

【セッションNo. 4】

プログラミングテクニックセッション

ワンランク上の開発手法

プログラムメンテナンス性を高めるひと工夫

株式会社ミガロ.

RAD事業部 営業・営業推進課

尾崎 浩司

# 【アジェンダ】

1. はじめに
2. 画面と機能を分離する開発手法
3. BDEからdbExpressへの移行手順
4. 将来のマルチデバイス対応を考慮した開発手法
5. まとめ

# 1. はじめに

## ■ 業務システム運用について

- システム運用開始後も様々なメンテナンスが発生
  - **業務の改善や法改正に伴う修正対応**
    - 設計・開発時点では気が付かなかった項目の追加
    - 使い勝手を良くするための画面の変更
    - 業務手順変更に伴うシステム改善
    - 消費税対応（2014年4月8%対応 → 今後10%対応〔軽減税率の行方？〕）etc....
  - **環境の変化に伴うバージョンアップ対応**
    - OSバージョンアップ対応（例：WindowsXP → Windows7/8/8.1）
    - 新しいデバイスへの対応検討（例：iPad 用機能の追加開発）etc....

## ■ 消費税率の変化

- 1997年以降17年間5%であった税率が8%に変更

The screenshot shows a software window titled 'テクニカルセミナーサンプル' with a sub-window '売上入力 新規'. It contains various input fields for sales data. A red box highlights the summary section with fields for '売上合計', '消費税額', and '税込金額'. A blue arrow points from a smaller version of this summary box in the lower-left area to the larger one in the upper-right area. Below the screenshot, a code block shows the calculation logic for tax and total amount.

//消費税額の計算

Tax.Value := Trunc(SalesAmount.Value \* 0.05);

//税込金額の計算

TotalAmount.Value := SalesAmount.Value + Tax.Value;

プログラムの中に消費税率(5%)がハードコードされている！

→ 税率を8%に変更する場合、全プログラムからハードコード部分を全て洗い出し、修正していく必要あり。メンテナンス性の考慮が不足。

機能を「サブルーチン化」すれば、メンテナンス効率が向上！

# ■ Windows クライアントOS環境の変化

- WindowsXPの正式サポートが2014年4月終了
  - Windows7、8、8.1への移行が急速に進行
  - BDEアプリケーションをそのまま実行すると、エラー発生の可能性

The screenshot shows a Windows XP desktop with a taskbar containing icons for 'ごみ箱', 'Internet Explorer', 'Adobe Reader 9', 'Windows Media Player', and 'WORK'. A window titled 'テクニカルセミナー' is open, displaying a table with columns '製品NO', '製品名', '板厚', '幅', and '長さ'. The table contains 8 rows of data. A blue arrow points from this window to a smaller version of the same window in the background. In the foreground, an error dialog box titled 'Bdesample' is displayed with a red 'X' icon. The error message reads: 'ネットワークの初期化に失敗しました。ファイルまたはディレクトリが存在しません。ファイル: C:\PDOXUSRS.NET アクセスは拒否されました。ディレクトリ: C:\。' with an 'OK' button at the bottom.

製品NO	製品名	板厚	幅	長さ
SNO01	ミガロ.製品01	10	100	5
SNO02	ミガロ.製品02	10	100	10
SNO03	ミガロ.製品03	10	100	15
SNO04	ミガロ.製品04	10	120	5
SNO05	ミガロ.製品05	10	120	10
SNO06	ミガロ.製品06	10	150	5
SNO07	ミガロ.製品07	10	150	10
SNO08	ミガロ.製品08	20	100	5

エラーを出さない為にはUACを切らないといけない！

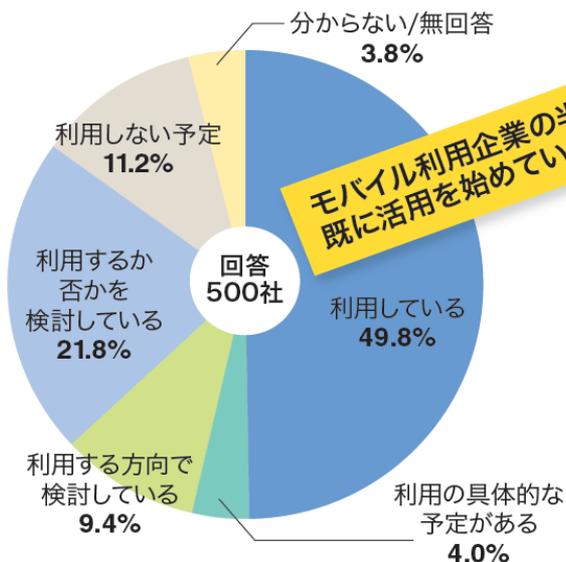
WindowsOSバージョンアップ時には、「BDE」を継続するか、「dbExpress」に置き換えるかの検討が必要！

# ■ 企業におけるコンピューター利用形態の変化

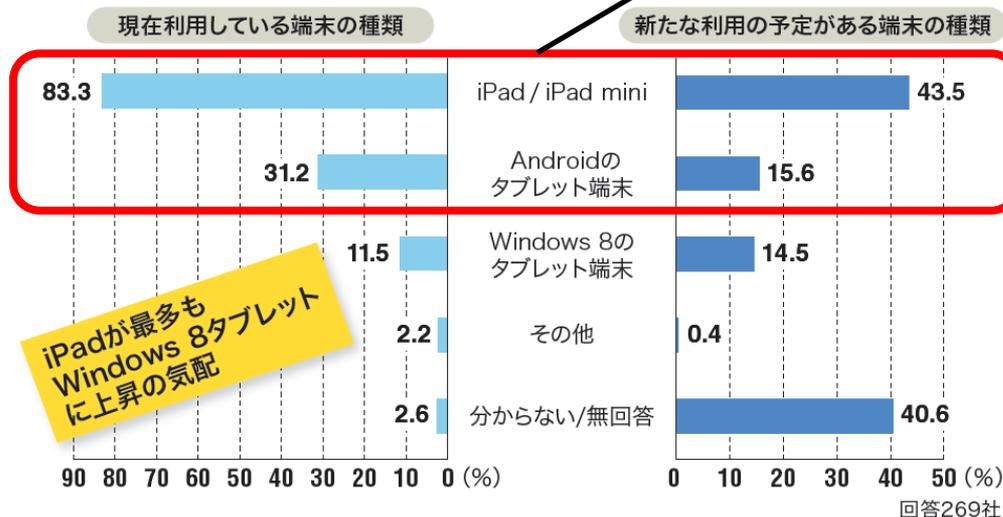
- ノートPCの代替としてタブレットを活用する企業が増加
  - これまでは、Windowsだけ対応すれば良かったが、今後はiOSやAndroid等Windows以外のクライアント端末の利用も検討が必要になる可能性大！

## モバイル通信導入企業のタブレット利用状況

iPad/Android選定企業が多い！



モバイル利用企業の半数が既に活用を始めている



iPadが最もWindows 8タブレットに上昇の気配

出典:ITpro Active [企業ネット実態調査2013]

Delphi/400アプリケーションの多層化を検討すれば、マルチデバイス対応が必要になった時に、最小限のコストで実現可能！

## ■ 今回のポイント

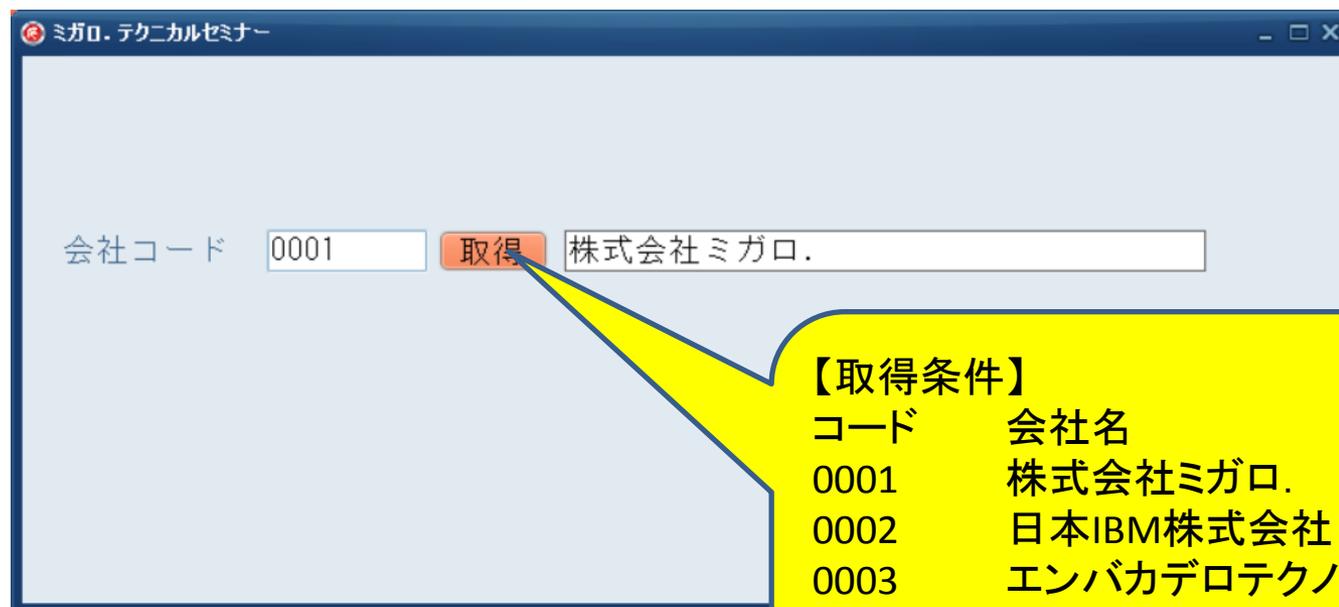
- アプリケーションメンテナンスに関する3つのトピックスをご紹介します！
  - **画面と機能を分離する開発手法**
    - メンテナンス性を向上させるサブルーチン作成方法をご紹介します！
  - **BDEからdbExpressへの移行手順**
    - 既存のBDEアプリケーションをWindows7以降に正式対応するdbExpressへ移行する方法をご紹介します！
  - **将来のマルチデバイス対応を考慮した開発手法**
    - 既存のC/Sアプリケーション機能を活用した多層化手順をご紹介します！

## 2. 画面と機能を分離する開発手法

## ■ サンプルアプリケーション

### • プログラム仕様

- 入力した会社コードで、会社名を取得して表示してほしい！

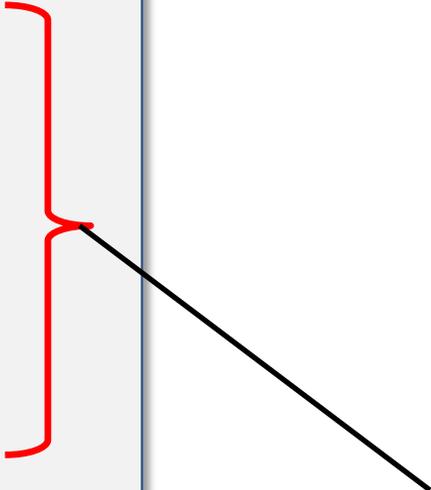


上記以外はエラー

## ■ サンプルアプリケーション

- 取得ボタン[btnGetComp]のクリック処理
  - 画面上の会社コードの値に応じて名称をセット

```
procedure TfrmMain.btnGetCompClick(Sender: TObject);
begin
  //会社コードにより名称を取得
  if edtCompCD.Text = '0001' then
    edtCompNM.Text := '株式会社ミガロ.'
  else if edtCompCD.Text = '0002' then
    edtCompNM.Text := '日本IBM株式会社'
  else if edtCompCD.Text = '0003' then
    edtCompNM.Text := 'エンバカデロテクノロジーズ合同会社'
  else
    raise Exception.Create('会社コードが正しくありません');
end;
```

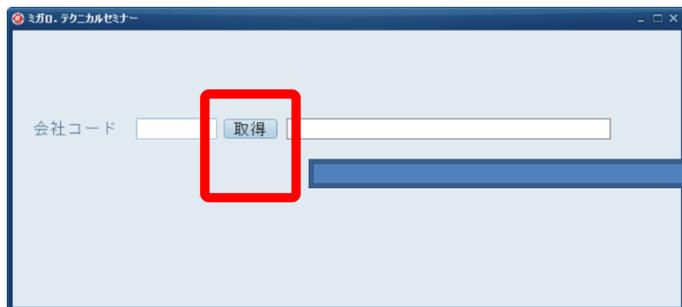


会社コード欄(edtCompCD)をそのまま、条件式に使用し、結果を会社名欄(edtCompNM)にセットしている。

→ GUI部品(画面)と、ロジック(機能)が混在したプログラムとなっている。

# ■ プログラムの修正が必要な場合

- 会社コード入力欄が、システム内に複数箇所あると...
  - ソースコードを1か所修正して、後はひたすら「コピペ」...



```
[SR *.dfm]
28
29
30 procedure TfrmMain.btnGetCompClick(Sender: TObject)
begin
//会社コードにより名称を取得
if edtCompCD.Text = '0001' then
edtCompNM.Text := '株式会社ミガロ'
else if edtCompCD.Text = '0002' then
edtCompNM.Text := '日本IBM株式会社'
else if edtCompCD.Text = '0003' then
edtCompNM.Text := 'エンパカテロテクノロジーズ合同会社'
else if edtCompCD.Text = '0004' then
edtCompNM.Text := '山田商店株式会社'
else
raise Exception.Create('会社コードが正しくありません');
end;
end.
```

