【セッションNo.2】

Delphi/400開発ノウハウお教えします
～ 現場で培った開発手法一挙公開～

株式会社ミガロ
システム事業部　システム2課
尾崎　浩司

・アジェンダ

1. Delphi/400で利用する基本的な開発手法のご紹介
   - ①QTEMP及びメンバーを使用したデータアクセス手法
   - ②クライアントデータセットを利用した画面開発手法
   - ③System　iの特徴を考慮したコンポーネント

2. Delphi/400連携活用事例
   - ①COMを利用した「駅すぱあと」情報の活用
   - ②COM作成によるExcel-VBAとの連携
   - ③WEBサービスを利用した為替情報の取得

3.まとめ

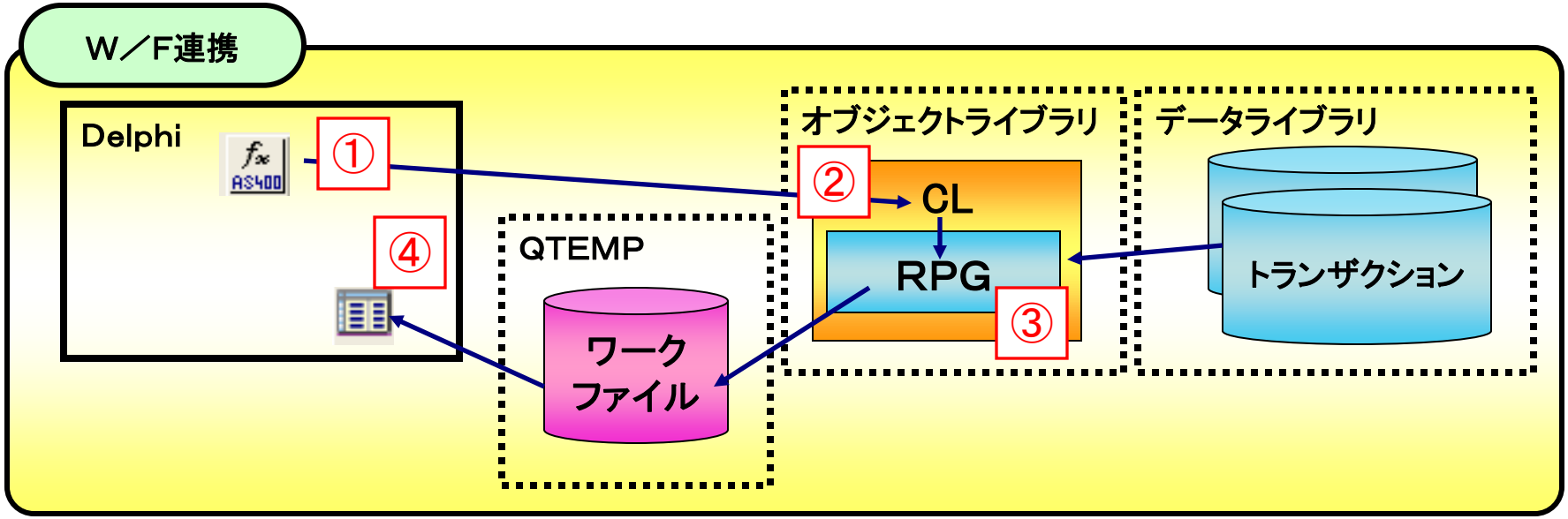
第1章

Delphi/400で利用する基本的な開発手法のご紹介

＜ データアクセス手法 ＞

- Delphi/400を使用したデータアクセスに関する開発手法
  - QTEMPを使用する方法
  - メンバーを使用する方法
    - CL/RPGと連携し、ワークファイル(W/F)に出力する手法
    - 処理結果のW/Fをセッション毎に識別する必要あり
  - SQLを使用する方法
    - Delphi単独で開発する手法

＜ W/Fを利用したデータアクセス ＞



■ アクセス手順

① Delphiより、TCall400コンポーネントを使用し、CLプログラムを起動
② CLプログラムにてワークファイルの初期化（作成、クリア）を行いRPGを起動
③ RPGプログラムにてトランザクションよりデータを取得し、ワークファイルへ出力
④ ワークファイルへの出力結果をTTableコンポーネントを使用し、Delphiで表示

＜ なぜワークファイルを使用するか ＞

- 既存のRPGロジックを**有効活用可能**
  - 従来のRPGプログラムが存在する場合、比較的容易に移行可能
- 処理が複雑であってもRPG処理のため、**高速化が期待**できる
  - ネイティブ言語であるRPGを使用できる
  - ビジネスロジックの変更に柔軟に対応可能
- ジャーナル環境でない場合の**信頼性向上**
  - Delphiは、実体のデータライブラリに直接アクセスしないため、異常終了等が発生しても、影響が少ない

- 逆に単純な照会処理等SQL利用が有効な場合は、SQLを使用
  - Delphi単体での開発が可能

＜ QTEMPかメンバーか ＞

- それぞれの特徴

	QTEMPを使用	メンバーを使用
ワークファイルの管理	容易 アプリ実行終了にて自動的に破棄	煩雑 メンバー名にてセッションを識別する仕組みが必要
保守性	煩雑 QTEMP内のワークファイルの内容を他のセッションから確認できない	容易 メンバー名の指定により他のセッションからワークファイルの内容を確認できる
高速性	中 QTEMP内にCPYF等で都度オブジェクトを生成する必要がある	速 既に存在するワークファイルオブジェクトにメンバー追加するのみでよい

