前坂 誠二

株式会社ミガロ.

システム事業部 システム2課

[Delphi/400] iOSモバイルアプリケーションによるファイル閲覧機能作成

略歴
1989 年 3 月 21 日生まれ
2011 年 3 月 関西大学 文学部卒業
2011 年 4 月 株式会社ミガロ. 入社
2011 年 4 月 システム事業部配属

現在の仕事内容
Delphi/400 を利用したシステム開発や保守作業を担当。Delphi、Delphi/400 の開発経験を積みながら、日々スキルを磨いている。

1. はじめに

近年、テレワークの導入や外出先で仕事をするといったように、場所にとらわれない働き方が増えている。これに伴って従来の PC 端末だけではなく、持ち運びに便利なタブレット端末を業務で使用する機会も増えている。

今はタブレット端末でも、社内のネットワークに接続すればファイルサーバー上のファイルを参照可能である。しかしタブレット端末を業務で使用する際に、「もっと効率よくファイルサーバー上の PDF ファイルや Excel ファイルなどを確認したい」と感じたことはないであろうか。

そこで本稿では、Delphi/400 10.2Tokyo を用いて、タブレット端末によるファイル閲覧機能の作成方法を紹介する。

本稿の実装例では、ファイルサーバーに配置されているファイルがコード単位で管理されているケースを想定し、一覧照会画面から画面タップのみで、そのままファイル内容を閲覧できる機能を作成する。【図 1】

なお、本稿で使用する Delphi/400 のバージョンは 10.2Tokyo、端末は iPad（iOS13.4.1）である。

2. 照会機能の実装

今回、開発フレームワークとして FireMonkey を使用し、DataSnap による 3 層構造でアプリケーションを作成する。3 層構造アプリケーションについては、2012 年発行のミガロ . テクニカルレポートにある『DataSnap を使用した 3 層アプリケーション構築技法』、また iOS での基本的な開発手法については 2014 年発行の『iOS/Android ネイティブアプリケーション入門』で詳しく説明しているので、それぞれ参考にしていただきたい。

照会機能については、上記レポートを参考に作成可能なので、本稿では詳細な説明は割愛する。今回は【ソース 1】の DDS より作成された「商品マスタ（ファイル名：MSSYHF）」を使用する。

2-1. DataSnap サーバープログラム

今回、サーバープログラムはサービスアプリケーションとして新規に作成する。新規作成のウィザードに沿って作成すると、以下の 2 つのユニットをもつプロジェクトが生成される。

(1) ServerControllerUnit1.pas
(2) ServerMethodsUnit1.pas

サーバープログラムをサービスアプリケーションとして登録する場合は、Windows のタスクマネージャーによりサーバーの開始・停止をコントロールするので、ServerControllerUnit1 の Name および DisplayName をタスクマネージャー上で確認しやすい名前に変更することを推奨する。【図 2】

次に、ServerMethodsUnit1 には IBM i と接続するためのデータベース関連のコンポーネントを配置する。本稿では FireDAC 接続を使用して IBM i と接続するため、TFDPhysCO400DriverLink、TFDTable、TDataSetProvider

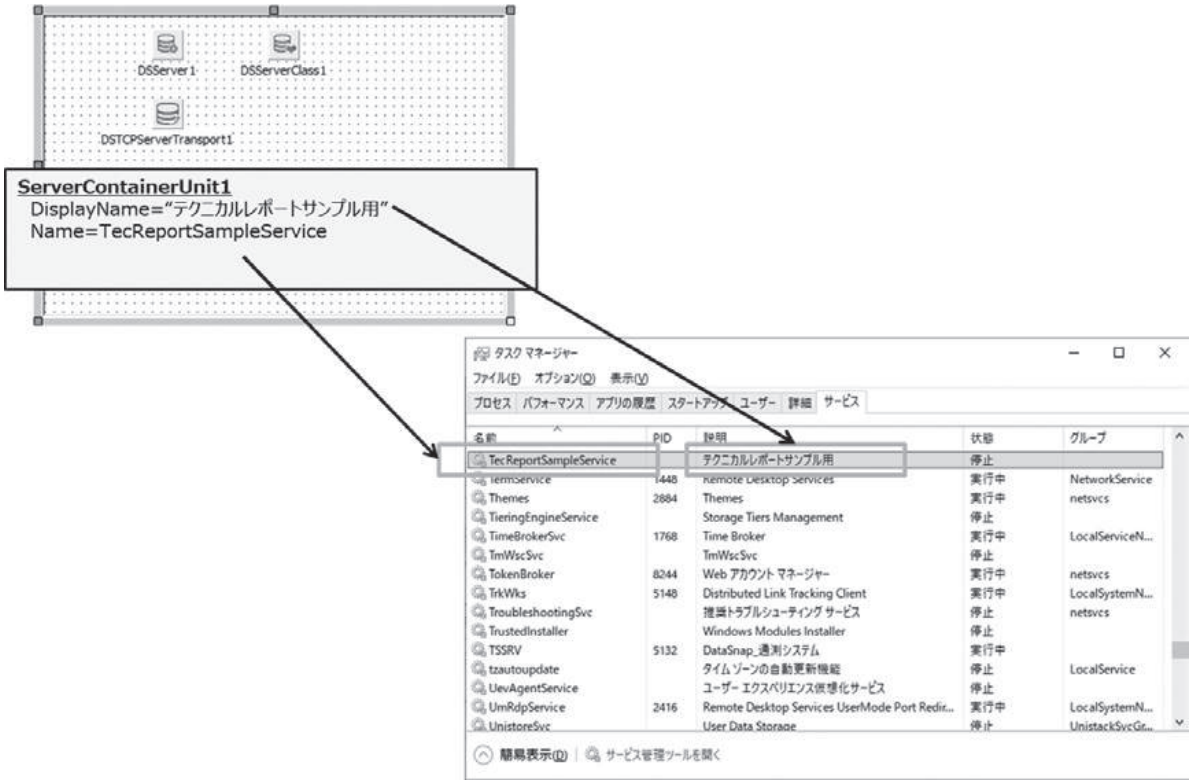
図1　本稿での作成画面

ソース1

DDS：商品マスタ

```
     ****************** データの始め **********************************
0001.00      A*************************************************************
0002.00      A*       FILE-ID    :    MSSYHF
0003.00      A*       FUNCTION   :    商品マスタ
0004.00      A*************************************************************
0005.00      A                                      UNIQUE
0006.00      A          R MSSYHR                    TEXT('商品マスタ')
0007.00      A            SYSHCD          5A        COLHDG('商品コード')
0008.00      A            SYSHNM         520        COLHDG('商品名')
0009.00      A          K SYSHCD
```

図2　サービスの名前指定



を貼り付け、プロパティを設定する。【図3】

一通りの設定が終わったら、サービスプログラムをコマンドプロンプトにてインストールし、タスクマネージャーからサービスの「開始」を実行する。【図4】

2-2. DataSnap クライアントプログラム

クライアントプログラムはiOS用のアプリケーションを作成するため、FireMonkeyのフレームワークを使って作成する。

[ファイル¦新規作成¦マルチデバイスアプリケーション]を選択し、新規作成を行うとForm1が生成される。画面デザインのスタイルはiOSに設定しておき、併せてプロジェクトマネージャのターゲットプラットフォームもiOSデバイスに設定しておく。【図5】

まずは、取得したマスタデータを表示するための画面レイアウト部分から作成する。TToolBar、TTabControlのコンポーネントを貼り付け、TTabControlは右クリックで「TTabItemの追加」を選択する。すると、TabItem1が追加されるので、追加したTabItem1上にTListViewを配置するようレイアウトする。【図6】

次に、先ほど作成したDataSnapサーバーアプリケーションとの接続処理部分を作成する。

TSQLConnection、TDSProviderConnection、TClientDataSetのコンポーネントを貼り付け、各プロパティの設定を行う。【図7】

あとは、画面表示時にconnServerのConnectedをTrueにしたあと、cdsSHDataClientをOpenし、lvSHDataにマスタの取得値を追加していけば、照会画面の完成である。【ソース2】【図8】

3. ファイル一覧機能の実装

ここでは、ファイルサーバーを参照してファイルの一覧を取得する。今回、ファイルサーバーには、「¥¥osknas02¥users¥osk前坂¥TecReport¥」に商品マスタの商品コード単位でフォルダが配置され、各フォルダ内にはその商品に関連するPDFファイルやExcelファイル等が配置されているものとする。【図9】

実装例では、前項で作成したプログラムを拡張してアプリケーションを作成するが、DataSnapサーバープログラムとDataSnapクライアントプログラムの両方を拡張する必要がある。そのため、まずは両方をひとまとめにしたプロジェクトグループを作成することを推奨する。【図10】

