【セッションNo. 2】

Delphi/400技術セッション

マルチデバイスに対応した IBMi業務システム開発のポイント

株式会社ミガロ.
RAD事業部　営業・営業推進課
尾崎　　浩司

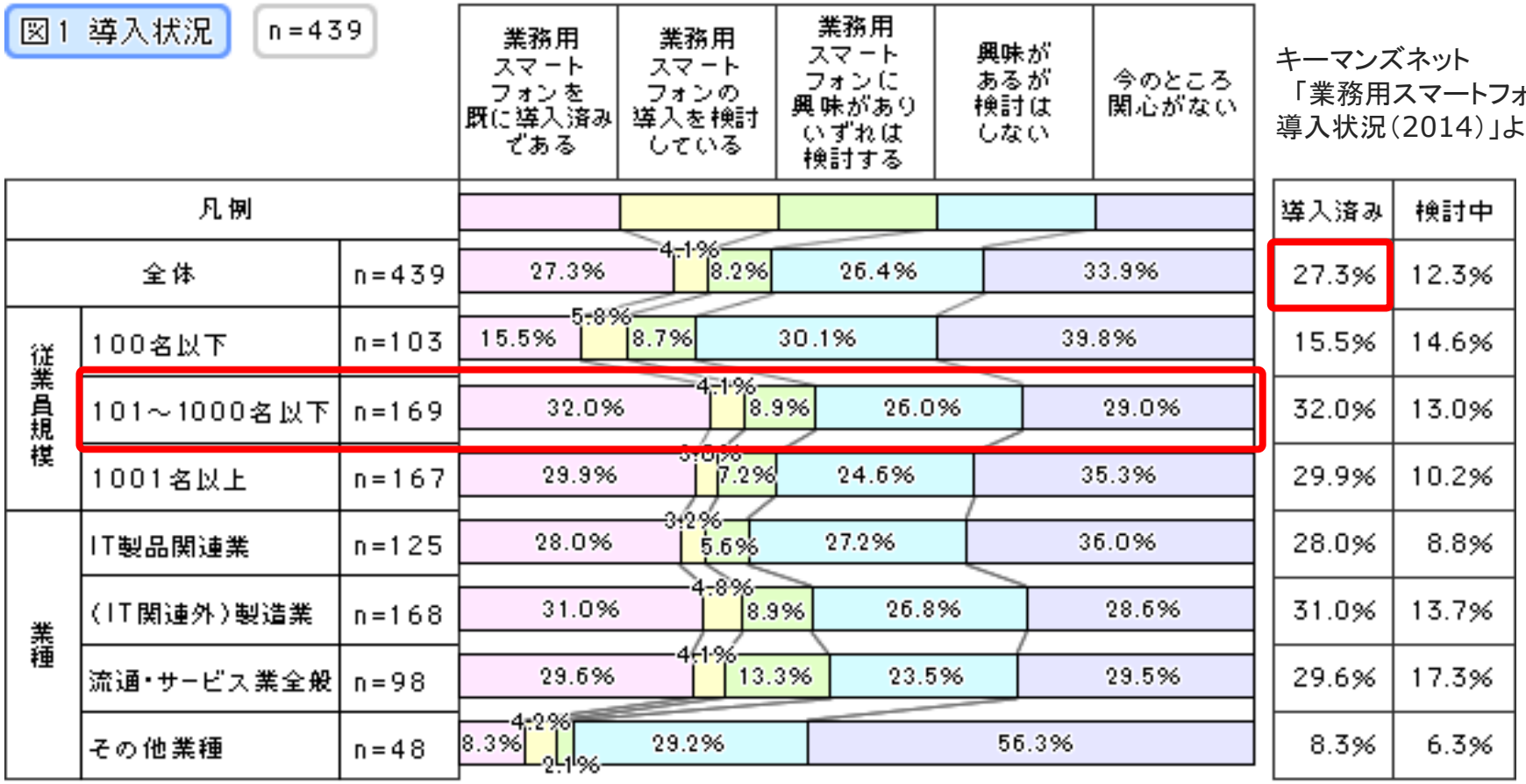
【アジェンダ】

1. はじめに
2. マルチデバイス対応画面レイアウト設計の考慮点
3. DataSnapを使用した3層構成アプリ開発手法
4. まとめ

１．はじめに

■ 企業のおけるスマートフォン導入状況

図1 導入状況　n＝439



			導入済み	検討中
全体		n＝439	27.3%	12.3%
従業員規模	100名以下	n＝103	15.5%	14.6%
従業員規模	101～1000名以下	n＝169	32.0%	13.0%
従業員規模	1001名以上	n＝167	29.9%	10.2%
業種	IT製品関連業	n＝125	28.0%	8.8%
業種	（IT関連外）製造業	n＝168	31.0%	13.7%
業種	流通・サービス業全般	n＝98	29.6%	17.3%
業種	その他業種	n＝48	8.3%	6.3%

キーマンズネット
「業務用スマートフォンの導入状況（2014）」より

- 全体で**27.3%**の企業が既に導入済。（2011年対比　19.6ポイント増）
- とくに従業員101～1000名の**中堅企業の伸び**が多い。（昨年比7.3ポイント増）。

