

吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課 顧客サポート

Delphi/400:OpenOffice実践活用

オープンソース OpenOffice.Org の「Calc」。これを題材にして、Delphi/400 において Excel 同様「Calc」もプログラムから利用できることを紹介する。

- OpenOffice とは
- Delphi/400 からの活用
- OpenOffice のプログラム操作
- 応用開発
- まとめ



略歴

1978年03月26日生
2001年龍谷大学法学部卒
2005年07月株式会社ミガロ、入社
2005年07月システム事業部配属
2007年04月RAD事業部配属

現在の仕事内容

Delphi/400 と JACi400 の製品試験、および月 100 件に及ぶ問い合わせやサポート、セミナー講師などを担当している。

1. OpenOffice とは

近年は Web 上から無償でダウンロードして、利用することができるソフトウェアが増えている。OpenOffice はそうした無償利用できるソフトウェアの 1 つである。

OpenOffice は、正式には「OpenOffice. Org (オープンオフィスオルグ)」というソフトウェア名である。名前の通り、オープンソースで開発されたオフィス統合環境ソフトとなっている。(※)【図 1】

【OpenOffice.org 日本語プロジェクト】
<http://ja.openoffice.org/>

一般に使われているマイクロソフトの Office 製品とも非常に互換性が高く、品質も非常によい。もちろん Excel や Word などの主要な機能 (アプリケーション) はほとんど揃っている。

例えば、Excel に対応する表計算ソフトとしては「Calc」、Word に対応するワープロソフトとしては「Writer」、

PowerPoint に対応するプレゼンテーションソフトとしては「Impress」等があり、そのほかペイント、HP 作成などの機能が用意されている。

サンプル画面は Excel と思われるかもしれないが、OpenOffice の Calc という機能である。外観も使い方もほぼ Excel と同じである。【図 2】

この OpenOffice は誰でも無償で利用できることもあり、個人だけではなく、企業や官公庁・教育機関など幅広く利用されている。マイクロソフトの Office 製品を購入して使用している場合でも、2 台目以降の PC には OpenOffice を導入することも少なくない。

また、OpenOffice はマイクロソフト製品ではないので、Linux や Solaris など Windows 以外のプラットフォームで使用できる。これも大きな特徴だと言えるだろう。

特にここ数年では、Windows2000 のメーカーサポートが終了したことにより、古くなった Windows2000 の PC を Linux に移行するなどのケースでも

OpenOffice が活用されている。

※本稿においては、バージョン 3.2.1 の OpenOffice.org を題材としている。

2. Delphi/400 からの活用

Delphi/400 では、IBM i から抽出したデータを Excel にアウトプットするプログラムを作ることが多い。本稿では、Excel と互換性が高い OpenOffice の Calc を題材にしている。Delphi/400 において、Excel 同様に Calc もプログラムから利用できることを紹介したい。

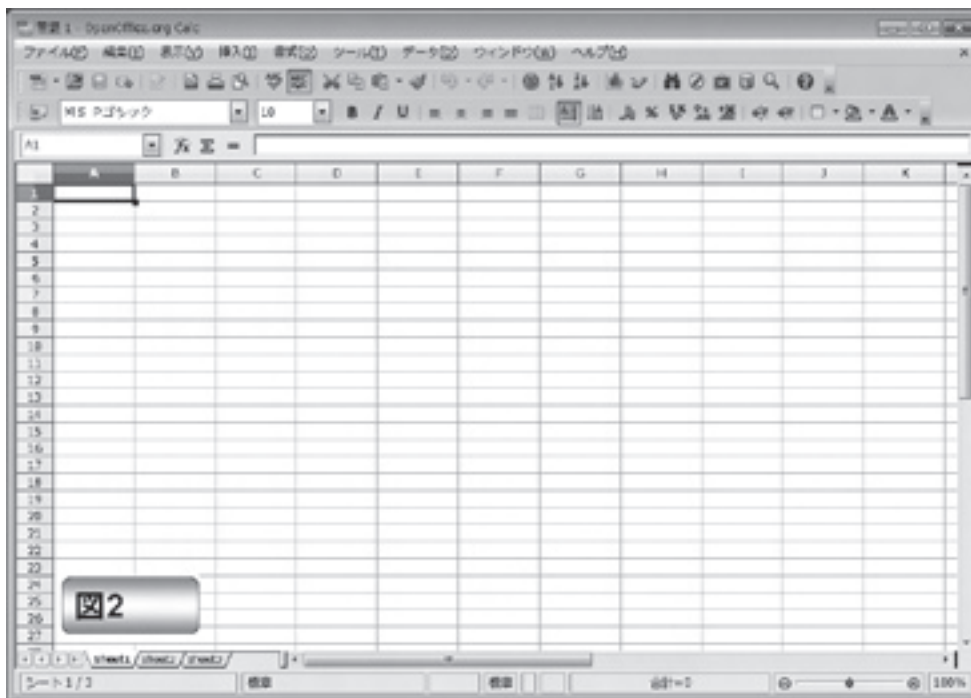
Delphi/400 で Excel を利用する場合には、通常 OLE (Object Linking and Embedding) という技術を使う。OLE とは簡単に説明すると、「①別のアプリケーションソフト (Excel) の機能を、あたかも②自分の機能 (Delphi/400) であるかのように提供することができる技術」である。

