに対応した汎用的な作りに組み替えることができるだろう。

6 まとめ

本稿では、基本的なExcelの操作から、応用的な調査方法まで説明してきた。これらを、DelphiでExcelを扱う際に、リファレンスとして活用してほしい。

具体的なコード例をコピーして使うのもよいし、また中身の仕組みがわかってくれば、自分で新しいExcel操作の調査もできるようになる。

DelphiからのExcelの操作はたくさんあるので、それらのテクニックをマスターして、ぜひ自分用のExcelリファレンスを作り上げてほしい。

M

図6

```
WBook.Worksheets[1].Select(True);            //WorkBookの先頭(1)Sheetを選択
```

図7

```
WBook.SaveAs('保存ファイル名.xls');
```

図8

```
WSheet.Range['A1','C5'].Copy;       //セルのA1からC5の範囲を選択してコピー
WSheet.Range['A6'].Select;          //セルのA6を選択
WSheet.Paste;                       //コピーした内容を貼り付け
```

図9

```
//セルのA1からC5の罫線スタイルを設定
WSheet.Range['A1','C5'].Borders[8].LineStyle := 1;
//セルのA1からC5の罫線の太さを設定
WSheet.Range['A1','C5'].Borders[8].Weight := 2;
```

図10

```
WSheet.Range['A1','C5'].MergeCells := true;    //セルのA1からC5を選択してセル結合
```

図11

```
WSheet.PrintOut ;       //WorkSheetを印刷
```

図12

```
MsApplication.DisplayAlerts := false;    //警告メッセージが出力されないよう設定
```

図13

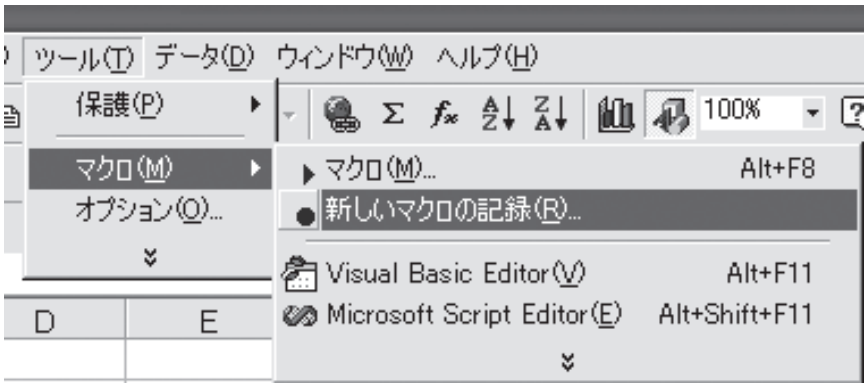


図14

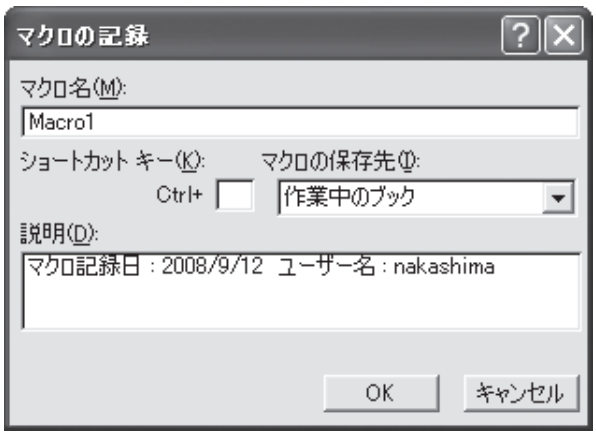


図15

図16

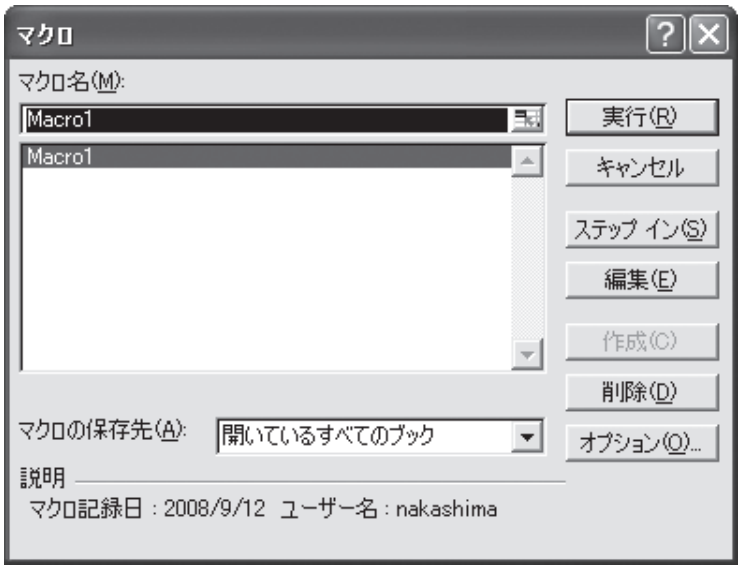


図17

```
' Macro1 Macro
' マクロ記録日 : 2008/9/12  ユーザー名 :nakashima
'

'
    Range("A1:C5").Select
    Selection.Copy
End Sub
```

セルのA1からC5を選択してコピー

図18

```
//バージョンを確認してファイル名(拡張子)を判断
if MsApplication.Version = '12.0' then
  WBook.SaveAs('C:¥Test.xlsx')
else
  WBook.SaveAs('C:¥Test.xls');
```