- 選択基準
  - より簡易に連携する仕組みを構築したい場合
    ⇒ QTEMPを使用
  - より高速性、保守性を考慮した仕組みを構築したい場合
    ⇒ メンバーを使用

＜ QTEMP 具体例＞

- アクセス方法
  - TTable(テーブルコンポーネント)
    - TableNameプロパティ
      - Table1.TableName := 'QTEMP/FILE1';
  - TQuery(クエリーコンポーネント)
    - SQLプロパティ
      - Query1.SQL.Text := 'SELECT * FROM QTEMP/FILE1 FOR FETCH ONLY ';

- 開発時の留意点
  - 設計画面時、QTEMPにオブジェクトがないため、詳細な設計処理が行えない
    - データライブラリにオブジェクトを配置することにより設計可能にする
  - 実行時、QTEMP内のデータに他のジョブからアクセスできない
    - AS400コンポーネントを使用し別ライブラリにコピーする処理を組み込む
      - AS400.RemoteCmd('CPYF FROMFILE(QTEMP/FILE1) TOFILE(DATLIB/FILE1) MBROPT(*REPLACE) CRTFILE(*YES)');

- QTEMP作成例 （CL）

```
#START:      PGM

/*オブジェクト確認*/
             CHKOBJ      OBJ(QTEMP/WMSSG) OBJTYPE(*FILE)
             MONMSG MSGID(CPF9801) EXEC(GOTO CMDLBL(#CPYTB1))
             GOTO        CMDLBL(#CLRTB1)
/*テーブル作成*/
#CPYTB1:
             CPYF        FROMFILE(*LIBL/WMSSG) TOFILE(QTEMP/WMSSG) +
                           CRTFILE(*YES)
             MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(#PGMERR))
/*テーブルクリア*/
#CLRTB1:
             CLRPFM      FILE(QTEMP/WMSSG)
             MONMSG MSGID(CPF0000) EXEC(GOTO CMDLBL(#PGMERR))
/*正常終了*/
             RETURN
/*エラー終了*/
#PGMERR:
             ENDPGM
```

QTEMP上のオブジェクトチェック

オブジェクトが存在しない場合QTEMPに作成

QTEMP中のファイルを初期化

＜メンバー 具体例＞

- アクセス方法
  - TTable(テーブルコンポーネント)
    - TableNameプロパティ
      - Table1.TableName := 'LIBRARY1/FILE1(MEMBER1)';
  - TQuery(クエリーコンポーネント)
    - メンバー名を指定したクエリーの記述不可

- 開発時の留意点
  - メンバーを指定したSQLが記載できない。
    - OVRDBFを使用し一つのオブジェクトとして処理できるようにする
  - メンバーの管理が煩雑
    - ジョブ名をメンバー名とすることにより、一意に扱えるようにする

- 初期処理例 （CL）

```
PGM          PARM(&JOBNM)

DCL VAR(&JOBNM)   TYPE(*CHAR) LEN(10) /*ＪＯＢ名       */

START:
   /********** ライブラリーリスト追加 ***************/

            ADDLIBLE    LIB(DATLIB)       POSITION(*LAST)
            MONMSG      MSGID(CPF0000)
            ADDLIBLE    LIB(OBJLIB)       POSITION(*LAST)
            MONMSG      MSGID(CPF0000)

   /********** ＪＯＢ名取得 *************************/
            RTVJOBA     JOB(&JOBNM)

/*プログラム終了*/
END:
            ENDPGM
```

CLパラメータにJOB名を持ち、クライアント側に返せるようにする

ライブラリリストの追加

JOB名の取得

- メンバー処理 （CL）

```
/*********物理ファイルメンバー追加＆クリア********/
          IF   COND(&SHORI = '1') THEN(DO)
               /* 物理ファイルメンバー追加*/
               ADDPFM       FILE(FILE_A) MBR(&JOBNM)
               MONMSG       MSGID(CPF0000)
               /* 物理ファイルメンバークリア*/
               CLRPFM       FILE(FILE_A) MBR(&JOBNM)
               MONMSG       MSGID(CPF0000)
          ENDDO

/*********物理ファイルメンバー削除***************/
          IF   COND(&SHORI = '2') THEN(DO)
               /* 物理ファイルメンバー削除*/
               RMVM         FILE(FILE_A) MBR(&JOBNM)
               MONMSG       MSGID(CPF0000)
          ENDDO
```