本格的に業務用として、スマートデバイスを採用する動きが加速！

■ スマートデバイス活用の方向性

Step1

電話機能（社外、内線）
メール機能
スケジュール管理

デバイス機能活用

Step2

電子カタログ
グループウェア
専用端末置き換え
（タブレットPOSレジ等）

パッケージソフト活用

Step3

基幹システムとの連携
- 営業支援
- 現場業務
- 経営分析　…

独自ソフト開発

【基幹システムとの連携例】

取引先情報と連動した経路検索

バーコードを利用した検品処理

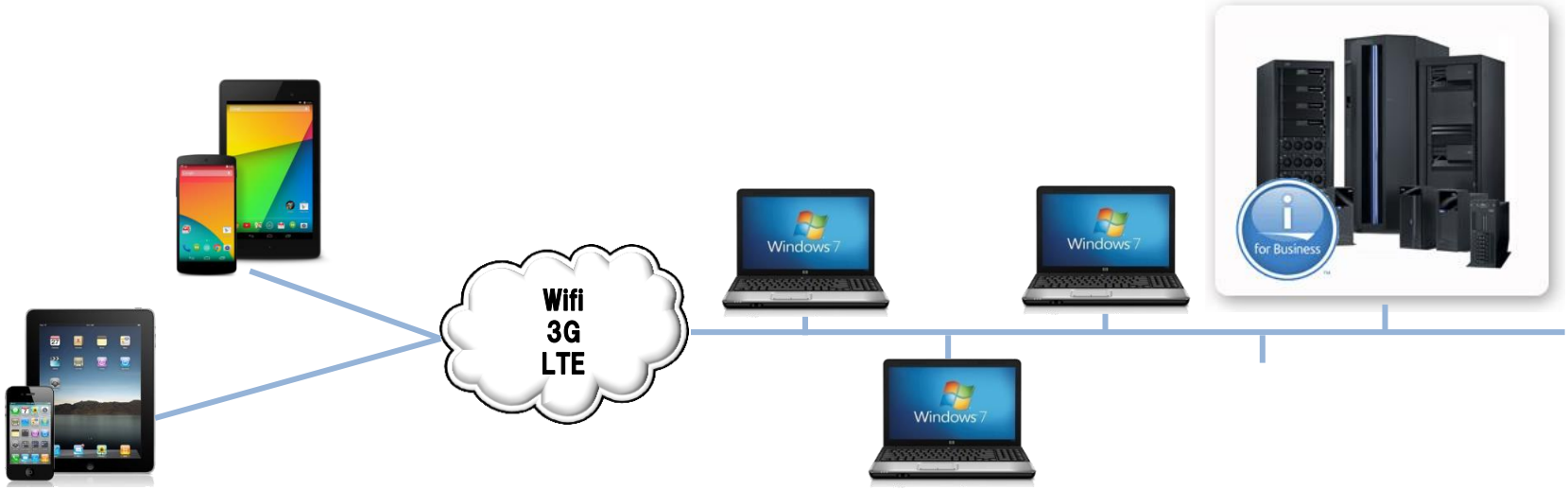
リアルタイムな売上分析

■ 今後のIBM i 業務システムの方向性

これまでのWindowsクライアントだけをターゲットにした開発

↓

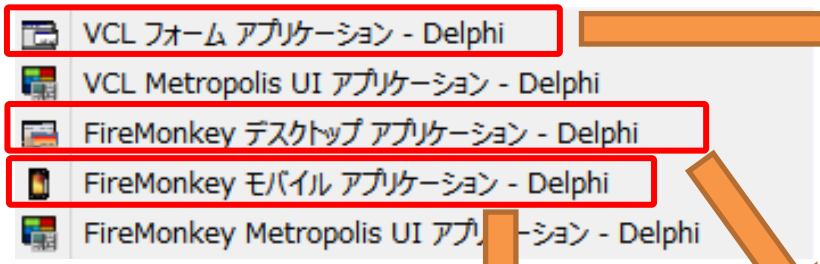
iOS/Androidといったスマートデバイスもターゲットとした開発

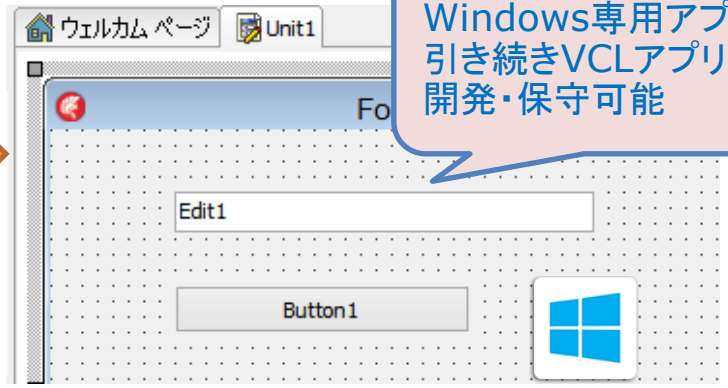


Delphi/400を使用すれば、今後のマルチデバイス化も対応可能！