プロジェクトグループを作成しておくことで、「プロジェクトを閉じる→プロジェクトを開く」を繰り返し行う手間が省け、メンテナンス効率を向上させられる。

3-1. DataSnap サーバープログラム

まず、DataSnapサーバープログラムの拡張を行う。ServerMethodsUnit1にTClientDataSet、TDataSetProviderを貼り付け、プロパティを設定する。また配置したcdsFileListには、画面表示用項目としてファイル名（FLNAME）、ファイルサイズ（FLSIZE）を追加し、内部保持用項目としてファイルパス（FLPATH）、フォルダ区分（FLDRKB）の合計4つの項目を追加する。【図11】

なお項目の追加は、フィールドエディタより右クリックで「フィールドの新規追加」にて追加する。

次にソースを修正する。ServerMethodsUnit1のpublic宣言部にprocedure GetFileList（ADIR: String）を追加し、ロジックを記述する。GetFileListの手続きでは、引数でフォルダパスを受け取り、受け取ったパスでフォルダ内を参照する。

フォルダ参照時、ファイル名、ファイルサイズ、ファイルのフルパスをcdsFileListにAppendする。今回はロジックで指定した拡張子のみを一覧表示の対象とし、Word（.docx、.doc）、Excel（.xlsx、.xls）、PDF（.pdf）、画像（.jpg、.jpeg、.png）のファイルを拡張子に指定する。

なお、参照対象がファイルでなくフォルダであった場合は、FLDRKBに1をセットする【ソース3】。ロジックの修正が完了すれば、コンパイルを実施し、再度サーバープログラムを配置し直す必要がある。サーバープログラムのサービス実行中に再配置しようとするとエラーになるので、再配置の際は一度タスクマネージャーよりサービスを停止する必要がある。【図12】

また、今回のようにファイルサーバー接続時にサーバーへのアクセス権限が必要であれば、サーバー起動の際に併せて権限を付与する必要がある。

これにはまず、タスクマネージャーのサービスタブ下部にある「サービス管理ツールを開く」を選択する。サービス管理対象の一覧が表示されるので、自身が作成したサービスを選択し、右クリックからプロパティを選択する【図13】。すると、プロパティのダイアログ画面が表示されるので、ログオンタブを指定する。

設定値を「ローカルシステムアカウント」から「アカウント」に変更し、ファイルサーバーにアクセス可能なアカウント情報を設定してOKを押下すれば、権限の付与は完了である【図14】。あとは再度右クリックから「開始」を選択すると、サービスが開始される。

3-2. DataSnap クライアントプログラム

次に、DataSnapクライアントプログラムを拡張する。画面レイアウトの修正でまず行うのは、以下の2点である。

1点目はDataSnapサーバープログラムで取得したファイル一覧を表示するためのTListViewの追加、2点目はファイル一覧から照会画面に戻るためのTButtonの追加である。

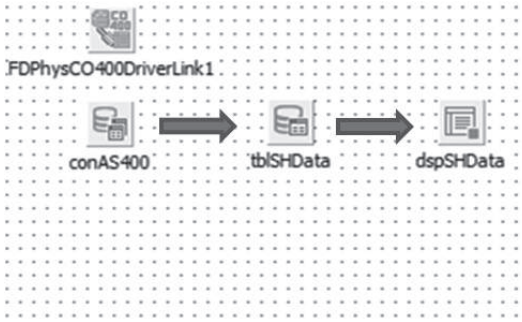
1点目のTListViewの追加は、前述したようにtbcMainより「TTabItemの追加」を行ったあと、TListViewを貼り付ける。2点目のボタン追加は、tbHeader上にTButtonを貼り付けるのみである。

画面レイアウトの修正が完了すると、次はDataSnapサーバープログラムとの連携用のコンポーネントを貼り付ける。TClientDataSet、TSqlServerMethodを貼り付け、プロパティを設定する。【図15】

ここでのコンポーネントの貼り付けは以上であり、次はソースを修正する。今回は商品一覧照会画面でレコードをタップした際、紐づく商品コードのファイル一覧を表示する。ソースの修正では、以下の2点の処理を追加する。

1点目は、商品のリストを選択した際のイベントを作成し、ファイル一覧データを取得して画面に表示する処理を追加する（追加処理1）。2点目は、前画面に戻るボタンを押下した際の処理を追加す

図3　コンポーネント設定値



FDPhysCO400DriverLink1 : TFDPhysCO400DriverLink1
設定値変更なし

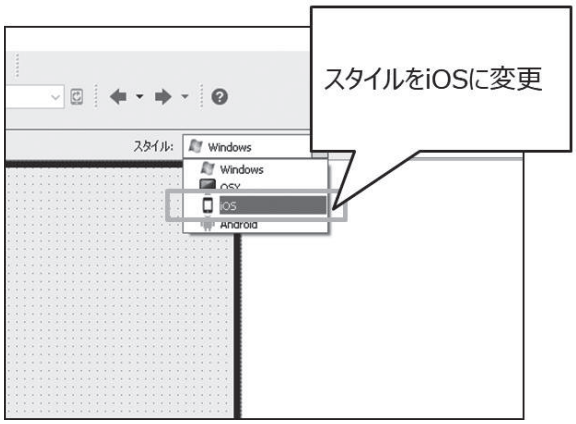
conAS400 : TFDConnection
DriverName=CO400
LoginPrompt=False
Params=(IBMi上のデータライブラリに接続する設定)

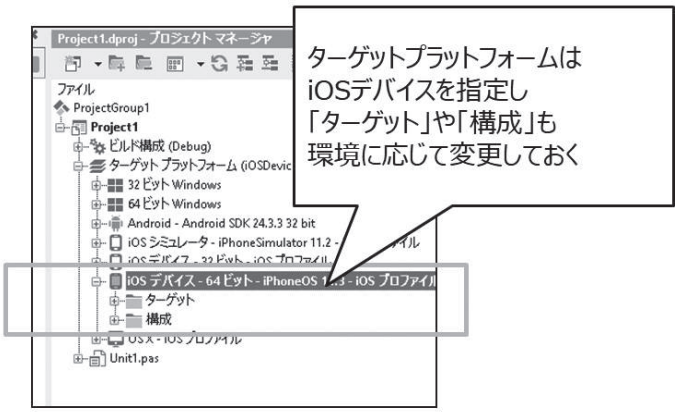
tblSHData : TFDTable
Connection=conAS400
TableName=MSSYHF

dspSHData : TDataSetProvider
DataSet=tblSHData

図4　サービスの開始

図5　iOSデバイスの指定





る（追加処理 2）。

追加処理 1 は、ファイル一覧データの取得と表示部分をまとめて、手続き GetFileListData にて記述する。

まずは Form1 の private 宣言部に、変数 FCurrDir と procedure GetFileListData を追加する。【ソース 4】

変数 FCurrDir では参照対象のフォルダパスを保持し、GetFileListData の手続きで svmGetFileList のパラメータとして渡す。その後、svmGetFileList の ExecuteMethod を呼び出すと、DataSnap サーバープログラムの GetFileList（【ソース 3】）が実行される。あとは cdsFileListClient を Open し、lvFileList に取得したファイル名（FLNAME）、ファイルサイズ（FLSIZE）を追加すると、GetFileListData の処理は完成である。

なお lvFileList に値をセットする際、取得内容がフォルダ（FLDRKB が 1）であった場合はファイルサイズが 0 となるため、フォルダ名のみを表示させている。【ソース 5】

次に、lvSHData の OnItemClickEx イベントにて処理を記述する。OnItemClickEx イベントでは、FCurrDir に参照フォルダパスの値をセットし、先ほど記述した GetFileListData の処理を呼び出す。

本サンプルでの参照フォルダは、「¥¥osknas02¥users¥osk 前坂 ¥TecReport¥」に商品コードを付与したパスであるため、まずは現在の選択行の商品コードを取得する処理を記述する。

商品コードは cdsSHData の SYSHCD で保持しており、RecNo のレコード位置を画面の選択行と同期をとることで、選択行の商品コードが取得可能である。OnItemClickEx イベントの引数である ItemIndex には選択行の番号を保持しているが、0 始まりであるため＋ 1 の値を RecNo にセットする。

RecNo の変更後、商品コード（SYSHCD）を付与した参照フォルダパスを FCurrDir にセットし、GetFileListData の処理を呼び出す。

あとは、ファイル一覧を表示するために ActiveTab を tbiFileList に変更する処理を追加すれば、追加処理 1 は完成である。【ソース 6】

追加処理 2 は、btnBack の OnClick イベントにて ActiveTab を tbiSHData に変更し、btnBack の Visible を False にする処理を記述する。また、画面起動時もボタンの表示は不要なので、Visible を False にしておく【ソース 7】。以上で、ファイル一覧機能の実装は完了である。【図 16】