メンバーの追加

クリア処理

メンバーの削除

- メンバー処理 （CLでのOVRDBF）

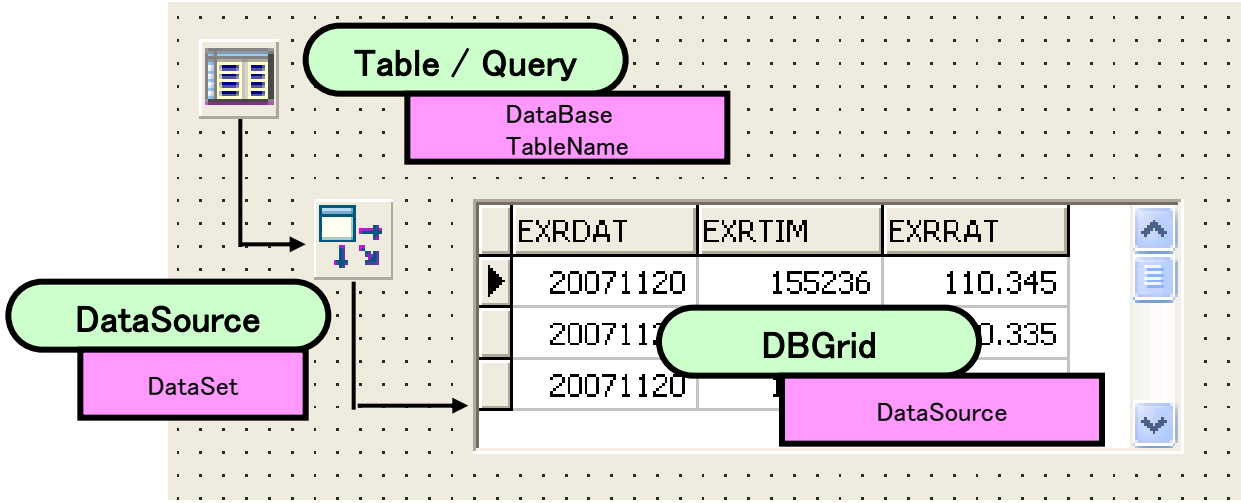
```
PGM         PARM(&FILE &JOBNM &SHORI)

            DCL VAR(&FILE)  TYPE(*CHAR) LEN(10)  /*ファイル      */
            DCL VAR(&JOBNM) TYPE(*CHAR) LEN(10)  /*メンバー      */
            DCL VAR(&SHORI) TYPE(*CHAR) LEN(01)  /*処理モード    */

            IF  COND(&SHORI = '1') THEN(DO)
                OVRDBF     FILE(&FILE) TOFILE(&FILE) MBR(&JOBNM) +
                           LVLCHK(*NO) OVRSCOPE(*JOB)
                MONMSG     MSGID(CPF0000)
            ENDDO

            IF  COND(&SHORI = '2') THEN(DO)
                DLTOVR     FILE(&FILE) LVL(*JOB)
                MONMSG     MSGID(CPF0000)
            ENDDO
/*プログラム終了*/
END:
            ENDPGM
```