つまり、Excel 側 (①) には OLE で利用できる機能が予め用意されているの

図1



図2



ソース1

ソース1: Calc操作を行うための共通変数・関数

```
private
vOpenOffice : Variant; //サービスマネージャ
vStarDesktop : Variant; //サービス
vDocument : Variant; //ドキュメント
vSheet : Variant; //シート
function dummyArray: Variant; //空プロパティ配列設定用

...
//空のプロパティ配列を返却
function TForm1.dummyArray: Variant;
begin
Result:= VarArrayCreate([0, -1], varVariant);
end;
```

で、Delphi/400 側 (②) はそれと呼び出して活用しているわけである。

Calc にもこの OLE の機能が用意されているので、①部分が Calc に変わるだけと考えてほしい。このようにイメージすると、Calc が Excel と同じ方法で、Delphi/400 から利用できることがわかる。

なお、本稿では、Delphi/400 で Excel 操作をプログラム作成したことがある開発者向けに、OLE での Excel 操作を類似例として取り上げ説明を加えているので、理解の一助としていただきたい。

3. OpenOffice のプログラム操作

前章では、Delphi/400 での OpenOffice の扱い方について触れた。ここからは具体的なプログラミングについて説明していきたい。

3-1. 共通変数/関数の定義

まず、OpenOffice の Calc をプログラム上で操作するために、共通変数や共通関数を用意する。これは必ずしも必要なわけではないが、用意しておくことで各プログラムで便利なので参考としてほしい。【ソース 1】

共通変数については、4つ用意している。この共通変数は Excel で考えるとそれぞれ、次のような役割となる。

Calc	Excel
サービスマネージャ	Office
サービス	Excel
ドキュメント	Book
シート	Sheet

● dummyArray

また 1つだけ、dummyArray という共通関数を用意している。この関数は、単純に空の配列 (ダミー配列) を返却するだけの関数である。この関数を用意すると、Calc をプログラムで操作する際に非常に便利になる。その理由を以下に説明する。

Calc では、用意されている OLE の機能のパラメータに、PropertyValue と

いう構造体がよく使用されている。このパラメータは、オプション名 (Name) と設定値 (Value) を配列で扱う構造体である。

例えば、ソース 2 を見てもらいたい。

【ソース 2】

このソースはファイルを保存する操作になる。この最後の行で、StoreToURL という関数に、PropertyValue のパラメータを渡している。この StoreToURL 関数を使用している行の前部分は、PropertyValue の配列を作成する内容である。

PropertyValue は便利なパラメータではあるのだが、パラメータを必要としない場合でも、このような受け渡しのためにわざわざ配列を作成しなければならない。

そこで、前述のソース 1 で作成した dummyArray 関数を使ってみると、ソース 3 のように簡略化することができるのである。【ソース 3】

ここまでが、プログラムの下準備となる。

●プログラム操作

続いて、具体的な Calc のプログラム操作を説明していく。大きく分けて次の 3 点の操作をおさえれば、Excel と同様に、Delphi/400 から自由に Calc を利用することができるだろう。