4. ファイル閲覧機能の実装

ここでは、前の項で作成したファイル一覧より、選択ファイルの内容を表示する機能を実装する。ファイルの内容を表示する仕組みとしては、サーバープログラムにて取得したファイルデータをアプリ内の DataContainer の tmp フォルダに保存し、保存した内容を TWebBrowser コンポーネントで表示する仕組みである。【図 17】

実装例では、前の項で作成したプログラムを拡張してアプリケーションを作成する。

4-1. DataSnap サーバープログラム

まずは、DataSnap サーバープログラムを拡張する。

ファイル一覧機能作成時と同様に、ServerMethodsUnit1 に TClientDataSet と TDataSetProvider を貼り付け、プロパティを設定する。配置した cdsFileData には、内部保持用項目として、ファイルデータ（FLDATA）を追加する。FLDATA にはバイナリーデータとしてファイル内容を保持するため、TBlobField として項目を作成しておく。【図 18】

次にソースを修正する。ServerMethodsUnit1 の public 宣言部に procedure GetFileData（AFilePath: String）を追加し、ロジックを記述する。GetFileData の手続きでは、引数で表示対象ファイルのフルパスを受け取り、配置した cdsFileData の FLDATA にファイルデータを保持する処理を記述する。

ファイルデータを保持と聞くと、一見複雑な処理記述をイメージするかもしれないが、Delphi/400 では、作成した TBlobField に対して LoadFromFile（ファイルパス）を呼び出すだけで簡単にファイルデータを保持できる【ソース 8】。ロジックの修正が完了すれば、あとは前の項で実施したのと同様に、プログラムを再配置する。

4-2. DataSnap クライアントプログラム

次に、DataSnap クライアントプログラムを拡張する。画面レイアウトの修正でまず行うのは、以下の 2 点である。

1 点目は DataSnap サーバープログラムで取得したファイル内容を表示するための TWebBrowser の追加、2 点目はファイル一覧からフォルダを選択した際の、1 つ前のフォルダに戻るための TButton の追加である。

1 点目の TWebBrowser の追加には、まず前述したのと同様に TTabItem を追加し、TWebBrowser を貼り付ける。また 2 点目のボタン追加についても、前述したのと同様に tbHeader 上に TButton を貼り付ける。

画面レイアウトの修正が完了すると、次は DataSnap サーバープログラムで作成したファイルデータの取得処理を実行するためのコンポーネントを貼り付ける。こちらも前述したのと同様に、TClientDataSet と TSqlServerMethod を貼り付け、プロパティを設定する。【図 19】

コンポーネントの貼り付けは以上であり、次はソースを修正する。ここでは、ファイル一覧画面でレコードをタップした際に紐づくファイルの内容を表示する。ソースの修正では次の 3 点の処理を追加する。

1 点目は、ファイル一覧を選択した際のイベントを作成し、ファイルデータを wbFileData に表示するソースを記述する（追加処理 1）。2 点目は、1 つ上のフォルダに戻るボタンを押下した際の処理を追加する（追加処理 2）。そして 3 点目は、画面破棄時に tmp フォルダ以下に作成したファイルを削除する処理を追加する（追加処理 3）。

追加処理 1 はファイルデータの取得と表示部分をまとめて、手続き GetFileDataDisp にて記述する。

まずは、uses 節に System.IOUtils を追加し、private 宣言部にて変数 FTmpDir、FRootDir、FDirPathCnt、FBefoDir と procedure GetFileDataDisp を追加する。【ソース 9】

次に画面生成時に、FTmpDir の値セット処理、FBefoDir 変数の Create

図6　TTabItemの追加

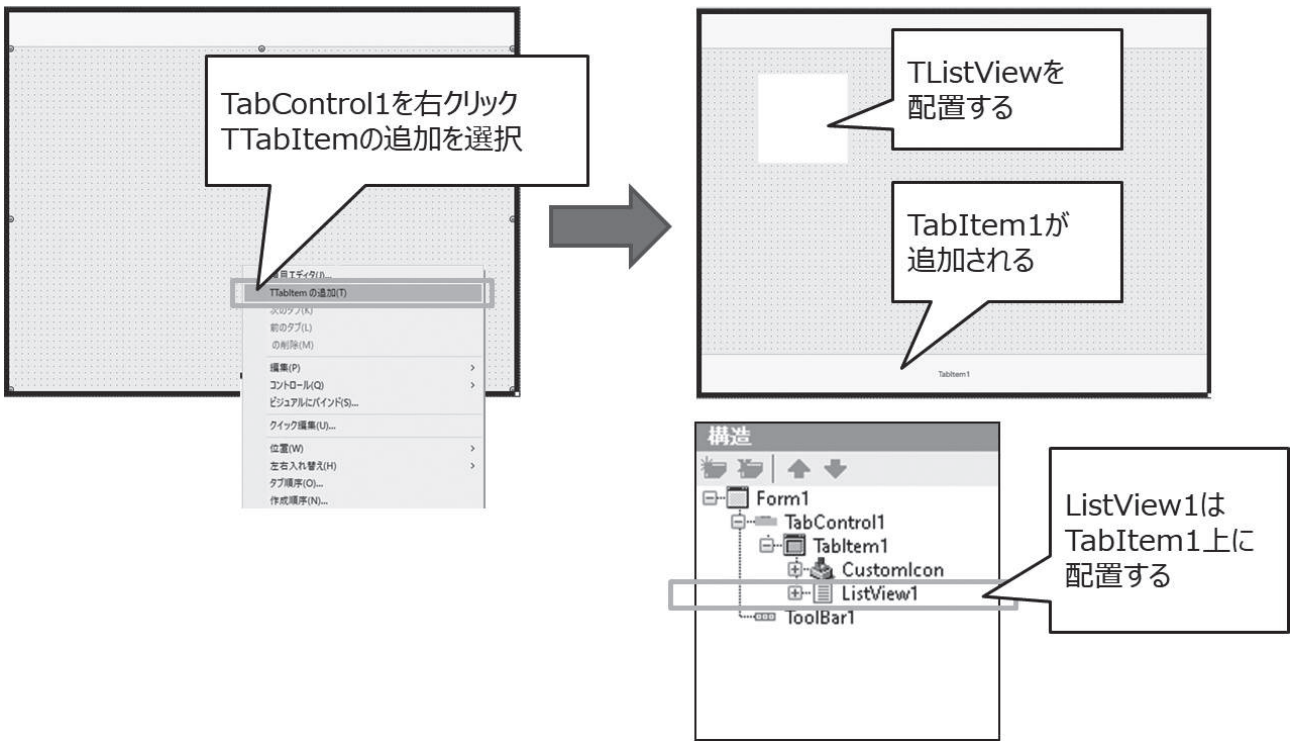
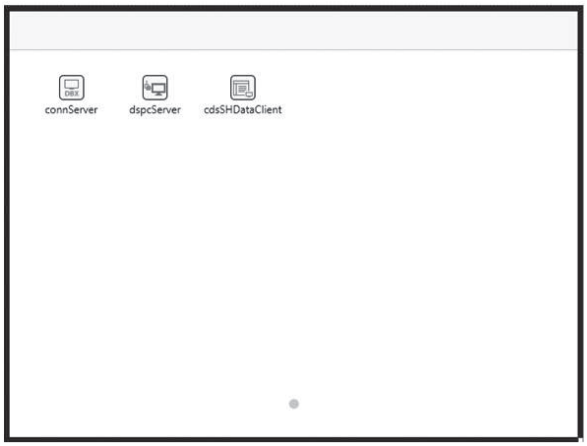


図7　コンポーネント設定値

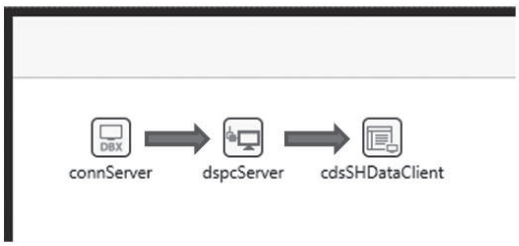


tbHeader : TToolBar
設定値変更なし

tbcMain : TTabControl
Align=Client
TabPosition=None

tbiSHData : TTabItem
設定値変更なし

lvSHData : TListView
Align=Client
ItemAppearance＞ItemHeight=80



connServer : TSQLConnection
Driver=DataSnap
ConnectionName=DataSnapCONNECTION
LoginPrompt=False
Params=(サーバのPort番号、HostNameを指定)

dspcServer : TDSProviderConnection
SQLConnection=connServer
ServerClassName=TServerMethods1

cdsSHDataClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspSHData

処理を記述し、商品一覧選択時に、FRootDirの値セット処理を追加する【ソース10】。FTmpDirでは、tmpフォルダのパスを内部保持している。

それから、GetFileDataDispの処理を記述する。GetFileDataDispでは、パラメータに表示対象のファイルパスを渡し、svmGetFileDataのExecuteMethodを呼び出す。その後、cdsFileDataClientをOpenし、SaveToFileメソッドを呼び出すだけでtmpフォルダへの保存が可能である。