ソースを修正

修正したソースを各ユニットに適用

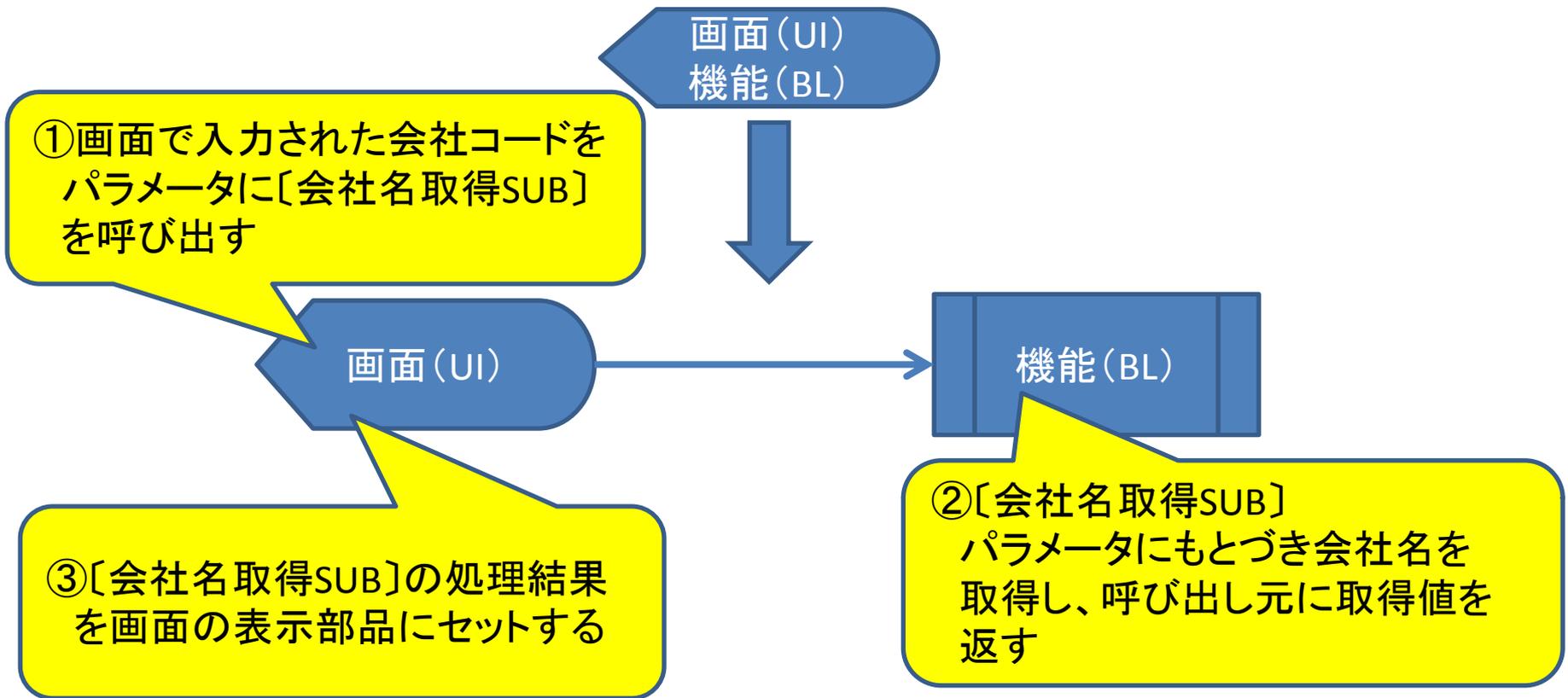


適用漏れがあるとバグ発生の原因!



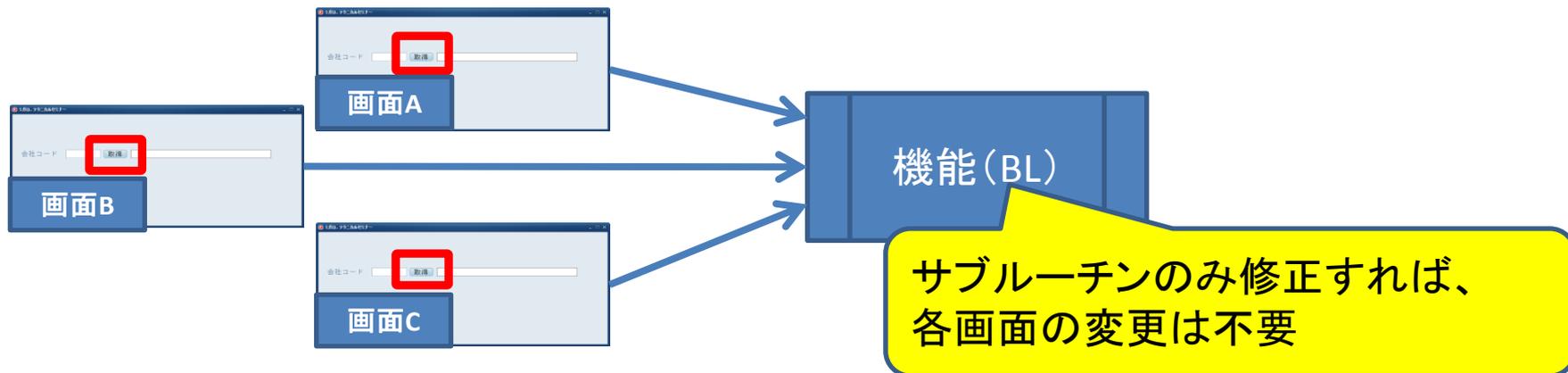
## ■ メンテナンス性を上げる為には…

- 画面（ユーザーインターフェース：UI）と機能（ビジネスロジック：BL）の分離を検討する！



## ■ 機能をサブルーチン化して分離するメリット

- 機能(BL)の仕様変更時、一カ所修正すればよい。



- 異なる画面(UI)要素にも同じ機能を利用可能。

ミガロ、テクニカルセミナー

会社コード 0001 取得 株式会社ミガロ。

ミガロ、テクニカルセミナー

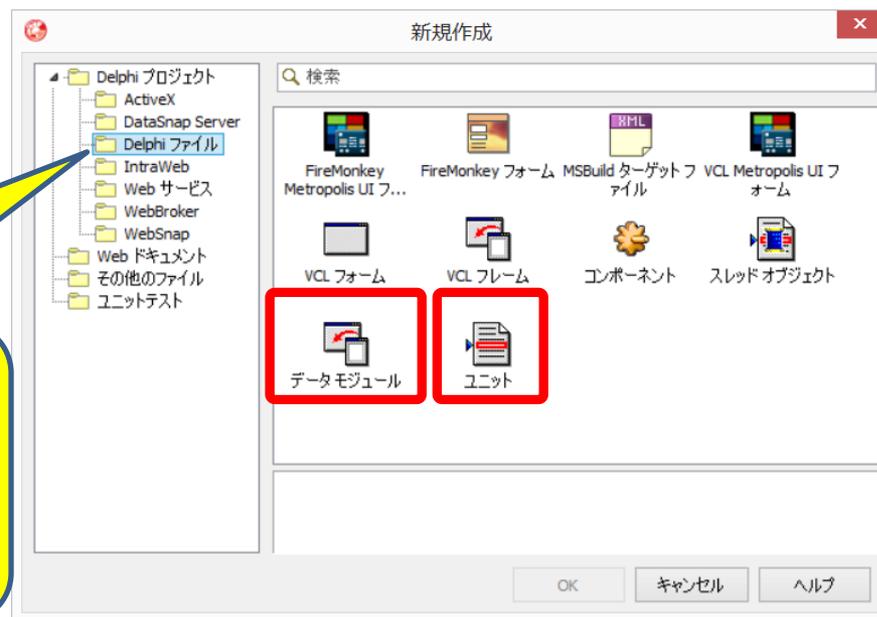
会社コード	会社名
0001	株式会社ミガロ、 日本IBM株式会社
0003	エンバカデロテクノロジーズ合同会社

異なる画面でも同じサブルーチンが使用可能

# ■ Delphi/400でのサブルーチン作成方法

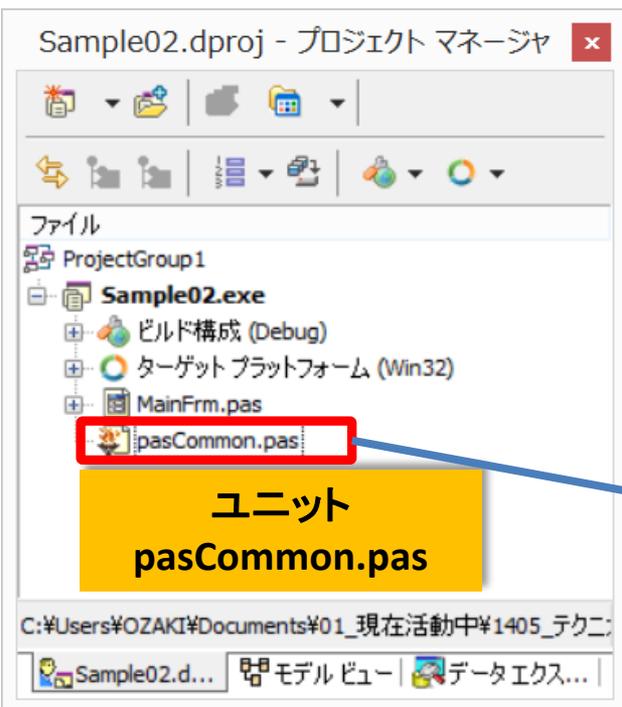
- 2つのサブルーチン作成方法
  - 方法(1) ユニットファイルの作成
    - ソースコードが追加できる
  - 方法(2) データモジュールの作成
    - 非ビジュアルコンポーネントが追加できる

[ファイル]→[新規作成]→[その他...]を選択。  
新規作成ダイアログから  
[Delphiプロジェクト]→[Delphiファイル]を選択。



## ■ サブルーチン化方法 (1)

- 共通サブルーチン用 ユニットファイルを作成



interface

【宣言部】

function GetCompNM (ACompCD: String): String;

implementation

【実装部】

function GetCompNM (ACompCD: String): String;  
begin

if ACompCD = '0001' then  
Result := '株式会社ミガロ.'

else if ACompCD = '0002' then  
Result := '日本IBM株式会社'

else if ACompCD = '0003' then  
Result := 'エンバカデロテクノロジーズ合同会社'

else

raise Exception.Create('会社コードが正しくありません');  
end;

〔会社名取得SUB〕

関数名: GetCompNM

引数: ACompCD(会社CD)

戻り値: 取得した会社名

## ■ サブルーチン化方法 (1)

- ユニットに定義した関数の呼び出し



### サブルーチン化方法(1) ユニット関数呼出

```
uses pasCommon; //----- 使用するユニット定義
```

```
procedure TfrmMain.btnGetCompClick(Sender: TObject);  
begin
```

```
    //会社コードにより名称を取得
```

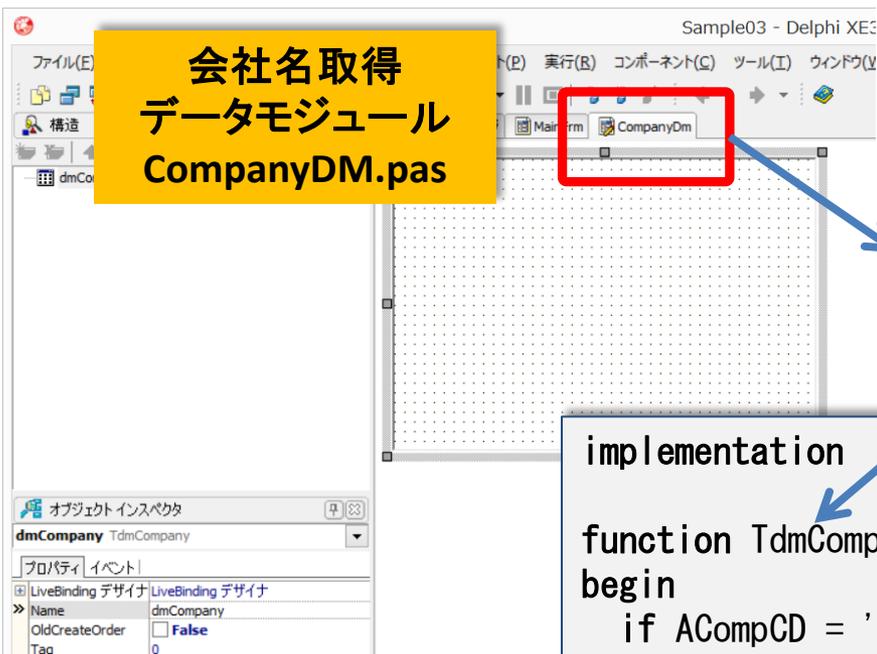
```
    edtCompNM.Text := GetCompNM(edtCompCD.Text);
```

```
end;
```