- ・起動と終了
- ・セルの編集
- ・ファイルの保存

例として、IBM i からデータを抽出して Calc でファイル保存する、という工程を想定して説明していきたい。

3-2. 起動と終了

Calc を起動するには、ソース 4 のようなプログラムになる。【ソース 4】

Calc を起動する場合、まず、前述したサービスマネージャ、サービス、ドキュメントを作成する。これは、Excel で Office、Excel、Book を作成する操作と同じである。

プログラムでは、新しいドキュメント (Book) を作成している。もし既に作成しているファイルを読み込む場合は、ソース 4 でコメントをしているように、

「既存ドキュメントを読み込む場合」のプログラムコードを使うことになる。この時、パラメータに既存のファイルを設定すれば、ファイルを読み込んで開くことができる。

次に Calc を終了する場合のプログラムは、ソース 5 を参考してもらいたい。

【ソース 5】

終了のプログラムは、起動の際に作成したサービスやサービスマネージャを終了して、破棄するだけである。

以上が、基本となる起動と終了のプログラム操作である。

3-3. セルの編集

次に、ドキュメントのセルを編集するプログラム操作の説明である。セルの値を編集することができれば、IBM i から取得したデータを書き込むことができる。つまり、これでデータの出力などに利用することもできるようになる。

セルを編集するプログラムは単純で、シート上のセル (Excel と同じ考え方) を指定して、値を代入するだけである。【ソース 6】

IBM i からデータを出力する場合は、1レコードずつ読み込んで、このセル書き込みを応用すれば実現できる。【ソース 7】 【図 3】

3-4. ファイルの保存

最後に、編集したファイルを保存することになる。プログラム操作は、ソース 8 のようになる。【ソース 8】。

ここでは、SaveDialog コンポーネントを使い、ダイアログを開いて保存するファイル名を指定させている。

Calc で保存や読み込むファイルは、標準では “.ods” という拡張子のファイルになる。もちろん Excel との互換性が高いので “.xls” のファイルを保存したり、読み込んだりすることもできる。(これについては、次章で説明する)。

注意点としては、通常 Windows のファイルパスは “\” 形式になるが、OpenOffice でのファイルパスは “/” 形式で指定する必要がある。このソース 8 においては、SaveDialog コンポーネントで取得したファイルパスを、StringReplace という置換関数を使って

ソース2

ソース2: PropertyValueのパラメータ

ファイル保存のAPIを利用する例

```
//パラメータ用プロパティ配列生成
vaProperties := VarArrayCreate([0, 0], varVariant);
//プロパティ用オプション生成
vProperty := vOpenOffice.Bridge_GetStruct('com.sun.star.beans.PropertyValue');
//プロパティ用オプション値設定
vProperty.Name := 'FilterName';
vProperty.Value := 'MS Excel 97';
//オプション内容をプロパティ配列に設定
vaProperties[0] := vProperty;
//ファイルを保存
vDocument.StoreToURL('file:/// + sFileName, vaProperties);
```

PropertyValueを作成してAPIへ渡す

ソース3

ソース3: PropertyValueのパラメータ(dummyArray)

ファイル保存のAPIを利用する例

```
//ファイルを保存
vDocument.StoreToURL('file:/// + sFileName, dummyArray);
```

オプション設定が必要ない場合、dummyArrayで代用する

ソース4

ソース4: Calcを起動する

```
procedure TForm1.btnCalcStartClick(Sender: TObject);
begin
//サービスマネージャ生成
vOpenOffice := CreateOleObject('com.sun.star.ServiceManager');
//サービス生成
vStarDesktop := vOpenOffice.CreateInstance('com.sun.star.frame.Desktop');
//新規ドキュメントを生成する場合
vDocument := vStarDesktop.loadComponentFromURL('private:factory/scalc'
, '_blank'
, 0
, dummyArray);

//既存ドキュメントを読み込む場合
// vDocument := StarDesktop.loadComponentFromURL('file:///C:/Temp/Test.ods'
// , '_blank'
// , 0
// , dummyArray);

//アクティブなSheetを設定(1番目のSheet)
vSheet := vDocument.Sheets.getByIndex(0);
//Sheet名変更
vSheet.Name := '新しいシート';
end;
```