■ Delphi/400によるマルチデバイス開発

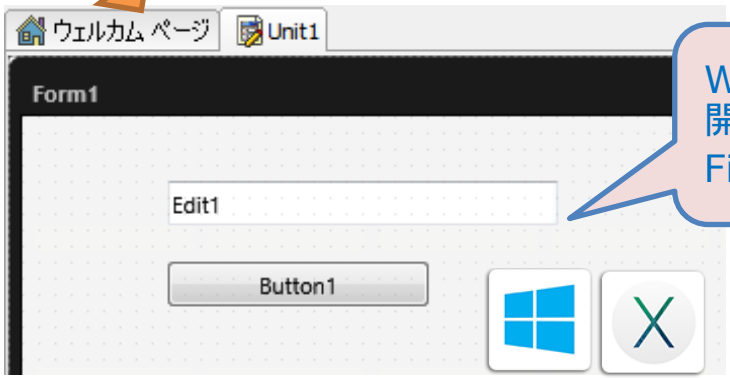
- VCL と FireMonkey





Windows専用アプリは、
引き続きVCLアプリで
開発・保守可能

iOS, Androidアプリが
開発可能な
FireMonkeyモバイル



Windows, Macアプリが
開発可能な
FireMonkeyデスクトップ

いずれも言語は全て
同じDelphi言語を使用

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Edit1.Text := 'Hello world';
end;
```

Delphi

マルチデバイス対応が一つのツール・言語で可能

■ C/S開発者にとってのマルチデバイス開発のポイント

- 画面（UI）
  - マルチデバイスに対応したFireMonkeyとは？
  - VCLとFireMonkeyとの違い
  - PCアプリとモバイルアプリの違い
  - モバイルアプリ画面設計の留意点
- 処理（BL）