OVRDBF処理

DLTOVR処理

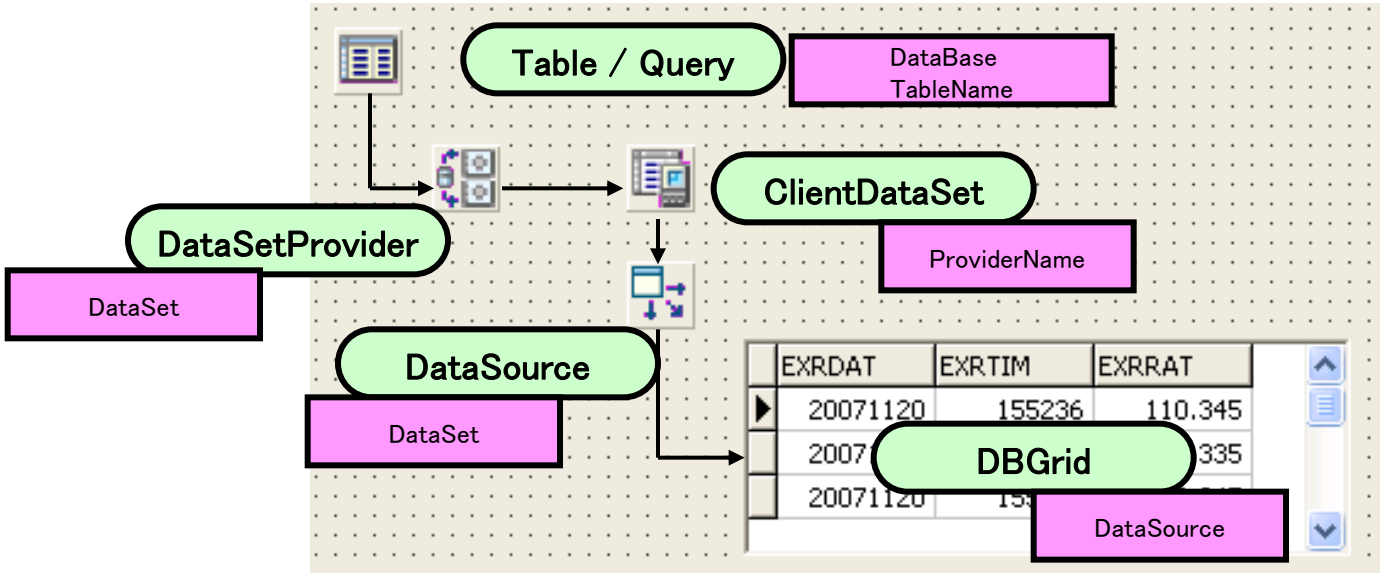
＜画面開発手法＞

- データアクセス結果を処理する手法
  - TTable / TQuery を使用する方法
    - BDEを使用する一般的なデータセット利用法



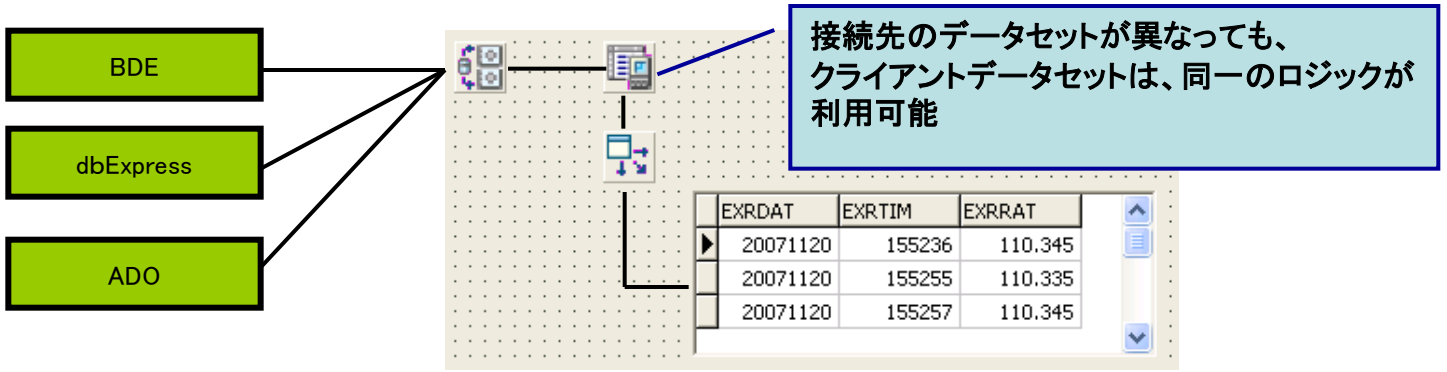
＜画面開発手法＞

- データアクセス結果を処理する手法
  - TClientDataSetを使用する方法
    - クライアントのメモリー上に保管するデータセットを利用する方法



＜ クライアントデータセットの使用 ＞

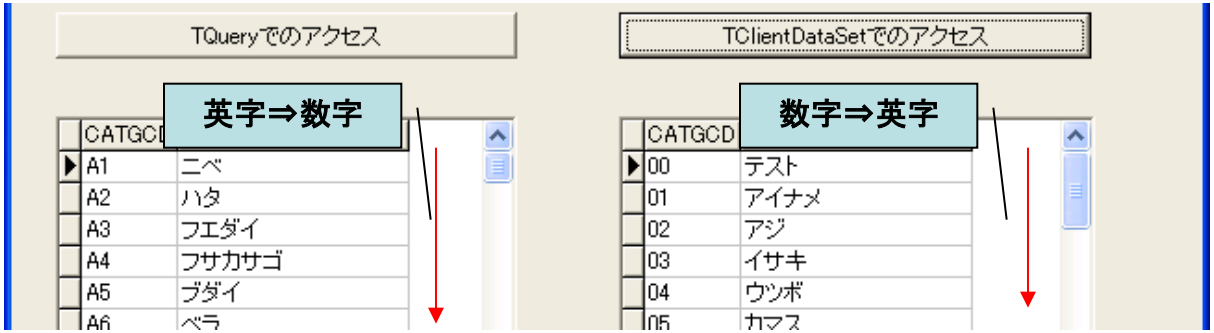
- 画面出力方法として、クライアントデータセットを利用
  - メリット
    - BDE、dbExpress等接続手法が異なる場合でもクライアント上のデータセットは、同様の仕組みで開発可能



    - データの出力件数にあわせたスクロール制御が可能
    - データがメモリに格納される為、クライアント上での容易な並べ替え、集計等が可能

＜ クライアントデータセットの使用 ＞

- クライアントデータセット使用の留意点
  - データ件数が多い場合、クライアント上に大量のデータを取り込む為に出力に時間がかかる
    - ClientDataSetのPacketRecordsプロパティの設定により一度に読み込む件数の調整が可能
      - 必要に応じて自動的に再読み込みを行う
  - データのソート順がクライアントPCの順序（数字⇒英文字）になってしまう



- DataSetProviderのOptionsプロパティにあるpoRetainServerOrderプロパティをTrueに変えることにより対応可能

＜画面開発手法＞

- System iをデータベースとするDelphi/400の開発において、文字列の扱いについては考慮が必要。
  - シフト文字(0E0F)の考慮が必要
  - 全角のみの入力フィールド(Jフィールド)の考慮が必要
  - CCSIDによっては、半角英小文字の使用が不可能なことに関する考慮が必要