データ取得処理をサブルーチン化することにより、画面(UI)と機能(BL)の分離が完了。

## ■ サブルーチン化方法 (2)

- 共通サブルーチン用 データモジュールを作成



[Ctrl]+[Shift]+[C]  
で雛形作成

【実装部】

【宣言部】

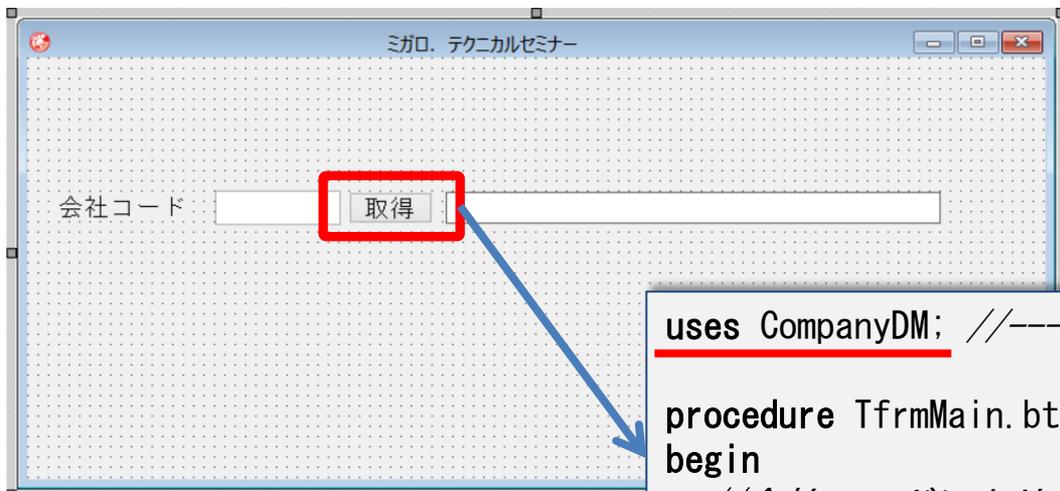
```
type
  TdmCompany = class (TDataModule)
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    function GetCompNM (ACompCD: String): String;
  end;
var
  dmCompany: TdmCompany;
```

implementation

```
function TdmCompany.GetCompNM (ACompCD: String): String;
begin
  if ACompCD = '0001' then
    Result := '株式会社ミガロ.'
  else if ACompCD = '0002' then
    Result := '日本IBM株式会社'
  else if ACompCD = '0003' then
    Result := 'エンバカデロテクノロジーズ合同会社'
  else
    raise Exception.Create('会社コードが正しくありません');
end;
```

## ■ サブルーチン化方法 (2)

- データモジュールに定義した関数の呼び出し



サブルーチン化方法(2)  
データモジュール関数呼出

```
uses CompanyDM; //----- 使用するユニット定義  
  
procedure TfrmMain.btnGetCompClick(Sender: TObject);  
begin  
    //会社コードにより名称を取得  
    edtCompNM.Text := dmCompany.GetCompNM(edtCompCD.Text);  
end;
```

データモジュールの関数として呼び出す。  
(dmCompany.GetCompNM)

サブルーチン化を検討する場合、  
ユニットファイルとデータモジュールのどちらを使用すればよい？

## ■ ユニットファイル と データモジュール

### • 仕様変更が発生！

- 会社コードからIBMi上の顧客マスタ(MCUSTP)を参照し、顧客名を取得して画面にセットするように変更してほしい！

セッション A - [24]

ファイル(E) 編集(E) 表示(V) 通信(C) アクション(A) ウィンドウ(W) ヘルプ(H)

ホスト: MIGARO15 ポート: 23

報告書の表

報告書 183

行の位置指定 . . . . . 1 . . . . . 2 . . . . . 3 . . . . . 4 . . . . . 5 . . . . . 6 . . . . . 7 . . . . .

行 + . . . . . 1 + . . . . . 2 + . . . . . 3 + . . . . . 4 + . . . . . 5 + . . . . . 6 + . . . . . 7 + . . . . .

顧客NO	顧客名	住所1
000001	1,221 ココナッツマリンショップ	大島町 4-976-3
000002	1,513 ダイブハウスタートル	東荻 5-8-7
000003	3,444 ダイビングベース新井	新井 2-14-3
000004	1,231 アクアダイビングセンター	明太区曾根 541
000005	1,351 亀山ダイブセンター	稲毛区亀山町 6
000006	1,380 ダイブショップブルーリーフ	鯖松町 23-738
000007	1,384 MHM ダイバーズクラブ	埴輪町 32

# ■ ユニットファイル と データモジュール

## • ユニットファイル

- ソースコードしか持っていないので、直接コンポーネントが使えない。

ユニットファイルは、ソースコードしかない為、

- ・ ソースから動的にコンポーネントを作成する
- ・ 別のデータモジュールを呼び出す

等の対応が必要になってしまう。

```
//-----会社コード取得関数
function GetCompNM (ACompCD: String): String;
begin
    //会社コードにより名称を取得
    if ACompCD = '0001' then
        Result := '株式会社ミガロ.'
    else if ACompCD = '0002' then
        Result := '日本IBM株式会社'
    else if ACompCD = '0003' then
        Result := 'エンバカデロテクノロジーズ合同会社'
    else
        raise Exception.Create('会社コードが正しくありません');
end;
```

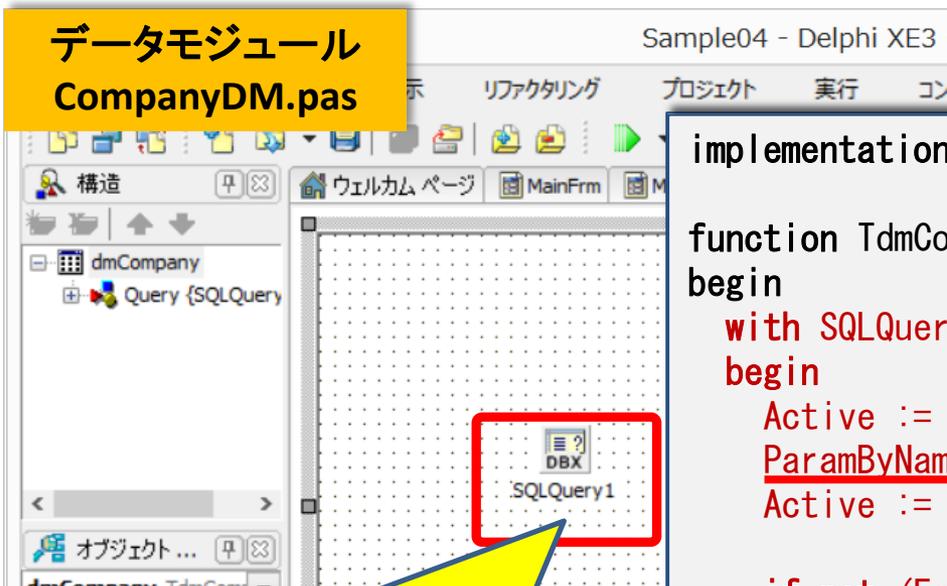
# ■ ユニットファイル と データモジュール

## • データモジュール

- 非ビジュアルコンポーネントが使用できる為、データベース処理を追加可能。

データモジュール  
CompanyDM.pas

【実装部】



### 【TSQLQuery】

SQLプロパティ:

```
SELECT * FROM MCUSTP  
WHERE CUSTNO = :CUSTNO
```

implementation

```
function TdmCompany.GetCompNM (ACompCD: String): String;  
begin  
  with SQLQuery1 do  
  begin  
    Active := False;  
    ParamByName('CUSTNO').AsInteger := StrToIntDef(ACompCD, 0);  
    Active := True;  
  
    if not (Eof and Bof) then  
      Result := FieldByName('CUSTNM').AsString  
    else  
      raise Exception.Create('会社コードが正しくありません');  
  end;  
end;
```

# ■ ユニットファイル と データモジュール

## • 画面(UI)ユニット

- 既に画面(UI)と機能(BL)が分離されているため、仕様変更があっても修正不要！

### 画面(UI)プログラムソース

```
uses CompanyDM; //----- 使用するユニット定義

procedure TfrmMain.btnGetCompClick(Sender: TObject);
begin
    //会社コードにより名称を取得
    edtCompNM.Text := dmCompany.GetCompNM(edtCompCD.Text);
end;
```

## • 実行結果

ミガロ. テクニカルセミナー

会社コード

顧客マスターから取得した顧客名が出力

# ■ ユニットファイル と データモジュール

## • ユニットファイル

- **業務に依存しないような汎用的な共通処理に最適**

(例) ログオンユーザー名取得関数  
ドキュメントフォルダ取得関数  
日付/時刻値⇔数値変換関数

- ソースコードのみ記述可能。
- usesリストに追加するだけで、使用可能。

アプリケーション全体で  
使用するデータモジュールを  
メインフォーム生成前に生成

## • データモジュール

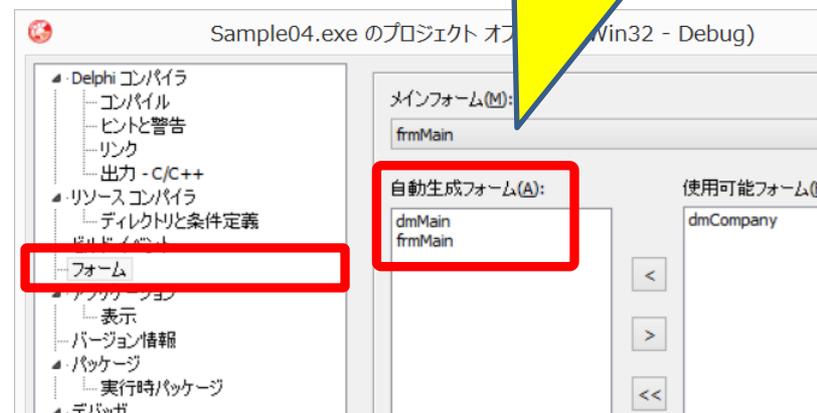
- **業務に関連する共通処理に最適**

(例) データアクセス処理  
帳票処理

- 非ビジュアルコンポーネント使用可。
- オブジェクトの為、事前の生成が必要。

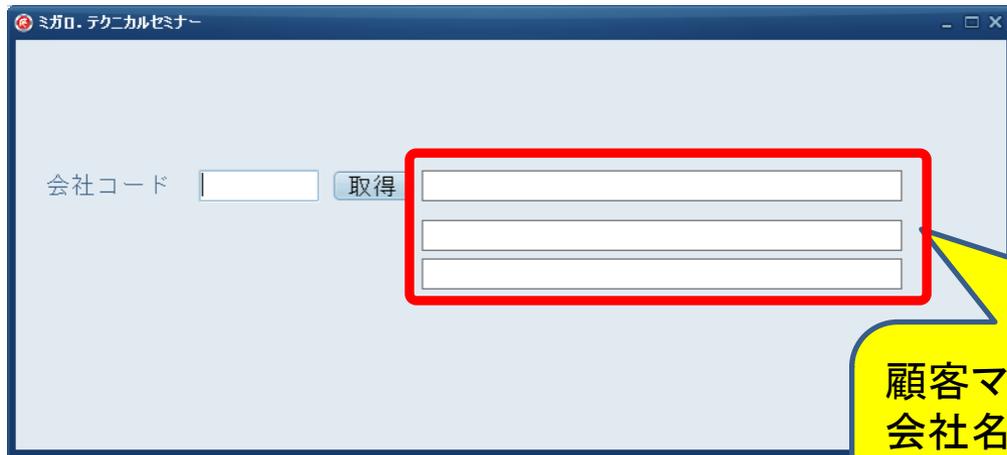
自動生成 (アプリケーション全体で使用するもの)

Createメソッド (特定の画面のみで使用するもの)



## ■ データモジュールのメリット

- さらに仕様変更が発生！
  - 会社名だけでなく、住所と電話番号も一緒に表示してほしい！



顧客マスタ(MCUSTP)より下記項目を取得  
会社名 : 顧客名 (CUSTNM)  
住所 : 住所1 (ADDR1)  
電話番号 : TEL (TEL)

サブルーチンから複数の値を取得するにはどうすればよい？

# ■ データモジュールのメリット

## • オブジェクト変数の活用

- データモジュールはオブジェクトであるため、一つのクラスの中にサブルーチン(手続き/関数)だけでなく、変数も定義可能。

```
type
  TdmCompany = class(TDataModule)
    SQLQuery1: TSQLQuery;
  private
    { Private 宣言 }
  public
    { Public 宣言 }
  function GetCompNM (ACompCD: String): String;
end;
```