新規ドキュメントは
"private:factory/scalc"を指定

_blank:
すでにファイルが開かれていても新しいウインドウを使って指定のファイルを開く

既存ドキュメントを開く場合はこちら

ソース5

ソース5: Calcを終了する

```
procedure TForm1.btnCalcEndClick(Sender: TObject);
begin
//サービス終了
vStarDesktop.terminate;
//サービス破棄
vStarDesktop := unassigned;
//サービスマネージャ破棄
vOpenOffice := unassigned;
end;
```


“¥”を“/”に変換している。通常は、このプログラムコードを真似すれば問題ない。

また Calc を終了する際に、ファイル変更の警告ダイアログを出したくない場合は、Modified := False というプログラムコードで、ファイル変更なしに設定しておくといよい。

以上で、Delphi/400 から OpenOffice の Calc を操作して、抽出データをファイルに出力することができた。

4. 応用開発

前章では、Delphi/400 からの基本的な Calc の操作方法を説明してきた。この章では、Calc を業務アプリケーションでさらに活用するためのテクニックを何点か紹介したい。

4-1. xls形式での保存

Calc での標準ファイル形式は、前述したように“ods”という拡張子になる。もちろん Excel のファイルを開いたり、保存したりすることもできる。

ソースを見てもらうとわかるが、基本のプログラムは“ods”での保存と同じである。異なる部分に注釈を入れているが、PropertyValue を作成して、パラメータにファイル形式が Excel であることを設定している。このパラメータによって、本来“ods”形式の保存を“xls”形式で保存することができるのである。【ソース 9】

4-2. pdf形式での保存

Calc では、Excel と同じように“xls”形式で保存できることを説明したが、実は“xls”形式だけでなく、“pdf”形式での保存も可能である。

つまり、作成した出力データを Excel 用にも、PDF 用にも加工することができる。プログラムはソース 10 を見てもらいたい。【ソース 10】

pdf ファイルを作成するとなると難しいプログラムを想像するかもしれないが、ソース 9 とソース 10 を比べると、実は 1 行しか差異がない。具体的には、PropertyValue で Excel 形式を指定していた内容が、PDF 形式に変わっただ

けである。これだけ指定すれば、PDF の変換処理は Calc がやってくれるのである。

簡単かつ PDF が作成できる実践的なテクニックなので、ぜひ活用いただきたい。

4-3. プリンタへの出力

続いて、ファイルとして保存するだけでなく、プリンタへ直接印刷するプログラムを説明する。

ソース 11 は、プリンタへの印刷プログラムになる。【ソース 11】

実はこのプログラムコードのうち、印刷を行っているのは、最後の print という 1 行だけである。デフォルトプリンタへの印刷であれば、この 1 行で実行できる。

では、その前のプログラムは何を行っているかということ、印刷するプリンタの指定である。通常、指定なしで印刷すると、その PC でデフォルトに設定されているプリンタへ印刷が行われる。

プリンタを指定したい場合は、このソースのように、PropertyValue でプリンタの名前をセットして、setPrinter という関数を実行するとプリンタを切り替えることができる。

以上で、作成したファイルをプリンタへ直接印刷することが実現できる。

4-4. その他応用操作

ここまでいろいろな Calc の操作方法を具体的に説明してきたが、実際にアプリケーションを開発する際には、これ以外の操作が必要になってくる場合もあるだろう。そのような場合は、プログラム操作をマクロで調べることができる。

図 4 と図 5 に、Calc でのマクロ操作方法を載せている。【図 4】【図 5】

操作としては、Excel でのマクロ操作と同じである。マクロを保存すると、操作した内容をプログラムコードとして確認することができるので、それを参考にすれば Delphi/400 プログラムに活用することが可能になる。

5. まとめ

以上本稿では、Delphi/400 で、OpenOffice の Calc を利用するための

プログラミングを説明してきた。Excel を OLE で利用したことがある開発者であれば、多少プログラムの書き方が違うだけで、意外に簡単に思われたのではないだろうか。