- PC5250の操作性を考慮すると、Tabキーだけでなく、Enterキーでの項目移動も考慮したほうが良いことが多い

⇒ 上記を考慮した**画面コンポーネント(Edit)を開発**して使用

＜TMGRMaskEditコンポーネント＞

- 標準のTMaskEditに対し下記機能拡張を実施。
  - Alignmentプロパティ: 文字列の横方向の配置を指定
    - taLeftJustify – 左寄せ
    - taRightJustify – 右寄せ
    - taCenter – 中央寄せ
  - CharSetプロパティ: 入力文字列の属性指定
    - dmNone – 属性指定なし
    - dmSBCSOnly – 半角文字列のみ入力可能
    - dmDBCSOnly – 全角文字列のみ入力可能
  - EnterNextプロパティ: Enterキー押下による項目移動の設定
  - PageCodeプロパティ: CCSIDにあわせた文字列属性を指定
    - JP_1 – CCSID=5026（半角英小文字使用不可）
    - JP_2 – CCSID=5035（半角英小文字使用可）
  - MaxLengthプロパティ: シフト文字を含む文字長の指定

第2章

Delphi/400連携活用事例

＜ Delphi/400の活用事例 ＞

- Delphi/400は、完全ネイティブなWindowsアプリケーションが開発可能
  - Delphi/400単体での開発だけでなく、いろいろなツール/ソリューションとの連携が可能

- 活用事例
  - COMを利用した「駅すぱあと」情報の活用
  - COMオブジェクトの作成によるエクセルとの連携
  - WEBサービスを使用した為替情報の活用

＜ COMとは ＞

- COMとは？
  - Common Object Model
    - ソフトウェアの再利用のための技術
      - 他のアプリ等と連携するために必要なインターフェース（メソッド、プロパティ等）を提供する

- 身近なところでは・・・
  - PowerPointの資料中にExcelの表を挿入する

No	データ
1	COMの利用
2	COMの作成
3	WEBサービスの利用

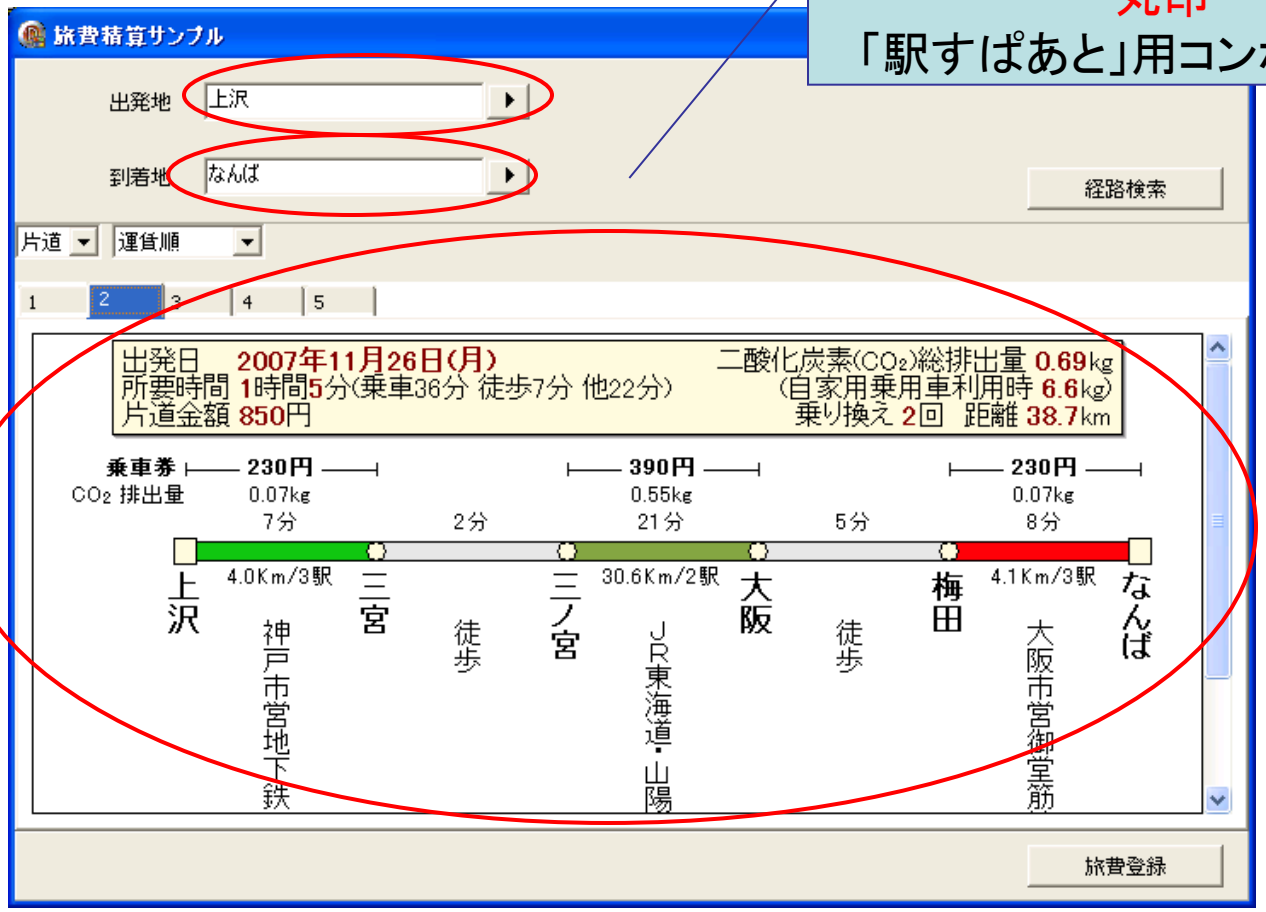
- 開発言語に左右されず作成・利用が可能
  - 他の開発ツール ⇒ Delphi
  - Delphi ⇒ Excel