改良前の宣言部

関数の為、結果値を1つしかセットできない

結果値を変数で返せるように変更  
(オブジェクト変数)

サブルーチンを関数→手続きに変更

```
type
  TdmCompany = class(TDataModule)
    SQLQuery1: TSQLQuery;
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    CompNM: String; //取得会社名
    CompADDR: String; //取得住所
    CompTEL: String; //取得TEL
    procedure LoadData (ACompCD: String);
end;
```

改良後の宣言部

# ■ データモジュールのメリット

## ● 機能(BL)ロジックの変更

【実装部】

```
procedure TdmCompany.LoadData (ACompCD: String);
begin
  //初期化
  CompNM := '';
  CompADDR := '';
  CompTEL := '';

  with SQLQuery1 do
  begin
    Active := False;
    ParamByName('CUSTNO').AsInteger := StrToIntDef(ACompCD, 0);
    Active := True;

    if not (Eof and Bof) then
    begin
      CompNM := FieldByName('CUSTNM').AsString;
      CompADDR := FieldByName('ADDR1').AsString;
      CompTEL := FieldByName('TEL').AsString;
    end
  else
    raise Exception.Create('会社コードが正しくありません');
  end;
end;
```

オブジェクト変数を初期化

オブジェクト変数に取得値をセット

# ■ データモジュールのメリット

## • 画面(UI)ユニット

- オブジェクト変数の値を画面にセットするよう修正

画面(UI)プログラムソース

```
uses CompanyDM; //----- 使用するユニット定義

procedure TfrmMain.btnGetCompClick(Sender: TObject);
begin
  //会社情報を取得
  dmCompany.LoadData(edtCompCD.Text);
  //会社コードにより名称を取得
  edtCompNM.Text := dmCompany.CompNM;
  edtCompAddr.Text := dmCompany.CompADDR;
  edtCompTel.Text := dmCompany.CompTEL;
end;
```

## • 実行結果

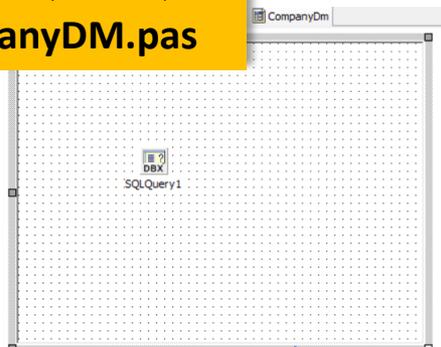
顧客名に加え、住所, TEL が出力

# ■ データモジュールのメリット

## ● 機能拡張性が向上

- 出力項目を追加する場合、オブジェクト変数を追加するだけで拡張可能。
- 追加項目を利用する画面 (UI) のみ修正すれば良い。

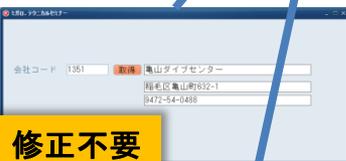
データモジュール  
CompanyDM.pas



```
type
  TdmCompany = class (TDataModule)
    SQLQuery1: TSQLQuery;
  private
    { Private 宣言 }
  public
    { Public 宣言 }
    CompNM: String; //取得会社名
    CompADDR: String; //取得住所
    CompTEL: String; //取得TEL
    CompFAX: String; //取得FAX
    CompCNTACT: String; //取得顧客担当者
  procedure LoadData (ACompCD: String);
end;
```

【宣言部】

オブジェクト変数の追加  
LoadData手続きに処理追加



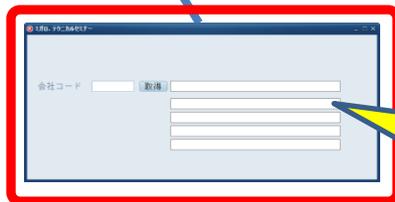
修正不要



修正不要



修正不要



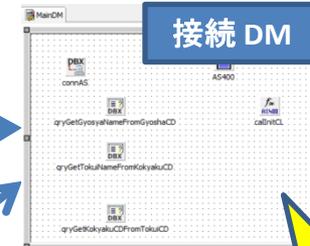
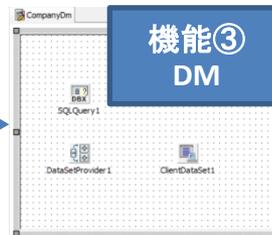
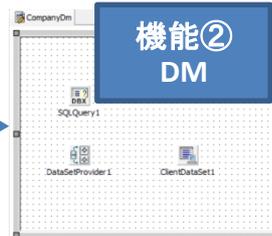
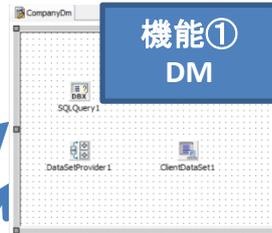
この画面にのみ、  
FAX番号と顧客担当者を  
追加したい

# ■ データモジュール使用例

画面(UI) Form

機能(BL) DM

接続管理用 DM



IBM iへの接続管理を実施  
TAS400 (ネイティブ接続)  
TSQLConnection (dbExpress)

IBMiへの接続は全て機能(BL)DMが担当

機能(BL)DMのサブルーチン呼出

SQLやCALLを使用してIBMiに問合せ

画面の仕様に関わらず同じ処理が利用可能

# 3. BDEからdbExpressへの移行手順

# ■ BDEについて

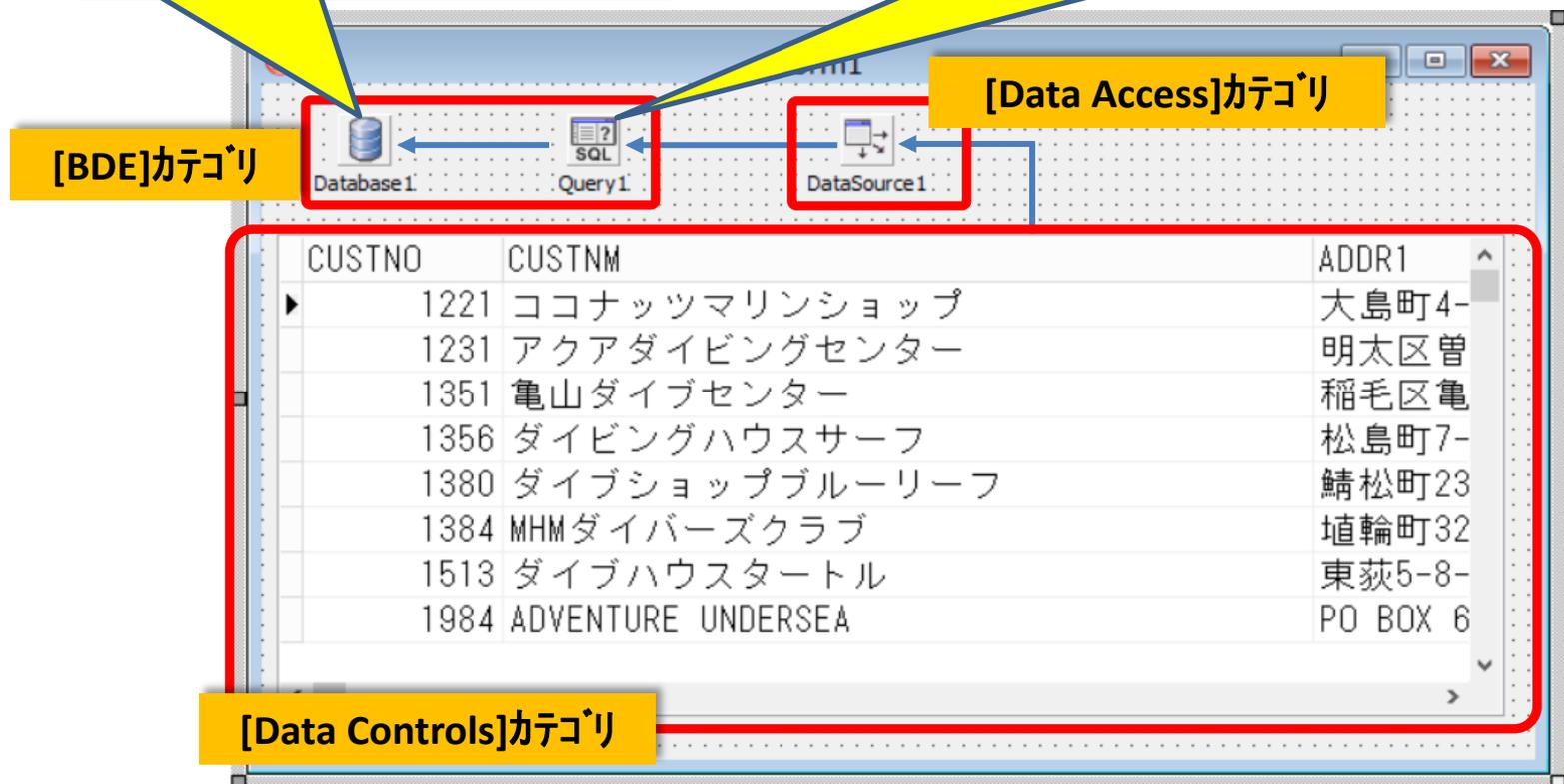
## • IDAPIを利用したデータベースエンジン

### 【TDataBase】

データベース接続を管理  
IBMi の場合 IDCO400ドライバ

### 【TQuery】

SQLの実行  
SELECTの場合、結果がDataSetとなる。



# ■ dbExpressについて

## ● 単方向データセット

- データベースエンジン軽量化の為、レコードバッファリング機能がなく、DataSetは、先頭からの順次読み込みのみが可能。
- TDBGridに必要なDataSetの双方向移動や、DataSetの更新機能を実現する場合には、TClientDataSetを併用する。

【TSQLConnection】

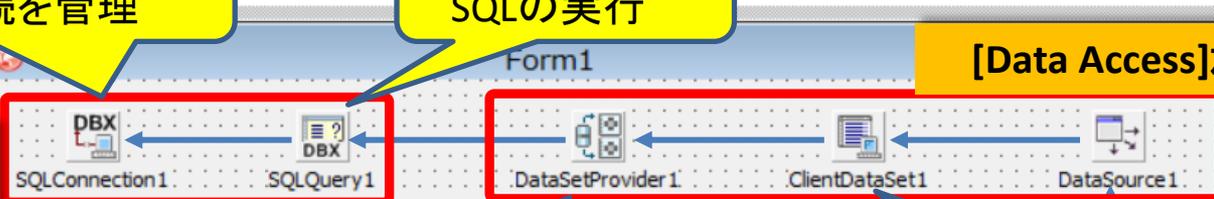
データベース接続を管理

【TSQLQuery】

SQLの実行

[Data Access]カテゴリ

[dbExpress]  
カテゴリ



【TDataSetProvider】

TClientDataSetへ  
DataSetを提供

【TClientDataSet】

メモリ上のDataSet

[Data Controls]  
カテゴリ

CUSTNO	CUSTNM	ADDR1
1380	ダイブショップブルーリーフ	相模区電 松島町7-
	MHMダイバーズクラブ	鯖松町23
	ダイブハウスタートル	埴輪町32
	ADVENTURE UNDERSEA	東荻5-8-
		PO BOX 6

## ■ Windows7以降のOSへの移行を検討

### • BDEの継続利用の場合

- IBM i 用 IDCO400ドライバは最新のV7R1対応版有。
- 正式対応OSは、WindowsXP迄。
  - 64bit ネイティブ環境での動作不可の為、32bit互換モードとなる。
  - Windows7以降で運用する場合は、BDEアプリケーションの設定変更やUAC等OS側の調整が必要。
    - 第11回テクニカルセミナー『Windows7に最適化したアプリ開発・運用テクニック』参照

### • dbExpressへ移行する場合

- TClientDataSetの追加検討が必要。
  - TDBGridやTDBNavigator等を使用する場合、必須。

BDEアプリケーションをdbExpressへ移行するにはどうすればよいか？

## ■ サンプルアプリケーション

### • プログラム仕様

- 入力会社コードに合致する受注データを取得して、一覧表示してほしい！

受注NO	顧客NO	受注日	出荷日	担当者NO	担当者名	金
1003	1351	20120412	20120503	114	田川 雄志	
1052	1351	20120106	20120107	144	有澤 裕之	
1055	1351	20120204	20120205	29	市原 秀之	
1067	1351	20120401	20120402			
1075	1351	20120421	20120422			
1087	1351	20120520	20120521			
1152	1351	20120407	20120407			
1155	1351	20120505	20120505			
1163	1351	20120614	20120614			
1255	1351	20121209	20121209			
1942	1351	20120821	20121230			
1971	1351	20120725	20121230			
1983	1351	20120725				
1985	1351	20120725				