ただしiOS端末でファイル保存の際は、濁点の付いたファイルは保存できないため、いったん別名に変換して保存しなければならない。

なお本稿では、ファイル名を連番値に変換して保存を実施している。あとはwbFileDataのNavigateメソッドで、「'file://'＋保存ファイルパス」を指定すれば、GetFileDataDisp処理の完成である。【ソース11】

次に、lvFileListのOnItemClickExイベントにて処理を記述する。OnItemClickExイベントではcdsFileListClientのレコード位置の同期後、選択行がファイルデータの場合に、先ほど記述したGetFileListDataの処理を呼び出し、フォルダの場合は前項で記述したファイル一覧のセット処理を呼び出す。【ソース12】

追加処理2は、btnUPDirのOnClickイベントにてFCurrDirの値を変更し、画面内容を再セットする処理を記述する。また画面起動時にボタンの表示は不要であるため、VisibleをFalseにしておく。【ソース13】

追加処理3は、OnDestroyイベントにて削除処理を記述する。【ソース14】

以上で、ファイル閲覧機能の実装は完了である。【図20】

なお、ExcelやWordは端末にインストールされていなくても閲覧可能であるが、Windowsにて作成されたファイルの場合は、罫線やオートシェイプなど一部表示されないものもあるので注意が必要である。

5. 他アプリへのファイル保存機能の実装

ここでは、前項で閲覧可能となったファイルに対し、端末にインストールされているiBookなど他アプリへの保存機能を追加する。

まずは、画面の上部にファイル保存のTButtonを追加する。ここでのソース修正については、DataSnapクライアントプログラムのみ行う。ファイル保存ではIOSapiのUIDocumentInteractionControllerの機能を利用し、ダイアログを表示して行うが、その際にファイルのフルパスとダイアログの表示位置を指定する必要がある。

そのため、まずはprivate宣言部にFFullFileNameとFPointの変数を追加する。FFullFileNameは、ファイルデータ表示処理のタイミングで値を保持するよう修正する。【ソース15】

次に、ファイル保存用に配置したボタンのOnTapイベントでタップした位置をFPointに内部保持し、OnClickイベントでファイル保存ダイアログの表示処理を記述していく。【ソース16】

以上で、他アプリへのファイル保存機能の実装は完了である。【図21】

6. さいごに

本稿では、iOSアプリでのファイル閲覧機能の作成について紹介してきた。FireMonkeyフレームワークの使用や3層構造といったテクニックも絡めながら実装例を紹介したため、難易度が高く感じられたかもしれない。

しかし実際には、サーバープログラムとクライアントプログラムを合わせても500ステップ前後の記述で実装できているのである。

今後、変化していく生活環境の中、プログラム開発者もより高度な要望や技術が求められていくだろう。Delphi/400では、このように少量のロジック記述で実現できることが多くあるので、高度な内容でもぜひ開発にチャレンジしていただきたい。その際、本稿で紹介した内容が少しでも参考になれば幸いである。

M

ソース2

OnShowイベント

```
{*******************************************************************************
 目的：画面表示時処理
 引数：
 戻値：
*******************************************************************************}
procedure TForm1.FormShow(Sender: TObject);
begin
  // サーバと接続
  connServer.Connected := True;

  // 商品マスタのデータ取得
  cdsSHDataClient.Open;

  // 画面にセット
  cdsSHDataClient.First;
  while not cdsSHDataClient.Eof do
  begin
    // 商品コード + スペース + 商品名
    lvSHData.Items.Add.Text :=
        cdsSHDataClient.FieldByName('SYSHCD').AsString + '  '
      + cdsSHDataClient.FieldByName('SYSHNM').AsString;

    cdsSHDataClient.Next;
  end;

  // 1行目を選択行に
  lvSHData.ItemIndex := 0;
end;
```

図8　照会画面

図9　参照フォルダ内容

図10　プロジェクトグループの作成

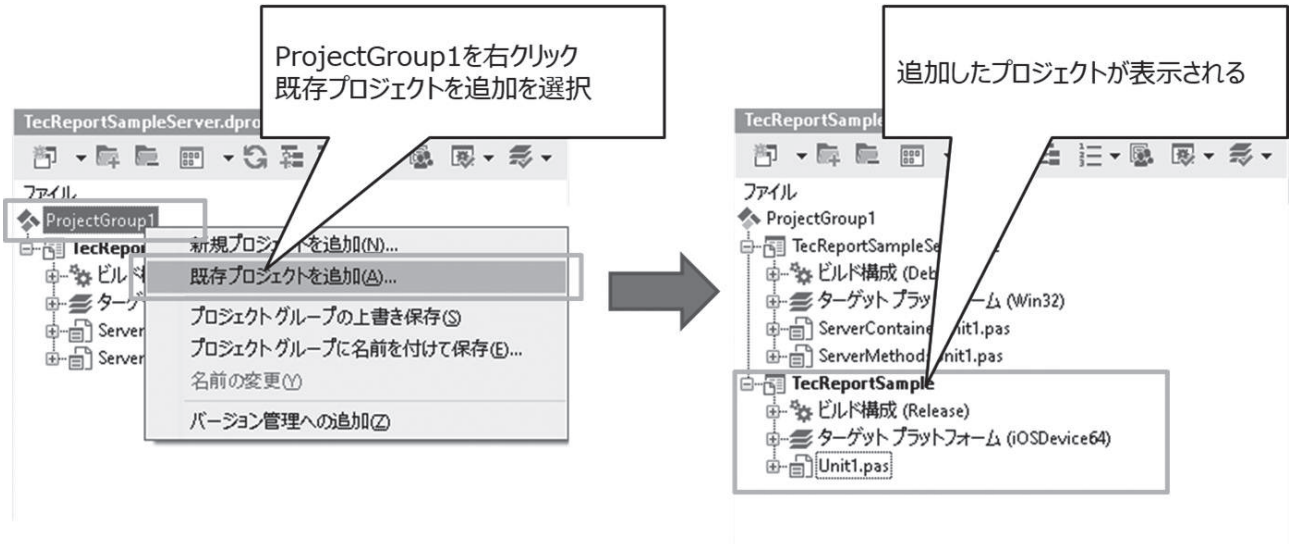
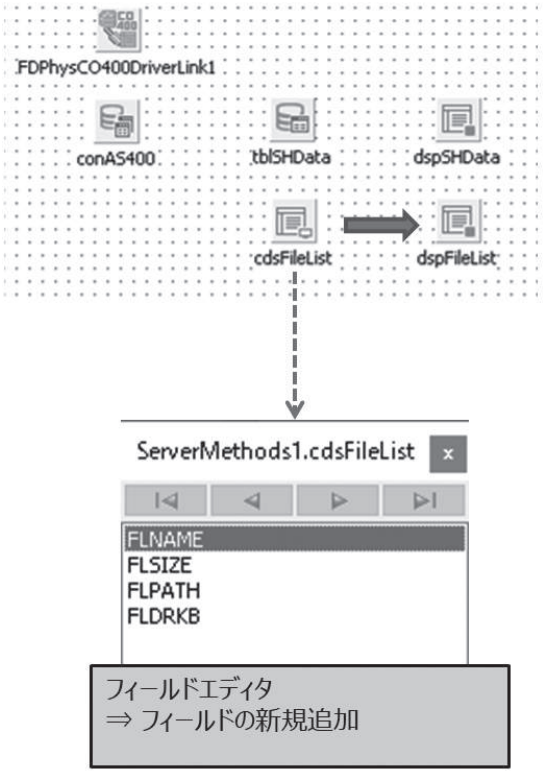