＜「駅すぱあと」情報の活用 （COM利用）＞

- 「駅すぱあとSDK」
  - http://ekiworld.net/service/package/sdk/index.html
  - COMコンポーネントを提供
  - Delphiより利用可能

- 目的
  - 出張精算システムにおける簡略化
    - 「駅すぱあと」で取得できる路線情報をシステムに取り込んで入力を簡素化したい
    - 申請のチェックを容易にしたい

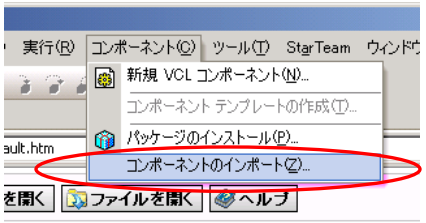
画面例



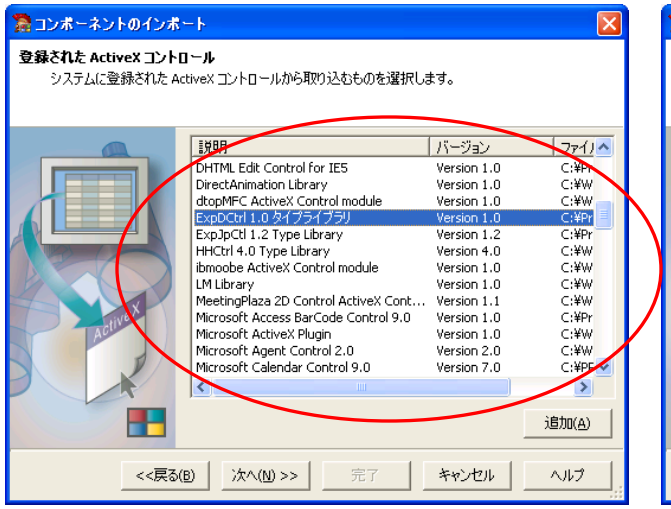
丸印
「駅すぱあと」用コンポーネント

■ 設定手順

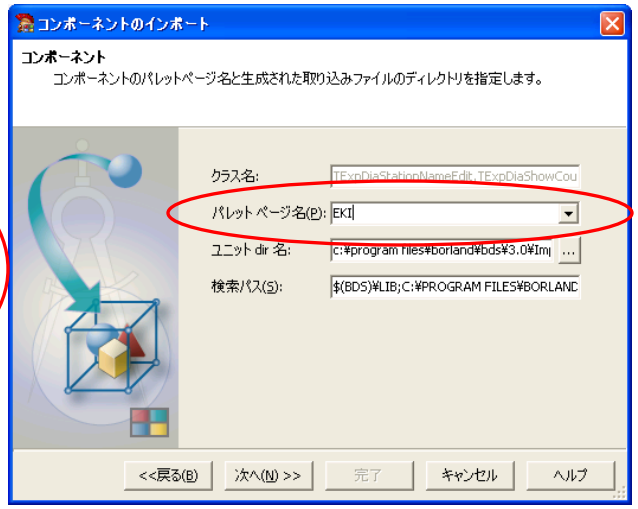
① メニューより「コンポーネント」→「コンポーネントのインポート」を選択（図1）
② 「ActiveXコントロールの取り込み」を選択
③ 取り込みしたいActiveXコントロールを選択（図2）
④ 「パレットページ名」を指定（図3）
⑤ 「ユニットの作成」にチェックをつけて、「完了」ボタンを押下
⑥ パッケージファイルを開き、⑤にて作成したユニットを追加し、「インストール」を実行



[図1]



[図2]



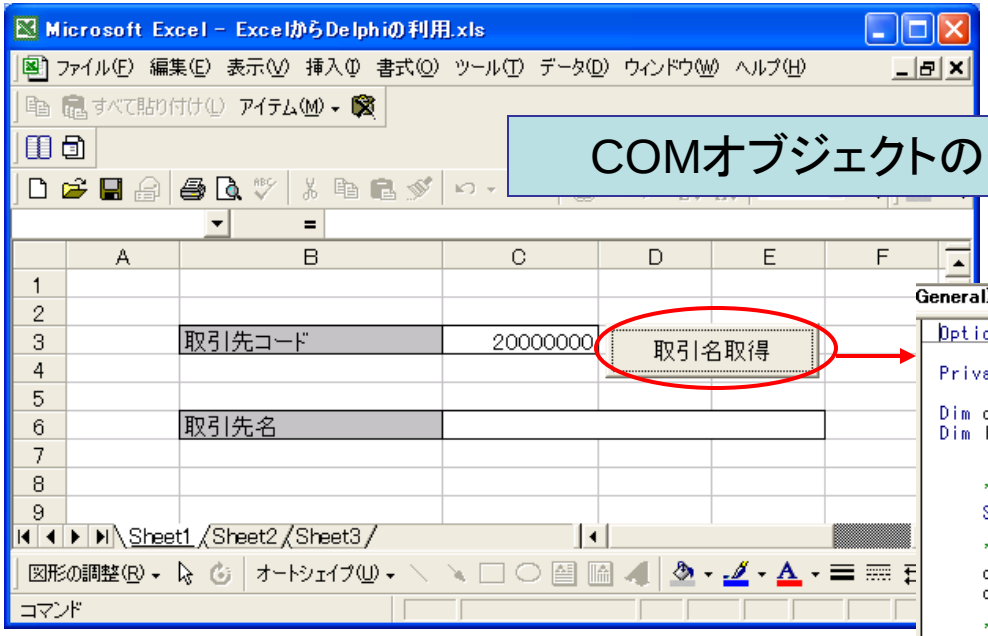
[図3]