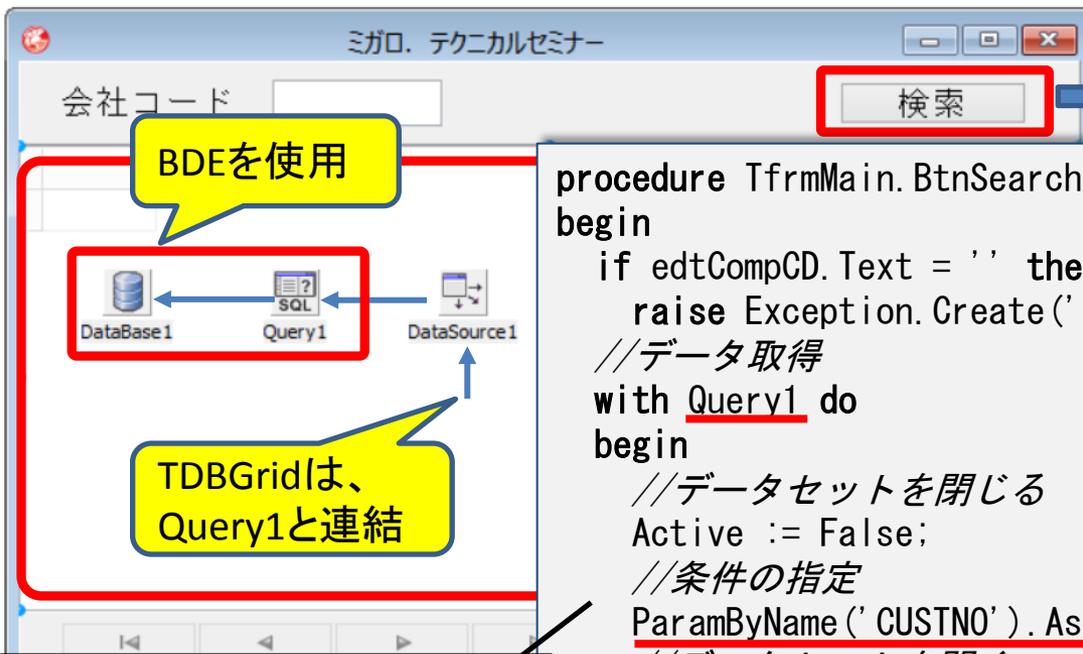
受注ファイル(FORDRL01)から顧客NO(CUSTNO)が、画面入力値と一致するデータを抽出する。その際に、担当者NO(EMPNO)をキーに、担当者マスタ(MEMPLP)より担当者名(EPNAME)を取得して出力する。

#### 【SQL】

```
SELECT OD.*, EM.EPNAME FROM FORDRL1 OD  
LEFT JOIN MEMPLP EM ON OD.EMPNO = EM.EMPNO  
WHERE OD.CUSTNO = :CUSTNO  
ORDER BY OD.CUSTNO, OD.ORDRNO
```

# ■ サンプルアプリケーション

## ● 検索ボタン[btnSearch]のクリック処理

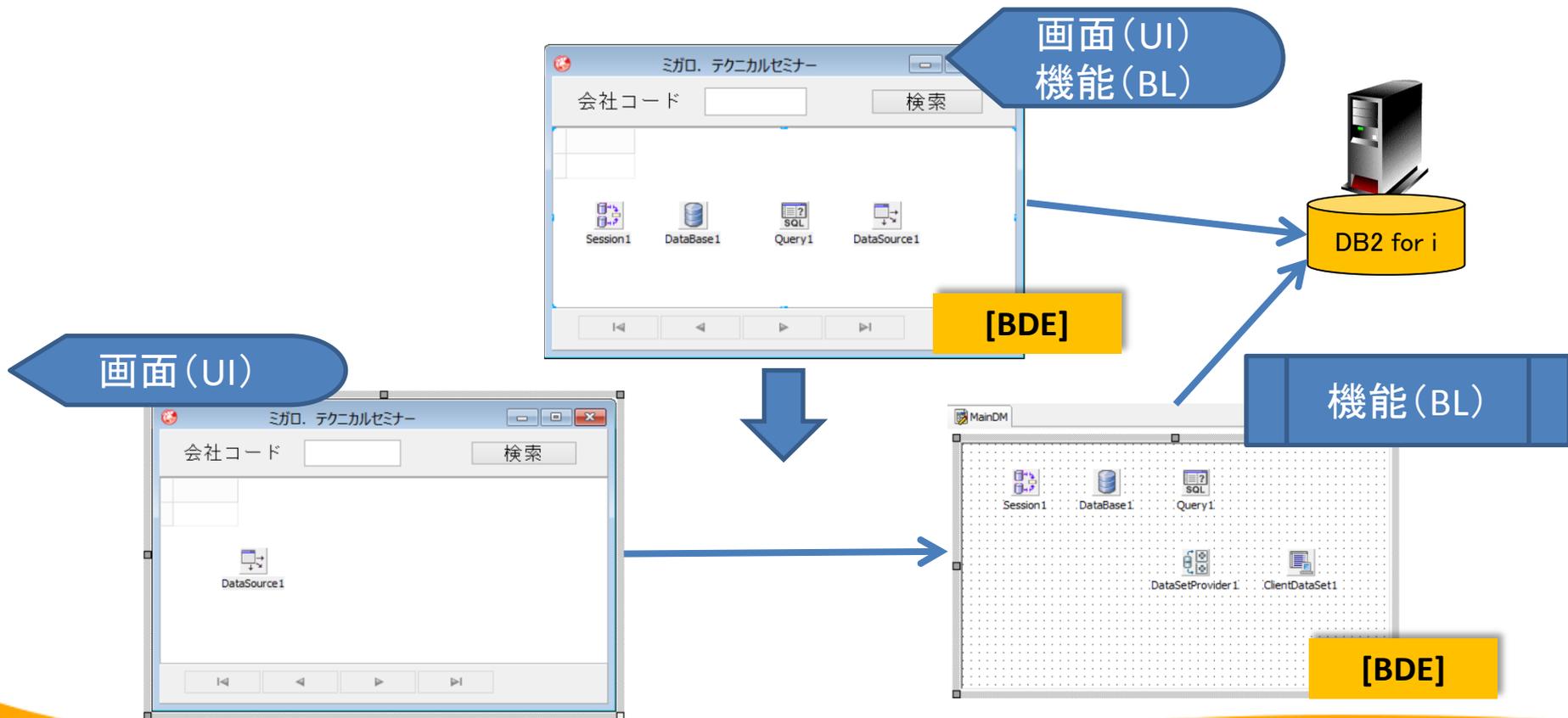


```
procedure TfrmMain.BtnSearchClick(Sender: TObject);
begin
  if edtCompCD.Text = '' then
    raise Exception.Create('会社コードが入力されていません');
  //データ取得
  with Query1 do
  begin
    //データセットを閉じる
    Active := False;
    //条件の指定
    ParamByName('CUSTNO').AsInteger := StrToIntDef(edtCompCD.Text, 0);
    //データセットを開く
    Active := True;
    //対象データが存在しない場合、データセットを閉じて終了
    if Eof and Bof then
    begin
      Active := False;
      raise Exception.Create('対象データが存在しません');
    end;
  end;
end;
```

会社コード(edtCompCD)を使用して、直接BDEのQuery1を実行。  
→ GUI部品(画面)をロジック(機能)が混在したプログラムとなっている。

## ■ BDEからdbExpressへの移行手順

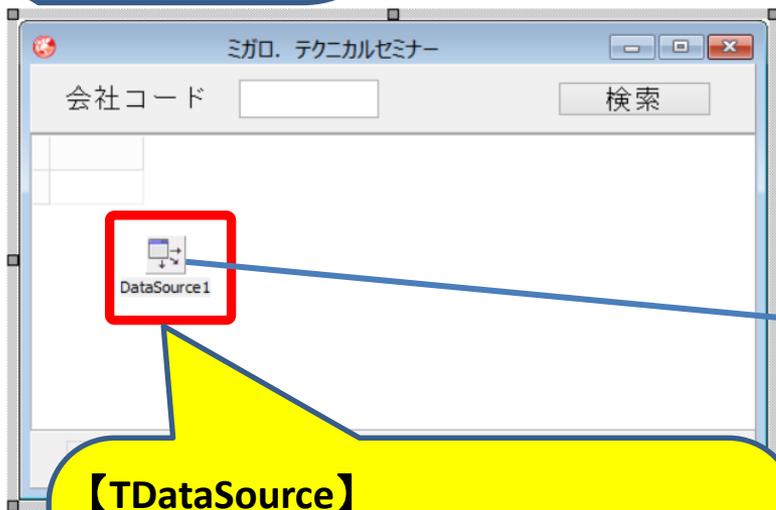
- (ステップ1) 画面(UI)と機能(BL)に分離
  - 画面(UI)の中からは、一切 IBMiへ直接アクセスしないように変更。
  - データモジュールの中からのみ、IBMiへBDEアクセスを行う。



## ■ (ステップ1) 画面(UI) と機能(BL) に分離

- dbExpressで必須のTClientDataSetは、BDEでも使用可能
  - 一度に移行せず、まずはBDEをTClientDataSet経由で利用するように変更

画面(UI)

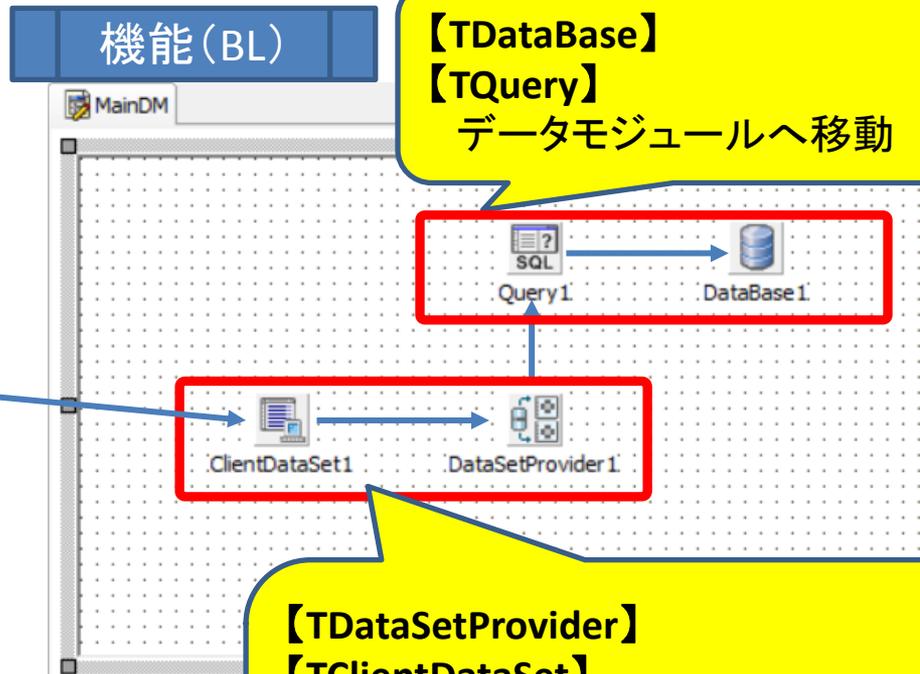


【TDataSource】

DataSet : データモジュール上の  
ClientDataSetを指定

画面は、BDEに依存しない  
TClientDataSetのみ参照

機能(BL)



【TDataBase】

【TQuery】

データモジュールへ移動

【TDataSetProvider】

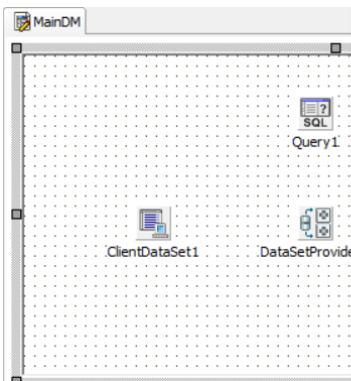
【TClientDataSet】

データモジュールに追加

※TQueryにDataSetイベントを定義している  
場合、TClientDataSetへイベントを移行

# ■ (ステップ1) 画面(UI) と機能(BL) に分離

- 機能(BL) データモジュール側プログラム
  - サブルーチンを追加し、P.36のロジックを移行



```
procedure TdmMain.OpenOrderData (ACMCD: String);  
begin  
  if ACMCD = '' then  
    raise Exception.Create('会社コードが入力されていません');  
  //データ取得  
  with ClientDataSet1 do // <---- Query1 → ClientDataSet1  
  begin  
    //データセットを閉じる  
    Active := False;  
    //条件の指定  
    Query1.ParamByName('CUSTNO').AsInteger := StrToIntDef(ACMCD, 0);  
    //データセットを開く  
    Active := True;  
    //対象データが存在しない場合、データセットを閉じて終了  
    if Eof and Bof then  
    begin  
      Active := False;  
      raise Exception.Create('対象データが存在しません');  
    end;  
  end;  
end;  
end;
```