図11　コンポーネント設定値



cdsFileList : TClientDataSet
設定値変更なし

dspFileList : TDataSetProvider
DataSet=cdsFileList

FLNAME : TStringField
Size=500　※ 配置ファイル名の長さに合わせて指定

FLSIZE : TIntegerField
設定値変更なし

FLPATH : TStringField
Size=1000　※ 配置ファイルパスの長さに合わせて指定

FLDRKB : TIntegerField
設定値変更なし

GetFileList(指定フォルダのファイル一覧を取得)

```
{*****************************************************************************
  目的: ファイル一覧取得処理
  引数: ADIR - フォルダ名
  戻値:
*****************************************************************************}
procedure TServerMethods1.GetFileList(ADIR: String);
var
  SearchRec: TSearchRec;  // ファイル情報
  slExt: TStringList;     // 取得対象の拡張子保持
begin
  slExt := TStringList.Create;
  try
    // 指定した拡張子のファイルを表示する
    slExt.CommaText := '.DOCX,.DOC,.XLSX,.XLS,.PDF,.JPG,.JPEG,.PNG';

    // データセット準備
    cdsFileList.Close;
    cdsFileList.CreateDataSet;
    cdsFileList.EmptyDataSet;

    // フォルダが存在しない場合、処理中断
    if not System.SysUtils.DirectoryExists(ADIR) then
    begin
      Exit;
    end;

    // 指定ディレクトリのすべての種類のファイルを列挙
    if (System.SysUtils.FindFirst(ADIR + '*.*', faAnyFile, SearchRec) = 0) then
    begin
      try
        while (System.SysUtils.FindNext(SearchRec) = 0) do
        begin
          // カレントディレクトリ・親ディレクトリ・隠しファイルは対象外
          if (SearchRec.Name <> '.') and
             (SearchRec.Name <> '..') and
             (not Bool(SearchRec.Attr and FILE_ATTRIBUTE_HIDDEN)) then
          begin
            // 対象の拡張子のファイルかチェック
            if (slExt.Count <> 0) and
               (slExt.IndexOf(UpperCase(ExtractFileExt(SearchRec.Name))) < 0) and
               (not ((SearchRec.Attr and faDirectory) > 0)) then
            begin
              Continue;
            end;

            // ファイル名を明細に追加
            cdsFileList.Append;

            // ファイル名
            cdsFileList.FieldByName('FLNAME').AsString  := SearchRec.Name;

            // ファイルサイズ（キロバイト表示）
            cdsFileList.FieldByName('FLSIZE').AsInteger := SearchRec.Size div 1024;

            // ファイルパス
            cdsFileList.FieldByName('FLPATH').AsString  := ADIR + SearchRec.Name;
            if ((SearchRec.Attr and faDirectory) > 0) then
            begin
              // ディレクトリ
              cdsFileList.FieldByName('FLDRKB').AsInteger := 1;
            end
            else
            begin
              // ファイル
              cdsFileList.FieldByName('FLDRKB').AsInteger := 0;
            end;
            cdsFileList.Post;
          end;
        end;
      finally
        System.SysUtils.FindClose(SearchRec);
      end;
    end;

    cdsFileList.First;

  finally
    FreeAndNil(slExt);
  end;
end;
```

図12　サービス停止方法

図13　サービス管理ツールの起動

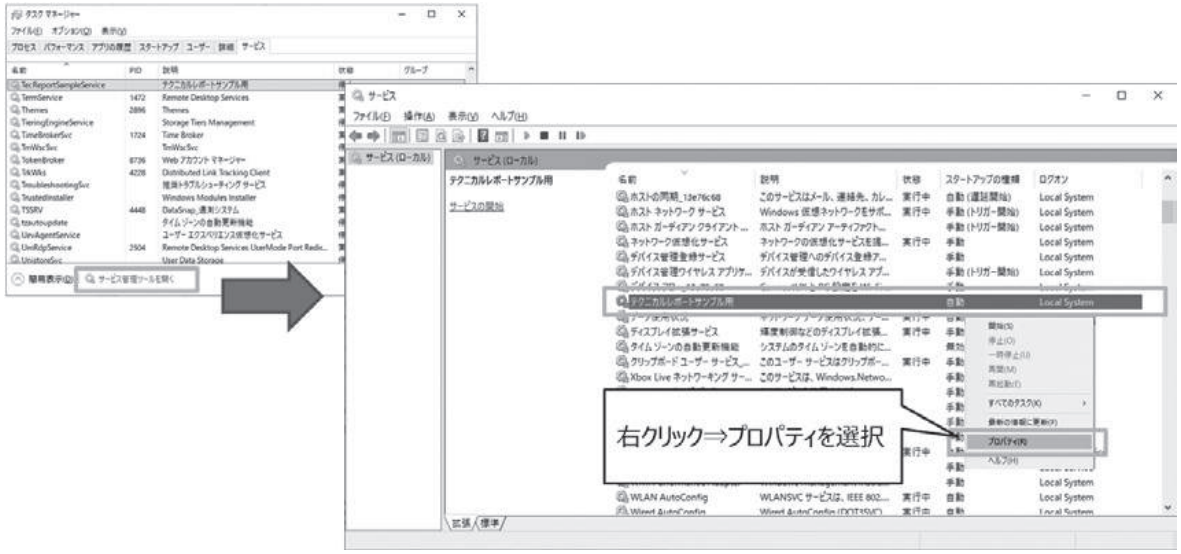


図14　ログオンアカウントの設定

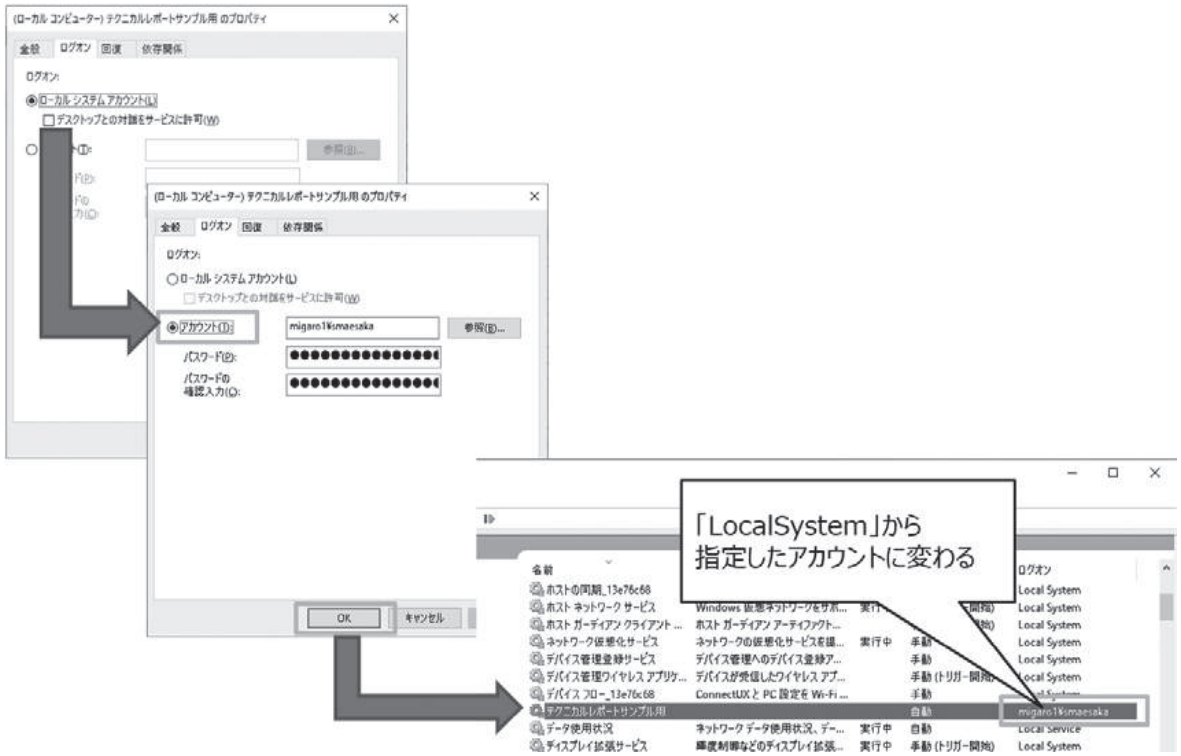


図15　コンポーネント設定値

tbiFileList : TTabItem
設定値変更なし

lvFileList : TListView
Align=Client
ItemAppearance> ItemHeight=80

btnBack : TButton
Align=MostLeft
Text="前画面へ戻る"

cdsFileListClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspFileList

svmGetFileList : TSqlServerMethods
SQLConnection=connServer
ServerMethodName=TServerMethods1.GetFileList

ソース4

private宣言部に追加

```
  Datasnap.DSConnect, Data.SqlExpr;

type
  TForm1 = class(TForm)
    tbHeader: TToolBar;
    tbcMain: TTabControl;
    tbiSHData: TTabItem;
    tbiFileList: TTabItem;
    connServer: TSQLConnection;
    dspcServer: TDSProviderConnection;
    cdsSHDataClient: TClientDataSet;
    cdsFileListClient: TClientDataSet;
    cdsFileDataClient: TClientDataSet;
    lvSHData: TListView;
    lvFileList: TListView;
    btnBack: TButton;
  private
    { private 宣言 }
    FCurrDir: String;                    // フォルダパスを保持
    procedure GetFileListData;           // ファイル一覧取得処理
  public
    { public 宣言 }
  end;

var
  Form1: TForm1;

implementation

{$R *.fmx}

{ TForm1 }

procedure TForm1.GetFileListData;
begin

end;
```