また、初めてここで OLE の操作プログラムを読んだ開発者でも、掲載しているプログラムコードをコピーして真似すれば、簡単にアプリケーションに組み込むことができるだろう。

OpenOffice は無償でダウンロードしてすぐに試すこともできるソフトウェアなので、ぜひチャレンジして開発の幅を広げていただきたい。

M

ソース6

ソース6:Calcでセル値を編集する

```
procedure TForm1.btnCalcCellSetClick(Sender: TObject);
begin
  //セルに値を代入
  vSheet.getCellRangeByName('C2').String := 'オープンオフィスに書き込み';
end;
```

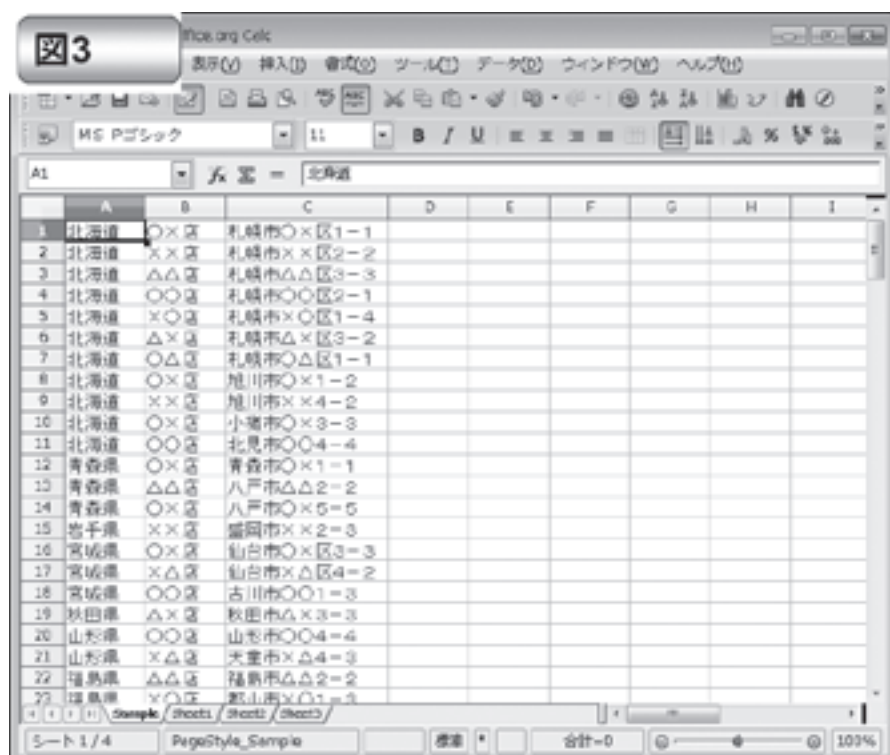
セルを指定してアクセス

ソース7

ソース7:Calcで読み込んだデータをセル値に出力する

```
procedure TForm1.btnCalcCellSetClick(Sender: TObject);
var
  i: Integer; //行数カウント
begin
  i := 1;
  with Table1 do
  begin
    Open;
    while not(EOF) do
    begin
      //ファイルを1レコードずつ読み込んでセルに値を代入
      vSheet.getCellRangeByName('A' + IntToStr(i)).String := FieldByName('Addr1').AsString;
      vSheet.getCellRangeByName('B' + IntToStr(i)).String := FieldByName('Addr2').AsString;
      vSheet.getCellRangeByName('C' + IntToStr(i)).String := FieldByName('Addr3').AsString;
      Inc(i);
    end;
  end;
  Close;
end;
```

図3



ソース8

ソース8: Calcでファイルを保存する

```

procedure TForm1.btnCalcSaveClick(Sender: TObject);
var
  sFileName : String; //保存ファイル名編集用
begin
  //保存ダイアログ実行
  if SaveDialog1.Execute then
  begin
    //ダイアログの保存パスを/形式に置換
    sFileName := StringReplace(SaveDialog1.FileName, '\', '/', [rfReplaceAll]);
    //ファイルを保存
    vDocument.StoreToURL('file:/// ' + sFileName, dummyArray);
    //変更保存等ダイアログを表示しない
    vDocument.Modified := False;
  end;
end;