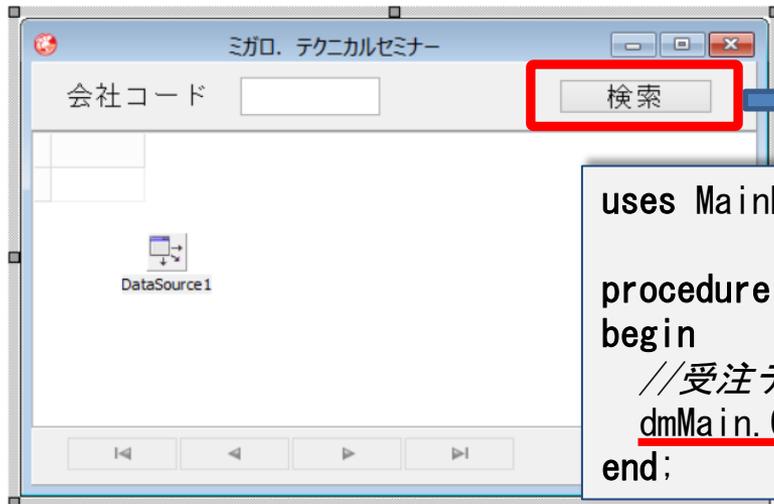
会社コードをパラメータとした  
OpenOrderData手続きを作成

Query1の操作を  
ClientDataSet1 に変更

SQLのパラメータは、  
Query1 を使用

# ■ (ステップ1) 画面 (UI) と機能 (BL) に分離

## ● 画面 (UI) フォーム側プログラム



```
uses MainDM; //----- 使用するユニット定義

procedure TfrmMain.btnSearchClick(Sender: TObject);
begin
  //受注データを開く
  dmMain.OpenOrderData(edtCompCD.Text);
end;
```

## ● 実行結果

受注NO	顧客NO	受注日	出荷日	担当者NO	担当者名	金
1003	1351	20120412	20120503	114	田川 雄志	
1052	1351	20120106	20120107	144	有澤 裕之	
1055	1351	20120204	20120205	29	市原 秀之	
1067	1351	20120401	20120402	34	出井 昌一	
1075	1351	20120421	20120422	11	権 真由美	
1087	1351	20120520	20120521	127	島田 和之	
1152	1351	20120407	20120407	24	島田 和之	

TDBGridは、TClientDataSetと連結。

※ TClientDataSetは、メモリ上にキャッシュされてから表示される。

## ■ (ステップ1) 画面(UI) と機能(BL) に分離

- TClientDataSetを利用する場合の留意点
  - オープン時、取得データ全件がメモリにキャッシュされる。
    - 大量のデータ取得時、レスポンスが悪化

一度にキャッシュするレコード数の調整が可能

TClientDataSet : PacketRecordsプロパティ  
件数を指定、初期値:-1の場合全件

- TClientDataSet上のデータセットはメモリ上しか更新されない。
  - 変更点をIBM iに適用する必要がある

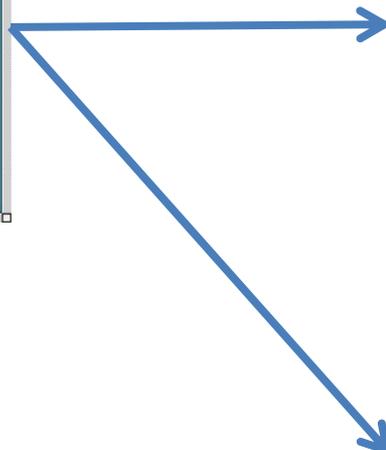
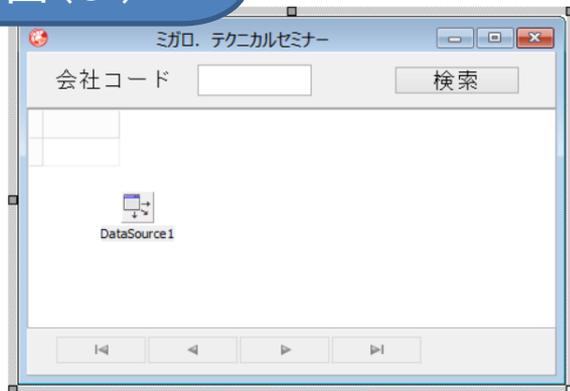
メモリ上の変更(Append/Edit/Delete)データの適用が可能

TClientDataSet : ApplyUpdatesメソッド

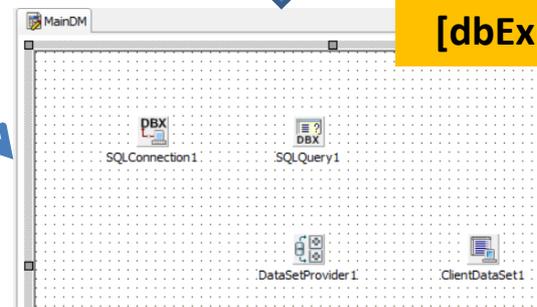
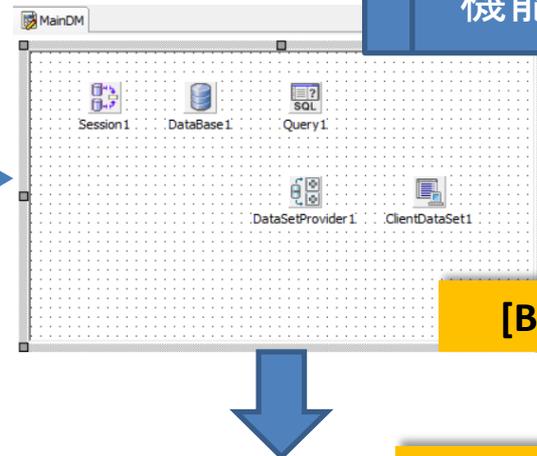
## ■ BDEからdbExpressへの移行手順

- (ステップ2) 機能(BL)のBDEをdbExpressに変更
  - データモジュール中のBDEコンポーネントをdbExpressコンポーネントに置換。
  - 機能(BL)の変更のみの為、画面(UI)は修正不要。

画面(UI)



機能(BL)



## ■ (ステップ2) 機能 (BL) のBDEをdbExpressに変更

- データモジュール内のみ変更
  - BDEからdbExpressへの変更
    - BDEコンポーネントからdbExpressコンポーネントへの貼り換え
    - TDataSetProviderの接続先DataSetを TSQLTable or TSQLQueryに変更

BDE	dbExpress
TDataBase	TSQLConnection
TTable	TSQLTable
TQuery	TSQLQuery
TSession	(不要)

- BDEロジックからdbExpressロジックへの変更
  - TDataBase から TSQLConnection に変更によるロジック変更
    - データベース接続処理
    - トランザクション処理 等

## ■ (ステップ2) 機能 (BL) のBDEをdbExpressに変更

### • dbExpressを利用する場合の留意点

- TTableで、ファイルメンバーを指定していた場合修正が必要。
  - TSQLTableは、内部がSQLベースの為、メンバー指定不可

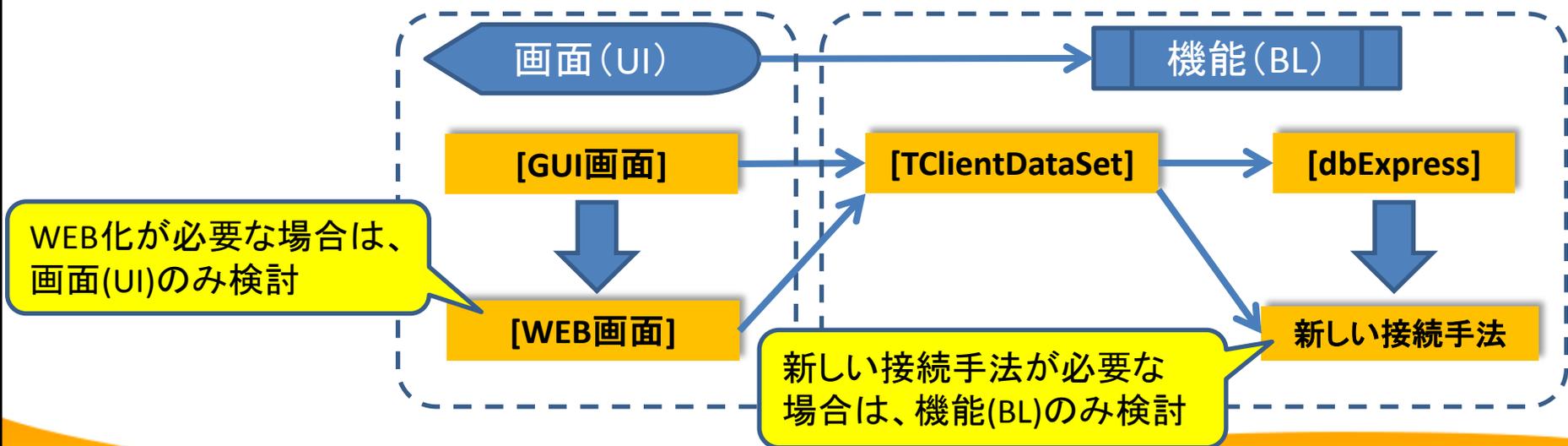
TSQLTableオープン前に OVRDBFを行う。  
TAS400 : RemoteCmdメソッド

- TClientDataSetの更新適用(ApplyUpdates)時エラーになる。
  - ブランク項目があると、[項目の値が必要です]エラーが発生

項目コンポーネントのRequiredプロパティをFalseに変更する。

## ■ dbExpressに移行するメリット

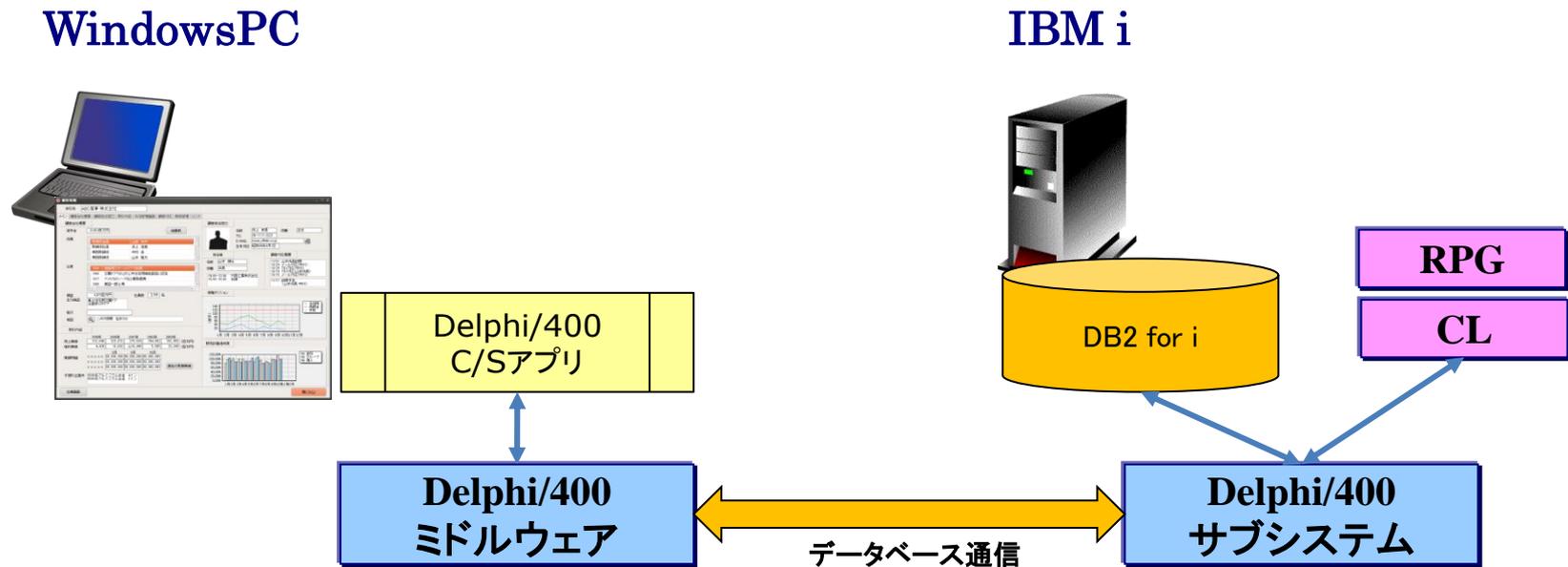
- 最新Windows環境への正式対応が可能
  - UAC環境や、64bit環境上で動作するアプリケーションが作成可能。
    - Windows7 → Delphi/400 ver.2010以降
    - Windows8、64bit版アプリ → Delphi/400 ver.XE3以降
- アプリケーション拡張性向上
  - TClientDataSetにより、機能(BL)が画面(UI)と分離。
  - 将来の環境変化に柔軟に対応可能。