追加

ソース5

GetFileListData（指定フォルダのファイル一覧を表示）

```
{*************************************************************************
 目的：ファイル一覧取得処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.GetFileListData;
begin
  // リストをクリア
  lvFileList.Items.Clear;
  cdsFileListClient.Close;

  // 参照対象のフォルダパスをパラメータに渡す
  svmGetFileList.ParamByName('ADIR').AsString := FCurrDir;

  // ファイル一覧を取得
  svmGetFileList.ExecuteMethod;
  cdsFileListClient.Open;

  // 画面にセット
  cdsFileListClient.First;
  while not cdsFileListClient.Eof do
  begin
    // フォルダの場合
    if cdsFileListClient.FieldByName('FLDRKB').AsInteger = 1 then
    begin
      // フォルダ名を表示
      lvFileList.Items.Add.Text :=
        cdsFileListClient.FieldByName('FLNAME').AsString;
    end
    // ファイルの場合
    else
    begin
      // ファイル名 + スペース + ファイルサイズ(KB)を表示
      lvFileList.Items.Add.Text :=
        cdsFileListClient.FieldByName('FLNAME').AsString + '    '
      + IntToStr(cdsFileListClient.FieldByName('FLSIZE').AsInteger) + 'KB';
    end;

    cdsFileListClient.Next;
  end;

end;
```

ソース6

OnItemClickEx（商品一覧の行選択時処理）

```
{*****************************************************************************
 目的：商品一覧選択時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.lvSHDataItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // 商品一覧のデータセットのレコード位置を同期
  cdsSHDataClient.RecNo := ItemIndex + 1;

  // 参照フォルダパスを保持
  // 固定参照先 + 商品コードのフォルダ
  FCurrDir :=
    '¥¥osknas02¥users¥osk前坂¥TecReport¥' +
    cdsSHDataClient.FieldByName('SYSHCD').AsString + '¥';

  // ファイル一覧を取得＆画面にセット
  GetFileListData;

  // ファイル一覧画面(タブ)を表示
  tbcMain.ActiveTab := tbiFileList;

  // 1行目を選択行に
  lvFileList.ItemIndex := 0;

  // 前画面に戻るボタンを表示
  btnBack.Visible := True;
end;
```

ソース7

OnClick（前画面へ戻るボタン処理）

```
{*****************************************************************************
 目的：前画面へ戻るボタン押下時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.btnBackClick(Sender: TObject);
begin
  if tbcMain.ActiveTab = tbiFileList then
  begin
    // 商品一覧を表示
    tbcMain.ActiveTab := tbiSHData;

    // 前画面へ戻るボタンを非表示
    btnBack.Visible := False;
  end;
end;
```

OnShow（画面表示時処理）

```
      + cdsSHDataClient.FieldByName('SYSHNM').AsString;

    cdsSHDataClient.Next;
  end;

  // 1行目を選択行に
  lvSHData.ItemIndex := 0;

  // 画面初期設定
  tbcMain.ActiveTab := tbiSHData;      // 商品一覧を表示
  btnBack.Visible   := False;          // 前画面へ戻るボタンを非表示
end;
```

追加

図16　ファイル一覧画面

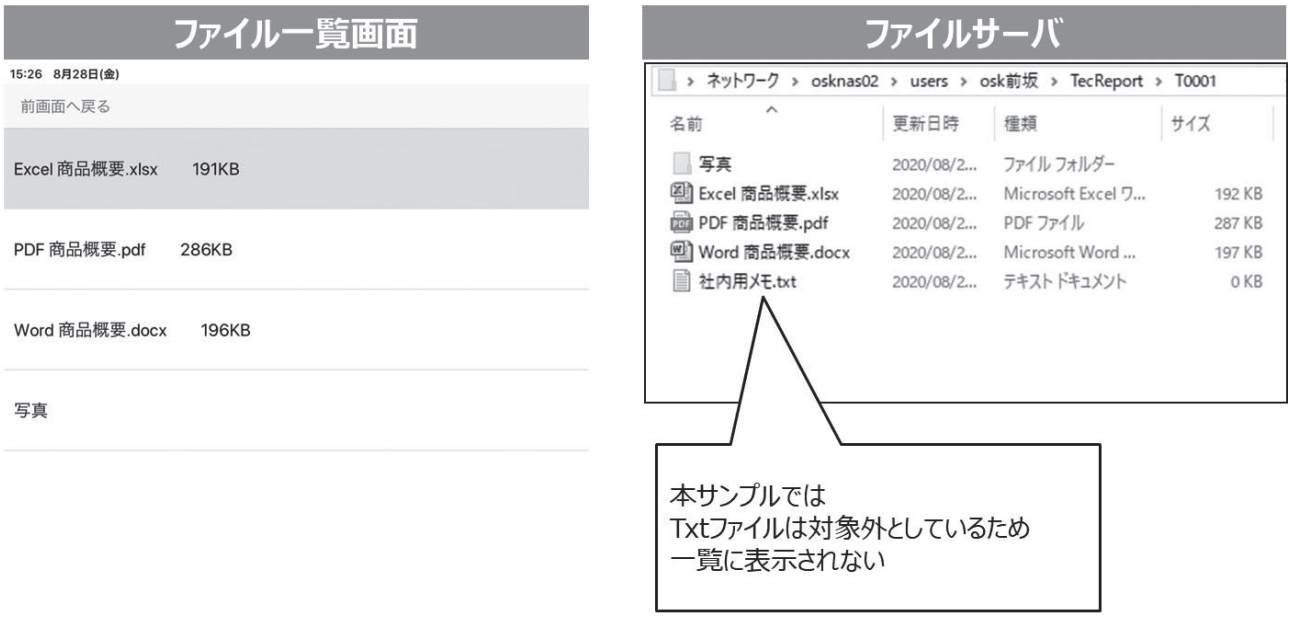


図17　ファイル内容表示の仕組み

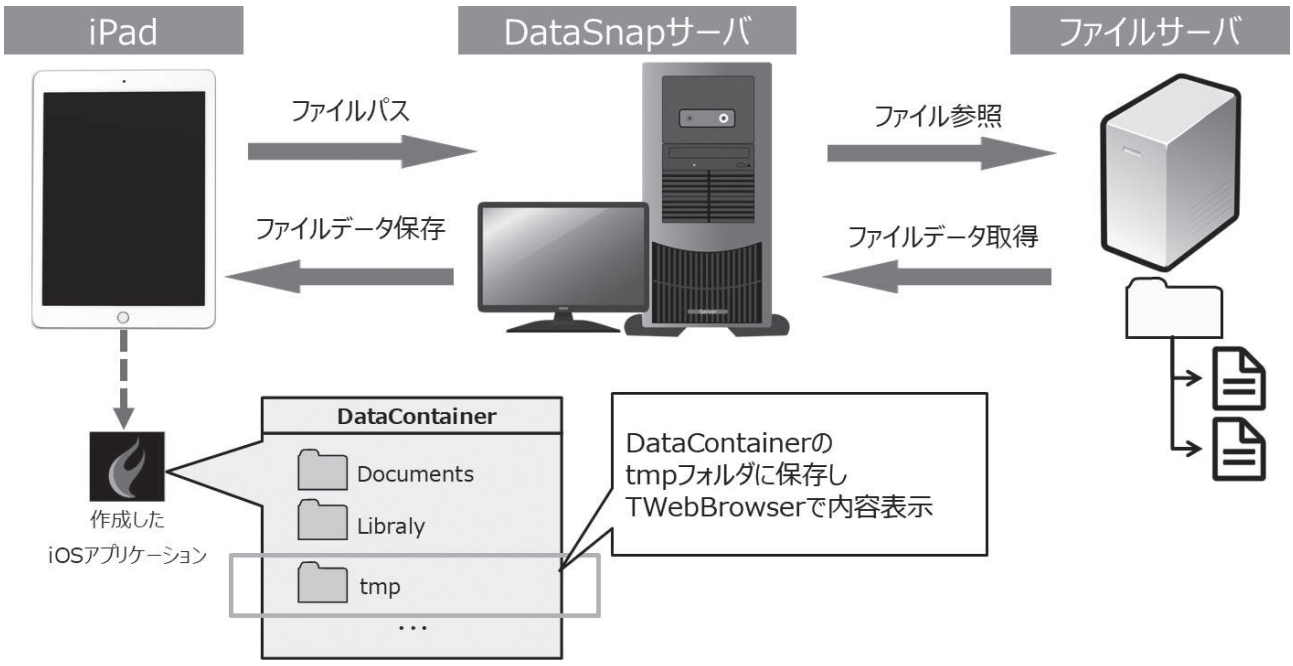
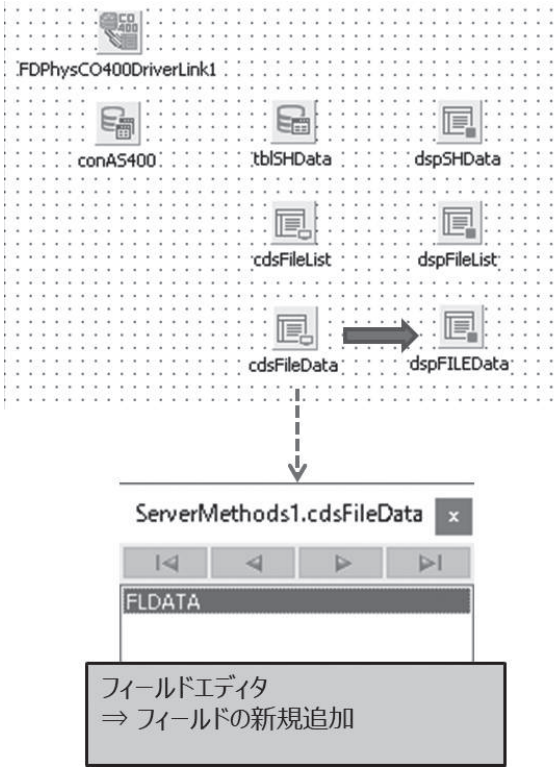