```

保存パスはURL形式で設定する。
 × ...¥...¥Test.ods
 ○ .../.../ Test.ods

保存パスを指定して保存APIを実行

Calcを終了する際に変更保存ダイアログを表示させない設定

ソース9

ソース9: Calcでxls保存する

```

procedure TForm1.btnCalcXlsClick(Sender: TObject);
var
  vaProperties : variant; //保存パラメータ用(配列)
  vProperty : variant; //保存オプション用(配列)
  sFileBase : String; //保存ファイル名編集用
begin
  //保存ダイアログ実行
  if SaveDialog1.Execute then
  begin
    //ダイアログの保存パスを/形式に置換
    sFileBase := StringReplace(SaveDialog1.FileName, '\', '/', [rfReplaceAll]);
    //パラメータ用プロパティ配列生成
    vaProperties := VarArrayCreate([0, 0], varVariant);
    //プロパティ用オプション生成
    vProperty := vOpenOffice.Bridge_GetStruct('com.sun.star.beans.PropertyValue');
    //プロパティ用オプション値設定
    vProperty.Name := 'FilterName';
    vProperty.Value := 'MS Excel 97';
    //オプション内容をプロパティ配列に設定
    vaProperties[0] := vProperty;
    //ファイルを保存
    vDocument.StoreToURL('file:/// ' + sFileBase, vaProperties);
    //変更保存等ダイアログを表示しない
    vDocument.Modified := False;
  end;
end;

```

基本はodsと同じ、xlsで保存する場合は別形式なのでフィルタープロパティで指定が必要

ソース10

ソース10: CalcでPDF保存する

```

procedure TForm1.btnCalcPDFClick(Sender: TObject);
var
  vaProperties : variant; //保存パラメータ用(配列)
  vProperty : variant; //保存オプション用(配列)
  sFileBase : String; //保存ファイル名編集用
begin
  //保存ダイアログ実行
  if SaveDialog1.Execute then
  begin
    //ダイアログの保存パスを/形式に置換
    sFileBase := StringReplace(SaveDialog1.FileName, '\', '/', [rfReplaceAll]);
    //パラメータ用プロパティ配列生成
    vaProperties := VarArrayCreate([0, 0], varVariant);
    //プロパティ用オプション生成
    vProperty := vOpenOffice.Bridge_GetStruct('com.sun.star.beans.PropertyValue');
    //プロパティ用オプション値設定
    vProperty.Name := 'FilterName';
    vProperty.Value := 'calc_pdf_Export';
    //オプション内容をプロパティ配列に設定
    vaProperties[0] := vProperty;
    //ファイルを保存
    vDocument.StoreToURL('file:/// ' + sFileBase, vaProperties);
  end;
end;

```

基本はodsと同じ、pdfで保存する場合は別形式なのでフィルタープロパティで指定が必要。

ソース11: Calcで印刷する

```

procedure TForm1.BtnCalcPrintClick(Sender: TObject);
var
  vaProperties : Variant; //プリンタ指定パラメータ用(配列)
  vProperty    : Variant; //プリンタ指定オプション用(配列値)
begin
  //パラメータ用プロパティ配列生成
  vaProperties := VarArrayCreate([0, 0], varVariant);
  //プロパティ用オプション生成
  vProperty := vOpenOffice.Bridge_GetStruct('com.sun.star.beans.PropertyValue');
  //プロパティ用オプション値設定
  vProperty.Name := 'Name';
  vProperty.Value := 'FirePrint'; //設定したいプリンタ名
  //オプション内容をプロパティ配列に設定
  vaProperties[0] := vProperty;
  //プリンター指定
  vDocument.setPrinter(vaProperties);

  //印刷
  vDocument.print(dummyArray);
end;
    
```

プリンタを指定

デフォルトプリンタへの印刷であればこれだけ

図4

図4: Calc上での操作をマクロで保存



図5

図5: Calc上での操作をマクロで保存