# 4. 将来のマルチデバイス対応を 考慮した開発手法

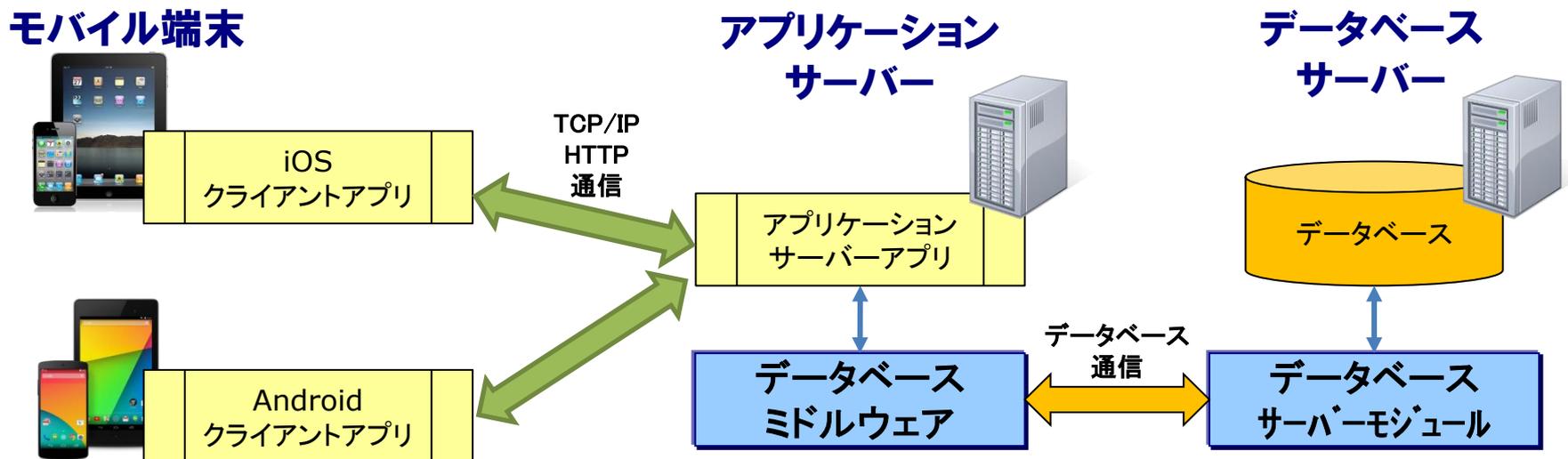
# ■ Windows端末からの一般的なデータベースアクセス

- クライアントPC上に専用のミドルウェアを導入
  - クライアントPC上に、データベースアクセス用のクライアントライブラリ (ミドルウェア)を導入し、直接データベースサーバーとやり取りを行う。



# ■ モバイル端末からのデータベースアクセス

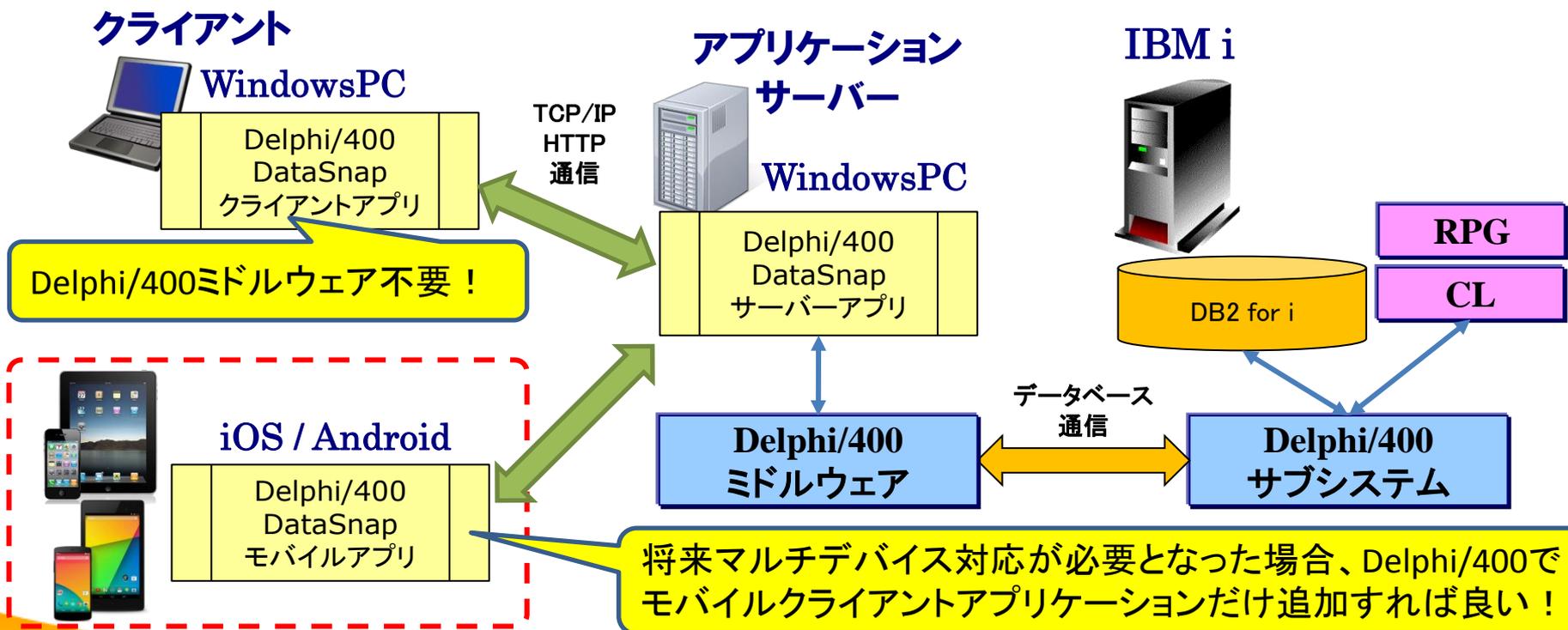
- iOS, Androidから直接のデータベースアクセスは不可
  - モバイル端末の性能は向上しているが、Windowsと比べるとOSレベルの制約が多く、データベースミドルウェアのようなものを実装することは困難。
  - モバイル端末からサーバーへの通信は、HTTPやTCP/IP通信が一般的。



モバイル端末を考慮したデータベースアプリケーションは、  
多層構成で構築する！

# ■ 将来のマルチデバイス対応を考慮した開発手法

- DataSnapによるアプリケーションの多層化を検討
  - 多層化アプリケーション用フレームワークとして、Delphi/400には、DataSnapが搭載！（Delphi/400 Version2009～）
  - DataSnapを使用することにより、簡単にクライアントとアプリケーションサーバー間のTCP/IPあるいはHTTPリモート通信を実現可能！

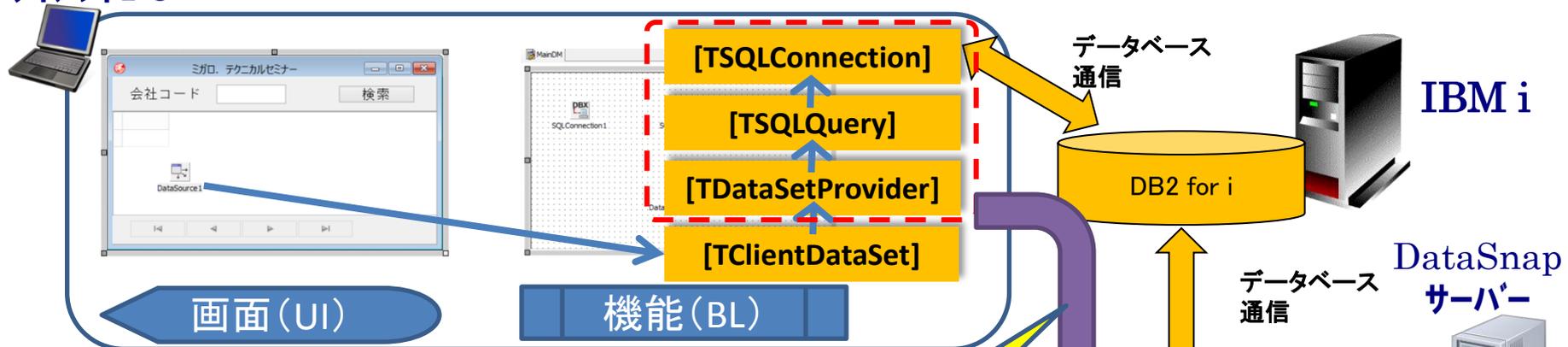


# ■ C/Sアプリケーションの多層化

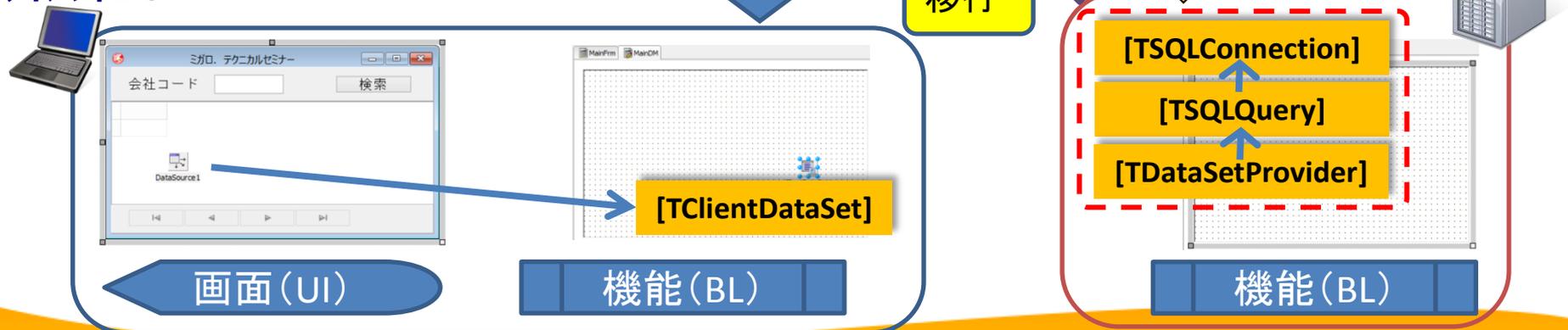
## ● サンプルアプリケーションの多層化を検討

- C/Sアプリケーションで使用していたdbExpress関連の処理をDataSnapサーバーに移行し、クライアントからIBM iの接続関連の処理を排除。

クライアントPC



クライアントPC



## ■ DataSnapサーバー作成方法

- [DataSnapサーバー]ウィザードを使用



- [プロジェクトの種類]
  - サーバーで常時実行する場合、「サービスアプリケーション」を選択。
- [サーバーの機能]
  - 「プロトコル」(TCP/IP or HTTP)、「サーバーメソッドクラス」 を選択。
- [ポート番号]
  - 複数のDataSnapサーバーアプリを稼働する場合、一意なポート番号を付与。
- [サーバーメソッドクラスの上位クラス]
  - 「TDSServerModule」を選択。

## ■ DataSnapサーバー作成方法

- TServerMethods1 (ServerMethodsUnit1.pas)
  - C/Sアプリケーションで定義していたデータベース処理を移行。

この中に、DataSnap  
サーバープログラムを作成

SQLプロパティ:  
SELECT OD.\*, EM.EPNAME FROM FORDRL1 OD  
LEFT JOIN MEMPLP EM ON OD.EMPNO = EM.EMPNO  
WHERE CUSTNO = **:CUSTNO** ORDER BY CUSTNO, ORDRNO  
→ SQL実行時、パラメータ(CUSTNO)に値が必要

クライアントPCからDataSetサーバーへパラメータはどうやって渡す？

# ■ DataSnapサーバー作成方法

## • 公開用 手続き/関数の作成

- Publicに宣言したサブルーチンは、DataSnapクライアントから呼出可能。

### 【宣言部】

```
type
  TServerMethods1 = class(TDSServerModule)
    ...
  private
    { private 宣言 }
  public
    { public 宣言 }
    procedure SetCUSTNO (CUSTNO: Integer);
end;
```

DataSnapクライアントから実行される  
手続きを宣言

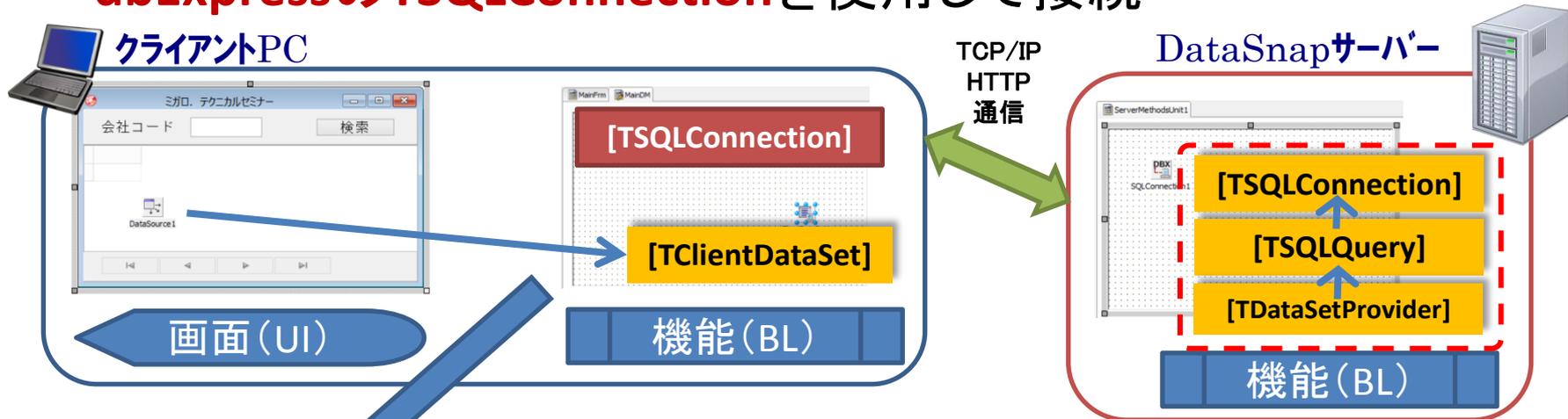
### 【実装部】

```
procedure TServerMethods1.SetCUSTNO (CUSTNO: Integer);
begin
  //SQLにパラメータをセット
  SQLQuery1.ParamByName('CUSTNO').AsInteger := CUSTNO;
end;
```