図18　コンポーネント設定値



cdsFileData : TClientDataSet
設定値変更なし

dspFileData : TDataSetProvider
DataSet=cdsFileData

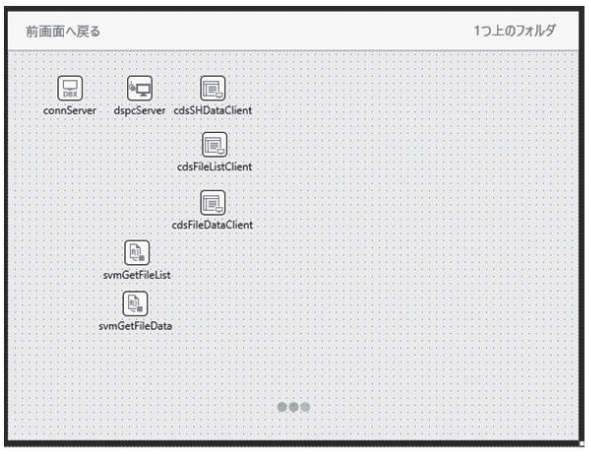
FLDATA : TBlobField
Size=1000000　※ 取得ファイルのファイルサイズに合わせて指定

ソース8

GetFileData（指定ファイルのデータを取得）

```
{*****************************************************************************
  目的：ファイルデータ取得処理
  引数：AFilePath - ファイルパス
  戻値：
*****************************************************************************}
procedure TServerMethods1.GetFileData(AFilePath: String);
begin

  // データセット準備
  cdsFileData.Close;
  cdsFileData.CreateDataSet;
  cdsFileData.EmptyDataSet;

  // ファイルが存在しない場合、処理中断
  if not System.SysUtils.FileExists(AFilePath) then
  begin
    Exit;
  end;

  // データセットにファイルを保持
  cdsFileData.Append;
  (cdsFileData.FieldByName('FLDATA') as TBlobField).LoadFromFile(AFilePath);
  cdsFileData.Post;

  // 先頭行
  cdsFileData.First;
end;
```

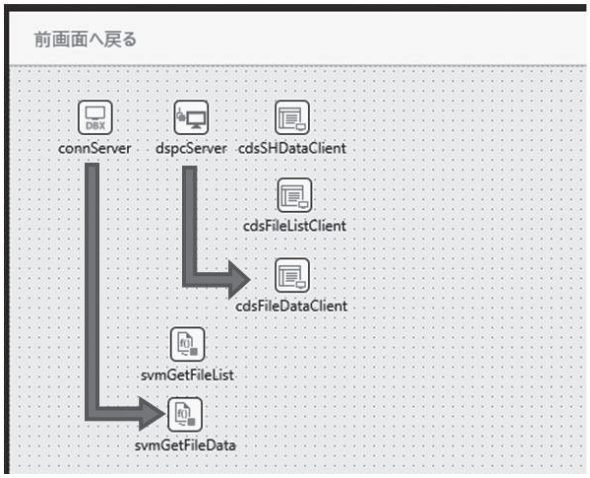
図19　コンポーネント設定値



tbiFileData : TTabItem
設定値変更なし

wbFileData : TWebBrowser
Align=Client

btnUpDir : TButton
Align=MostRight
Text="1つ上のフォルダ"



cdsFileDataClient : TClientDataSet
RemoteServer=dspcServer
ProviderName=dspFileData

svmGetFileData : TSqlServerMethods
SQLConnection=connServer
ServerMethodName=TServerMethods1.GetFileData

ソース9

uses節に追加

```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.TabControl,
  FMX.Controls.Presentation, FMX.StdCtrls, Data.DBXDataSnap, Data.DBXCommon,
  IPPeerClient, FMX.ListView.Types, FMX.ListView.Appearances,
  FMX.ListView.Adapters.Base, FMX.ListView, Data.DB, Datasnap.DBClient,
  Datasnap.DSConnect, Data.SqlExpr, Data.FMTBcd, FMX.WebBrowser,
  System.IOUtils;

type
  TForm1 = class(TForm)
```

追加

private宣言部に追加

```
private
  { private 宣言 }
  FCurrDir: String;                 // フォルダパスを保持

  FTmpDir: String;                  // 端末内保存用フォルダパス
  FRootDir: String;                 // 初期表示フォルダパス
  FDirPathCnt: Integer;             // 保存ファイル数カウント
  FBefoDir: TStringList;            // 順に選択したフォルダパスを保持

  procedure GetFileListData;        // ファイル一覧取得処理

  procedure GetFileDataDisp;        // ファイルデータ表示処理
```

追加

追加

FormCreate(画面生成時処理)

```
{*************************************************************************
 目的：画面生成時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.FormCreate(Sender: TObject);
begin
  // tmpフォルダのパス取得
  FTmpDir  := System.IOUtils.TPath.GetTempPath;

  // 内部保持用変数生成
  FBefoDir := TStringList.Create;
end;
```

OnItemClickEx(商品一覧選択時処理)

```
{*************************************************************************
 目的：商品一覧選択時処理
 引数：
 戻値：
*************************************************************************}
procedure TForm1.lvSHDataItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // 商品一覧のデータセットのレコード位置を同期
  cdsSHDataClient.RecNo := ItemIndex + 1;

  // 参照フォルダパスを保持
  // 固定参照先 + 商品コードのフォルダ
  FCurrDir :=
    '¥¥osknas02¥users¥osk前坂¥TecReport¥' +
    cdsSHDataClient.FieldByName('SYSHCD').AsStri + '¥';

  // 商品一覧選択時のパスを保持しておく
  FRootDir := FCurrDir;
```

追加

ソース11

GetFileDataDisp(ファイルデータ表示処理)

```
{*******************************************************************************
 目的: ファイルデータ表示処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.GetFileDataDisp;
var
  sFullFileName: String;
  sSaveFileName: String;
begin
  cdsFileDataClient.Close;

  // 表示対象のファイルパスをパラメータに渡す
  svmGetFileData.ParamByName('AFILEPATH').AsString :=
    cdsFileListClient.FieldByName('FLPATH').AsString;

  // ファイルデータを取得
  svmGetFileData.ExecuteMethod;
  cdsFileDataClient.Open;

  // 端末内保存用にファイル名を変更（取得ファイル名⇒連番値）
  // ※ 濁点のファイル名が正しく処理されないため
  Inc(FDirPathCnt);
  sSaveFileName :=
    ChangeFileExt(cdsFileListClient.FieldByName('FLPATH').AsString, '');
  sSaveFileName :=
    StringReplace(
      cdsFileListClient.FieldByName('FLPATH').AsString,
      sSaveFileName,
      FormatFloat('00000', FDirPathCnt), [rfReplaceAll]);

  // デバイスに保存
  // フォルダ作成
  ForceDirectories(
    TPath.Combine(FTmpDir, FormatFloat('00000', FDirPathCnt) + '/'));

  // ファイル保存
  sFullFileName :=
    TPath.Combine(
      FTmpDir, FormatFloat('00000', FDirPathCnt) + '/' + sSaveFileName);
  (cdsFileDataClient.FieldByName('FLDATA')
    as TBlobField).SaveToFile(sFullFileName);

  // ファイル内容の表示
  wbFileData.Navigate('file://' + sFullFileName);
end;
```

ソース12

OnItemClickEx（ファイル一覧の行選択時処理）

```
{*****************************************************************************
 目的：ファイル一覧選択時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.lvFileListItemClickEx(const Sender: TObject; ItemIndex: Integer;
  const LocalClickPos: TPointF; const ItemObject: TListItemDrawable);
begin
  // ファイル一覧のデータセットのレコード位置を同期
  cdsFileListClient.RecNo := ItemIndex + 1;

  // 選択行がフォルダの場合
  if cdsFileListClient.FieldByName('FLDRKB').AsInteger = 1 then
  begin
    //  1つ上のフォルダを保持
    FBefoDir.Add(FCurrDir);

    // カレントフォルダをセット
    FCurrDir :=
      FCurrDir + cdsFileListClient.FieldByName('FLNAME').AsString + '¥';

    // ファイル一覧再表示
    GetFileListData;
  end
  else
  // 選択行がファイルの場合
  begin
    // ファイル内容表示処理
    GetFileDataDisp;

    tbcMain.ActiveTab := tbiFileData;
  end;

  // 先頭フォルダでない場合、1つ上のフォルダボタンを表示
  btnUPDir.Visible := (FCurrDir <> FRootDir);
end;
```