＜ Excelとの連携　（COMの作成）＞

- ExcelからDelphi/400を呼び出す
  - COMオブジェクトをDelphiで開発

- 目的
  - エクセルのシート上にSystem iから情報を取得した結果を反映
    - エクセルをユーザーインターフェースとした画面構築により、ユーザーレベルでの変更を容易にしたい

■ 画面例 （作成したオブジェクトのExcelからの使用例）



COMオブジェクトの作成

```
General)                                    (Declarations)
Option Explicit

Private Sub CommandButton1_Click()

Dim objD4TecObj As Object '---- 取引先情報取得オブジェクト
Dim lngMCTRCD As Long     '---- 取引先コード


    ' 取引先情報取得オブジェクトの生成
    Set objD4TecObj = CreateObject("D4TecObj.D4TecObject")

    'AS/400接続情報のセット
    objD4TecObj.UserName = ActiveSheet.Range("C13").Value  ' サインオンユーザー名
    objD4TecObj.PassWord = ActiveSheet.Range("C14").Value  ' サインオンパスワード

    '取引先コードのセット
    lngMCTRCD = ActiveSheet.Range("C3").Value
    objD4TecObj.MCTRCD = lngMCTRCD

    'データの取得(メソッドの実行)
    objD4TecObj.GetData

    '取得した取引先名のセット
    ActiveSheet.Range("C6").Value = objD4TecObj.MCTRNM

    ' オブジェクトの開放
    Set objD4TecObj = Nothing

End Sub
```

プロパティの利用

メソッドの利用

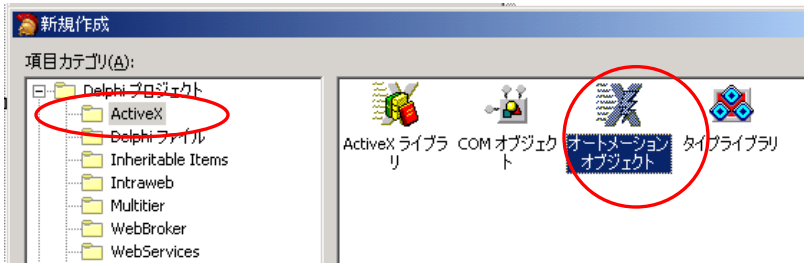
■ 設定手順

① 新規プロジェクトを作成
② メニューより「ファイル」→「新規作成」→「その他」を選択
③ 選択カテゴリ「ActiveX」から「オートメーションオブジェクト」を選択（図1）
④ CoClass名欄にオブジェクト名を指定
⑤ 必要に応じて「プロパティ」「メソッド」を追加（図2）
「プロパティ」の場合、属性を「タイプ」に指定
例） 数字型の場合“Long”、文字列型の場合“BSTR”を指定
⑥ 生成されたソースにユーザーロジックを組み込む
⑦ 画面を持たないプログラムの場合、プロジェクトファイルに下記行を追加

```
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.ShowMainForm := False;  // 行追加
Application.Run;
```

⑧ 実行時、メニューより「実行」→「実行時引数」を選択する。「パラメータ」欄にレジストリ登録する場合は“/regserver”を、レジストリ解除する場合は“/unregserver”を指定

[図1]



[図2]

＜ WEBサービスとは ＞

- WEBサービスとは？
  - XML形式のプロトコルを利用したメッセージ送受信の技術を利用したサービス

- 身近なところでは・・・
  - WEBで商品を調べると、そこから購入情報にアクセスできる
    - Amazon Webサービス
      - http://www.amazon.co.jp/gp/feature.html?docId=451209

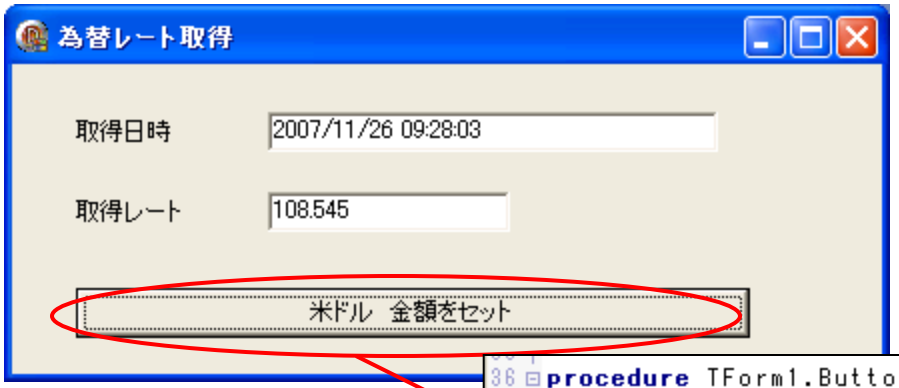
- ビジネスで利用できる情報もWEBサービスとして提供されている。

＜ 為替情報の取得 （WEBサービスの利用）＞

- Webservicex.Net
  - http://www.webservicex.net/WCF/webServices.aspx
  - ビジネス等でも使用可能なWEBサービスが登録
    - 為替情報、天気情報 ・・・