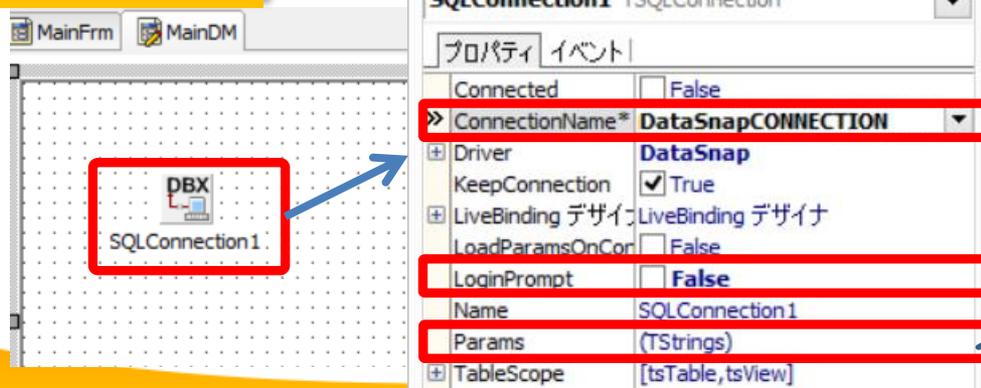
SetCUSTNO手続きが実行されると、  
SQLのパラメータがセットされる

# ■ DataSnapクライアント作成方法

- DataSnapサーバーへの接続方法
  - **dbExpressのTSQLConnection**を使用して接続



【クライアントPC】  
データモジュール



DataSnapCONNECTION を選択

下記パラメータを指定  
Host : サーバーIPアドレス  
Port : ポート番号

## ■ DataSnapクライアント作成方法

- TClientDataSetをDataSnapサーバーへ接続
  - TDSProviderConnection を使用して接続する。

【クライアントPC】  
データモジュール

DataSnapサーバー  
プログラムを定義した  
クラス名を入力。  
(P.52参照)

オブジェクト インспекタ  
DSPProviderConnection1 TDSProviderConnection1

プロパティ	イベント
Connected	<input type="checkbox"/> False
LiveBinding デザイン	LiveBinding デザイン
Name	DSPProviderConnection1
ServerClassName	TServerMethods1
SQLConnection	SQLConnection1
Tag	0

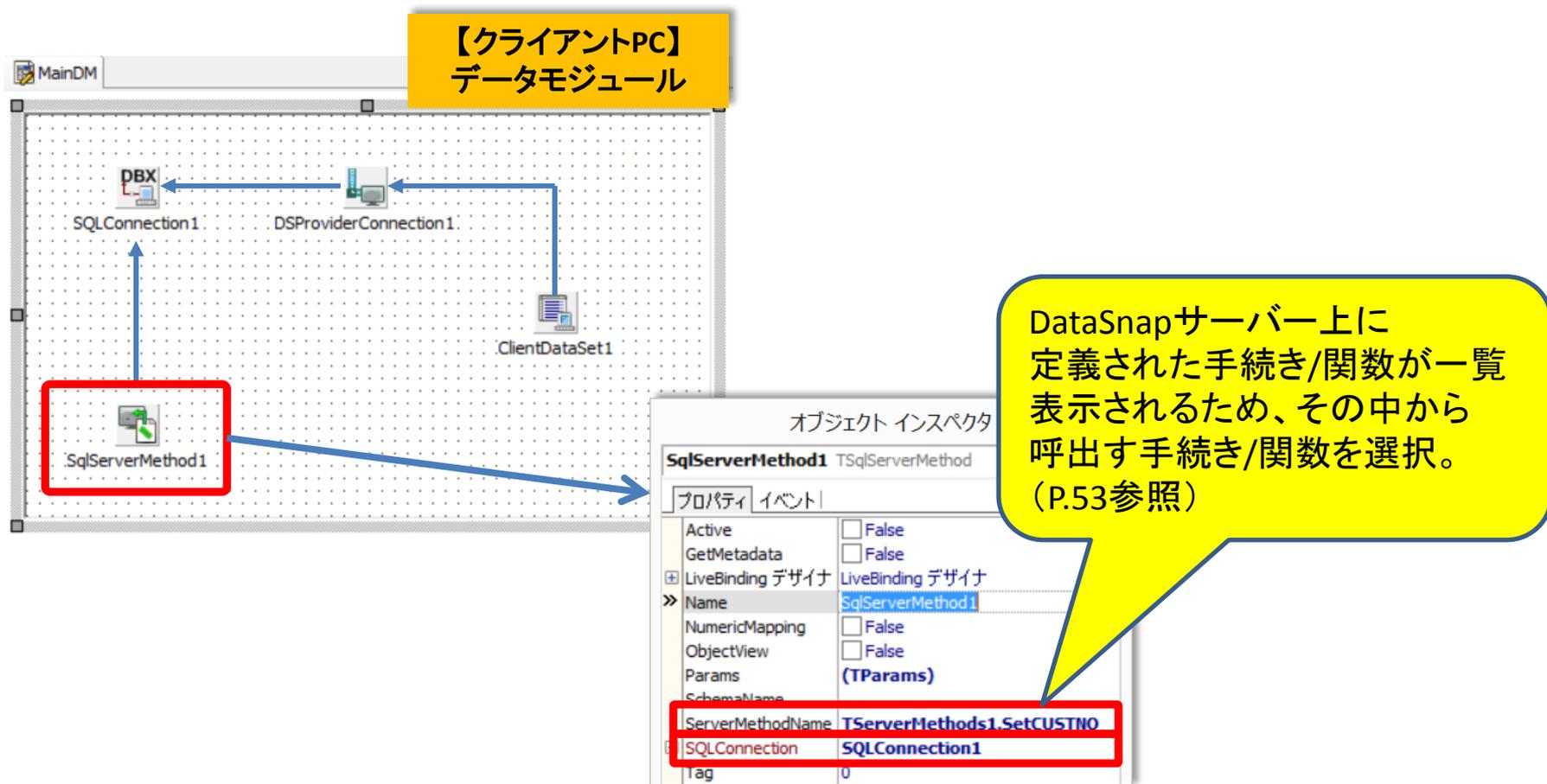
オブジェクト インспекタ  
ClientDataSet1 TClientDataSet

プロパティ	イベント
Name	ClientDataSet1
ObjectView	<input checked="" type="checkbox"/> True
PacketRecords	-1
Params	(TParams)
ProviderName	DataSetProvider1
ReadOnly	<input type="checkbox"/> False
RemoteServer	DSPProviderConnection1
StoreDets	<input type="checkbox"/> False
Tag	0

DataSnapサーバー上に定義された  
DataSetProviderが一覧表示されるため、  
その中から接続するDataSetProviderを  
選択。

# ■ DataSnapクライアント作成方法

- DataSnapサーバー上の手続き/関数 呼出方法
  - **TSqlServerMethod** を使用して、公開された手続き/関数を指定する。



## ■ DataSnapクライアント作成方法

- クライアントのデータモジュールプログラム変更
  - Query1 を参照していた部分を SqlServerMethod1 に変更

```
procedure TdmMain.OpenOrderData (ACMCD: String);
begin
  if ACMCD = '' then
    raise Exception.Create('会社コードが入力されていません');
  //データ取得
  with ClientDataSet1 do
    begin
      //データセットを閉じる
      Active := False;
      //条件の指定
      SqlServerMethod1.ParamByName('CUSTNO').AsInteger := StrToIntDef(ACMCD, 0);
      SqlServerMethod1.ExecuteMethod;
      //データセットを開く
      Active := True;
      //対象データが存在しない場合、データセットを閉じてお
      if Eof and Bof then
        begin
          Active := False;
          raise Exception.Create('対象データが存在しません');
        end;
      end;
    end;
end;
```

変更後のソース

引数を持つ手続き/関数は、ParamByNameメソッドで、引数を指定。

ExecuteMethodメソッドでDataSnap上の手続き/関数を呼び出す。

# ■ DataSnapアプリケーションの実行

## ● 実行例

会社コードをDataSnapサーバーに指定したうえでClientDataSetを開く



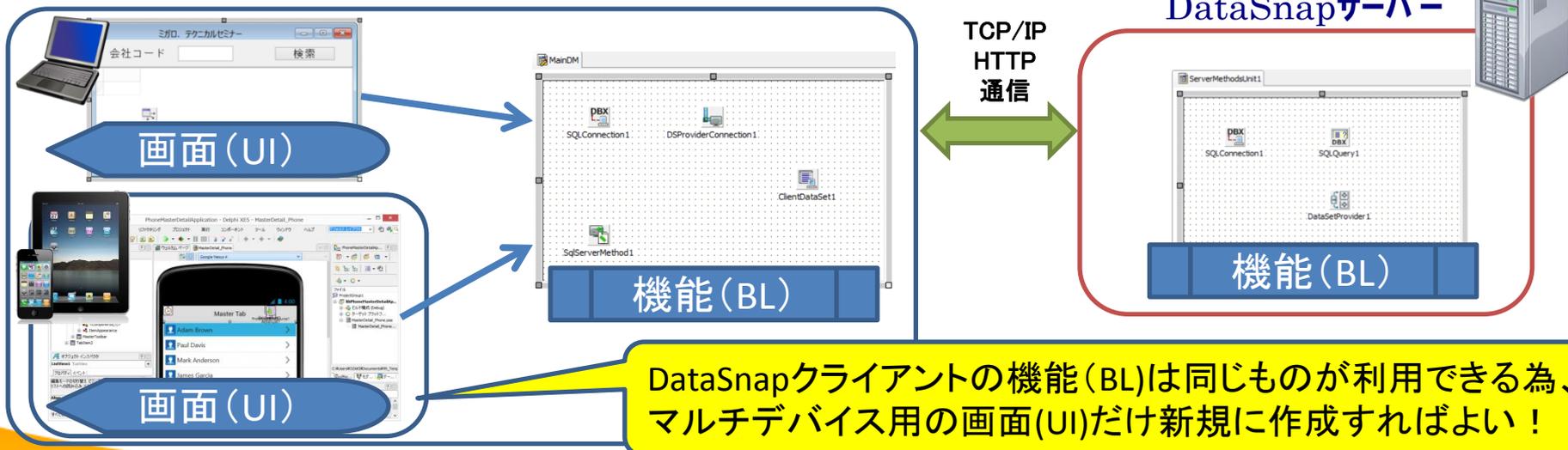
受注NO	顧客NO	受注日	出荷日	担当者NO	担当者名	金
1003	1351	20120412	20120503	114	田川 雄志	
1052	1351	20120106	20120107	144	有澤 裕之	
1055	1351	20120204	20120205	29	市原 秀之	
1067	1351	20120401	20120402	34	出井 昌一	
1075	1351	20120421	20120422	11	権 真由美	
1087	1351	20120520	20120521	127	島田 和之	
1152	1351	20120407	20120407	24	岸ト サロ	



DataSnapServer IPAddress  
192.168.0.161

## ● 将来のマルチデバイス対応時の拡張方法

- 多層化により、マルチデバイス用画面(UI)の作成に注力できる！



# 5. まとめ

## ■ まとめ

- 画面と機能を分離する開発手法
  - 画面(UI)と機能(BL)を分離したサブルーチン化のメリット。
    - 機能(BL)に関する仕様変更時、画面(UI)の修正は不要。
    - 同じ機能(BL)を異なる画面(UI)要素にも適用可能。
  - 機能(BL)をデータモジュール化すれば、機能拡張も容易。
- BDEからdbExpressへの移行手順
  - (ステップ1) TClientDataSetを使用した 画面(UI)と機能(BL)の分離。
  - (ステップ2) データモジュール内のBDEをdbExpressに変更。
- 将来のマルチデバイス対応を考慮した開発手法
  - DataSnapを使用したC/Sアプリケーションの多層化を実施。
  - WindowsとAndroid/iOSで、異なる画面(UI)でも同じ機能(BL)が使用可能。

**ご清聴ありがとうございました**