ソース13

OnClick（1つ上のフォルダボタン押下時処理）

```
{*****************************************************************************
 目的：1つ上のフォルダボタン押下時処理
 引数：
 戻値：
*****************************************************************************}
procedure TForm1.btnUPDirClick(Sender: TObject);
begin
  // カレントフォルダに1つ上のフォルダパスをセット
  FCurrDir := FBefoDir[FBefoDir.Count - 1];

  // カレントフォルダにセットしたフォルダパスを削除
  FBefoDir.Delete(FBefoDir.Count - 1);

  // データ再表示
  GetFileListData;
end;
```

OnShow（画面表示時処理）

```
  // 1行目を選択行に
  lvSHData.ItemIndex := 0;

  // 画面初期設定
  tbcMain.ActiveTab := tbiSHData;      // 商品一覧を表示
  btnBack.Visible   := False;          // 前画面へ戻るボタンを非表示
  btnUPDir.Visible  := False;          // 1つ上のフォルダボタンを非表示

end;
```

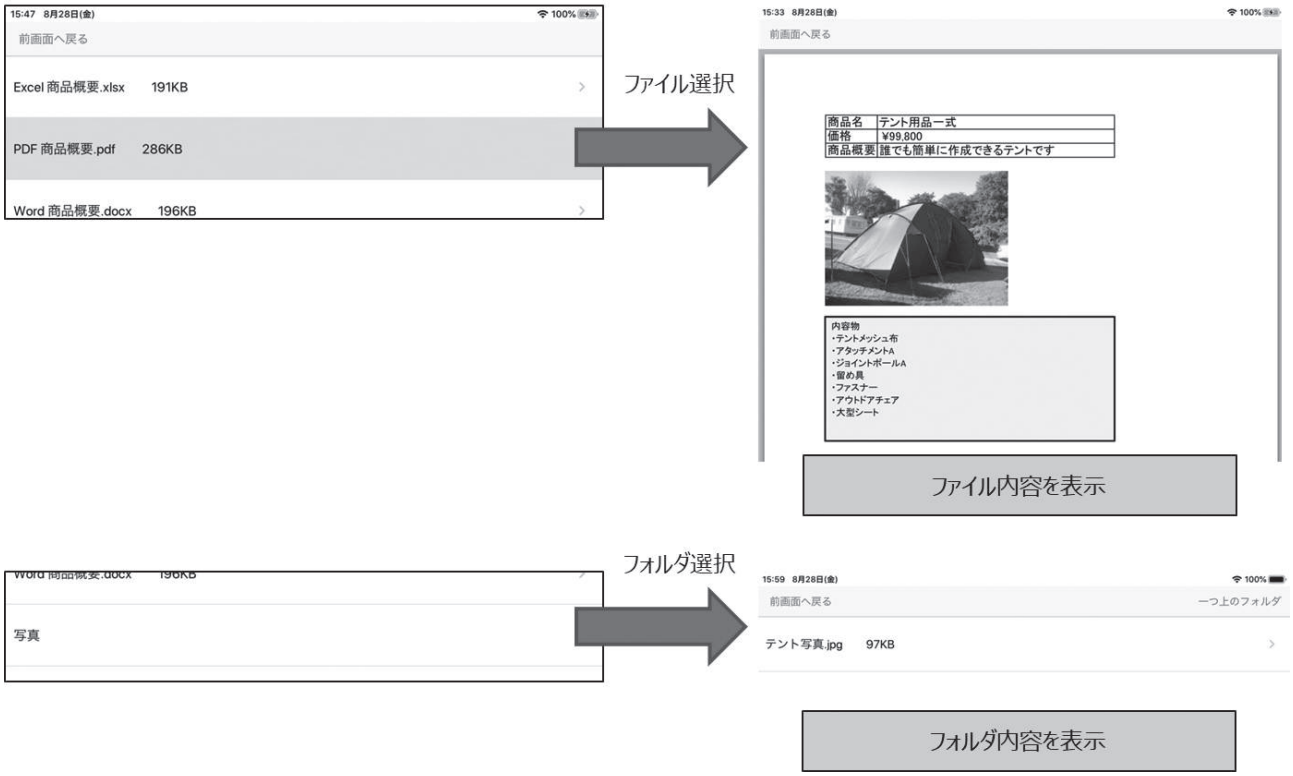
追加

ソース14

OnDestroy(画面破棄時処理)

```
{*****************************************************************************
  目的：画面破棄時処理
  引数：
  戻値：
*****************************************************************************}
procedure TForm1.FormDestroy(Sender: TObject);
var
  DirList: TStringDynArray;
  iLoop: Integer;
begin
  // 作成したtmpフォルダ以下のフォルダを削除
  DirList := TDirectory.GetDirectories(FTmpDir);
  for iLoop := 0 to Length(DirList) - 1 do
  begin
    TDirectory.Delete(DirList[iLoop], True);
  end;
end;
```

図20　ファイル閲覧画面



ソース15

private宣言部に追加

```
  private
    { private 宣言 }
    FCurrDir: String;                  // フォルダパスを保持

    FTmpDir: String;                   // 端末内保存用フォルダパス
    FRootDir: String;                  // 初期表示フォルダパス
    FDirPathCnt: Integer;              // 保存ファイル数カウント
    FBefoDir: TStringList;             // 順に選択したフォルダパスを保持

    FFullFileName: String;             // 表示ファイル名
    FPoint: TPointF;                   // タップ位置を保持

    procedure GetFileListData;         // ファイル一覧取得処理

    procedure GetFileDataDisp;         // ファイルデータ表示処理
  public
```

追加

GetFileDataDisp(ファイルデータ表示処理)

```
  (CustFileDataClient.FieldByName('FDATA') as TBlobField).SaveToFile(sFullFileNa

  // ファイル内容の表示
  wbFileData.Navigate('file://' + sFullFileName);

  // 内部保持
  FFullFileName := sFullFileName;
end;
```

追加

ソース16

uses節の追加

```
uses
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Va
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs, FMX.TabCo
  FMX.Controls.Presentation, FMX.StdCtrls, Data.DBXDataSnap, Data.DBXCommo
  IPPeerClient, FMX.ListView.Types, FMX.ListView.Appearances,
  FMX.ListView.Adapters.Base, FMX.ListView, Data.DB, Datasnap.DBClient,
  Datasnap.DSConnect, Data.SqlExpr, Data.FMTBcd, FMX.WebBrowser,
  System.IOUtils,
  iOSapi.Foundation, iOSapi.UIKit, Macapi.Helpers, FMX.Platform.Ios;
```

追加

OnTap、OnClick(ファイル保存ボタン押下時処理)

```
{*******************************************************************************
 目的: ファイル保存ボタン押下時処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.btnFileSaveClick(Sender: TObject);
var
  URL: NSURL;
  FController : UIDocumentInteractionController;
  iTop, iLeft, iBottom, iRight: Integer;
begin
  FController := TUIDocumentInteractionController.Wrap(
  TUIDocumentInteractionController.Alloc.init);

  // ファイルパスを渡す
  URL := TNSUrl.Wrap(TNSUrl.OCClass.fileURLWithPath(StrToNSStr(FFullFileName)));
  FController.setURL(URL);

  // ダイアログ表示位置指定
  iTop    := Trunc(FPoint.Y);
  iLeft   := Trunc(FPoint.X);
  iBottom := iTop  + 10;
  iRight  := iLeft + 5;

  FController.presentOptionsMenuFromRect(
      RectToNSRect(TRect.Create(iLeft, iTop, iRight, iBottom)),
      WindowHandleToPlatform(self.Handle).View,
      true);
end;

{*******************************************************************************
 目的: ファイル保存ボタンタップ時処理
 引数:
 戻値:
*******************************************************************************}
procedure TForm1.btnFileSaveTap(Sender: TObject; const Point: TPointF);
begin
  FPoint := Point;
end;
```

図21　ファイル保存


18:02　8月28日(金)
100%
前画面へ戻る
ファイル保存
00002
PDF書類 · 293 KB
商品名　テント用品一式
価格　¥99,800
商品概要　誰でも簡単に作成できるテ
Skitchにコピー
SmartLibrary
ブックにコピー
その他
コピー
マークアップ
プリント
Skitch
"ファイル"に保存
アクションを編集...
内容物
・テントメッシュ布
・アタッチメントA
・ジョイントポールA
・留め具
・ファスナー
・アウトドアチェア
・大型シート
指定したアプリ内に保存可能
任意のフォルダなどに保存可能