- 目的
  - 為替情報の自動登録
    - 日々変化する為替情報を都度手入力せずとも、システムに反映したい

■ 画面例



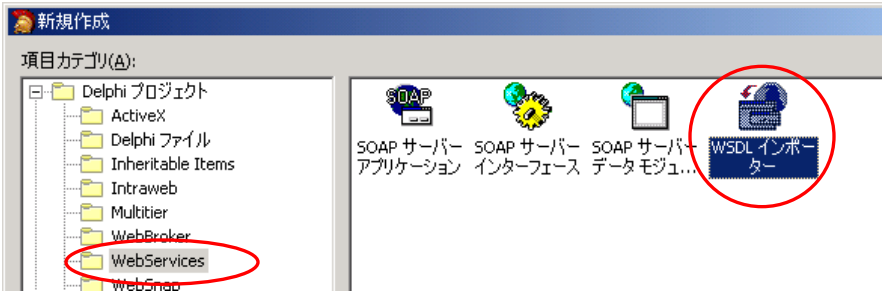
SOAPオブジェクト生成

USD ⇒ JPN のレート取得

```
36 procedure TForm1.Button1Click(Sender: TObject);
37 var
38   rw: CurrencyConvertorSoap;
39   cRate: System.Currency;     // ---- 通常のCurrency型はSystemユニットに定義
40   dDate: TDateTime;
41 begin
42   // 通貨情報を取得
43   rw := GetCurrencyConvertorSoap();
44   cRate := rw.ConversionRate(USD, JPY);
45   dDate := Now;
46
47   // 取得結果を画面表示
48   edtDate.Text := FormatDateTime('YYYY/MM/DD HH:NN:SS', dDate);
49   edtRate.Text := CurrToStr(cRate);
50
51   // 取得結果をデータベースに登録
52   with qryInsertRate do
53   begin
54     ParamByName('EXRDAT').AsInteger := StrToInt(FormatDateTime('YYYYMMDD', dDate));
55     ParamByName('EXRTIM').AsInteger := StrToInt(FormatDateTime('HHNNSS', dDate));
56     ParamByName('EXRRAT').AsCurrency := cRate;
57     ExecSQL;
```

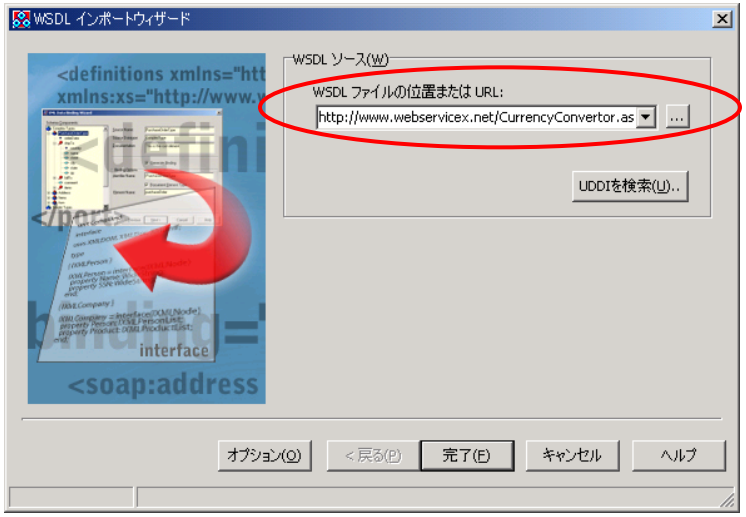
設定手順

① 新規プロジェクトを作成
② メニューより「ファイル」→「新規作成」→「その他」を選択
③ 選択カテゴリ「WebServices」から「WDSLインポーター」を選択（図1）
④ WDSLファイルをあらわすURLを入力（図2）
⑤ 「完了」ボタン押下
⑥ WEBサービスを使用するフォームのuses節に生成ユニットを追加



[図1]

[図2]



第3章

まとめ

＜ まとめ ＞

- **データアクセス手法**
  - 処理の複雑さにあわせて、SQLを使用する方法 と W/Fを使用する方法　とを使い分けます。
  - W/Fは、できるだけ簡易にRPG連携を実現する場合はQTEMPを、メンテナンス性・レスポンスを重視する場合メンバーを使用します。
- **クライアントデータセットの利用**
  - 使用するデータセットにかかわらず、クライアントは同じ仕組みで構築が可能です。
  - System iを意識したソート順の考慮が必要です。
- **コンポーネントの活用**
  - System iを考慮したコンポーネントを使用することにより、文字列の扱いを容易にしています。
- **Delphi/400とツール・ソリューションとの連携**
  - COM、WEBサービス等の技術を積極的に利用することにより、アプリケーション開発の幅を広げることが可能です。