  - 2層アプリと3層アプリの違い
  - DataSnapを使用したアプリケーション構築方法
  - IBMi上のデータ参照/更新処理
  - サービスアプリによる実行環境構築

主にVCLを使用したC/S型アプリケーションを作成している開発者に
マルチデバイス開発のポイントを紹介！

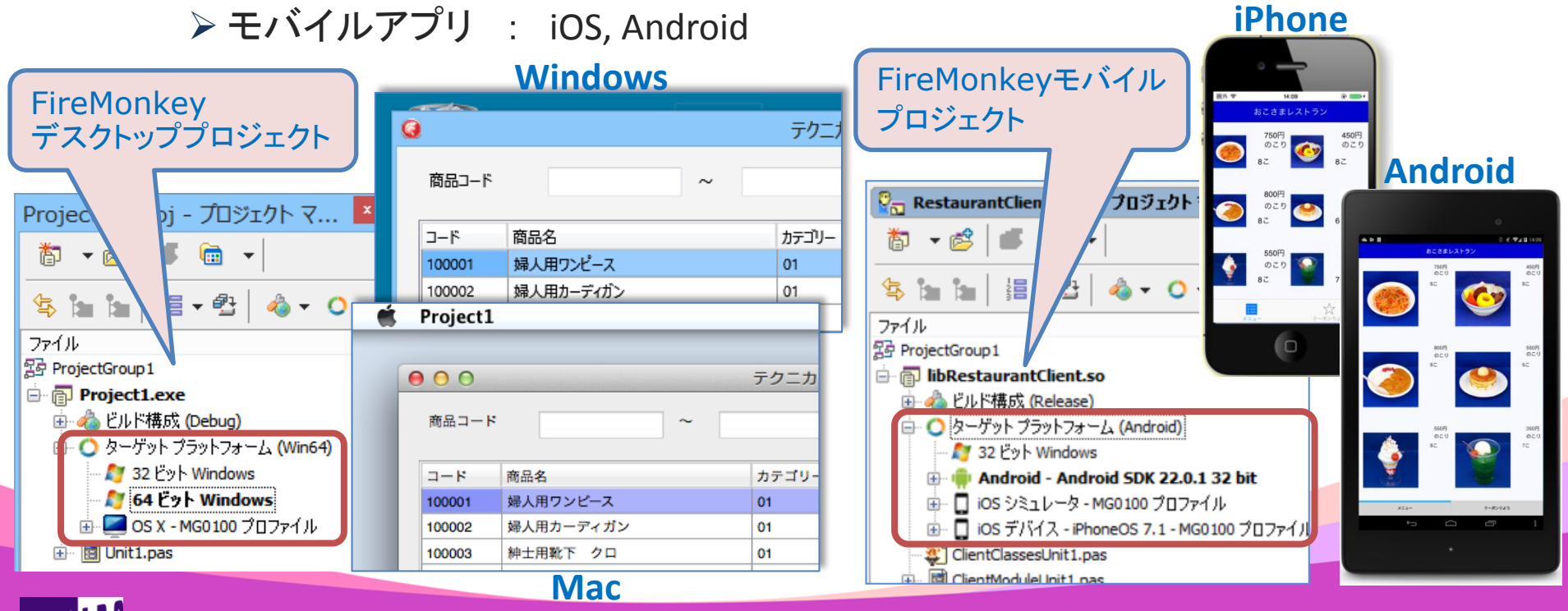
２．マルチデバイス対応
画面レイアウト設計の考慮点

FireMonkeyとは？

- マルチデバイスに対応したフレームワーク
  - Delphi/400 XE5では、Windows/Mac/iOS/Androidに対応
- WindowsAPIを使用して描画するVCLと違い、グラフィックス処理装置（GPU）へ直接描画
  - 3D表現やアニメーション効果等は、VCLより簡単に実現可能。
- VCL同様コンポーネントを使用した開発が可能
  - 但しVCLと同名のコンポーネントでも使用方法が異なる。
    - Top/Leftプロパティ → Positon. X, Y プロパティ
    - Captionプロパティ → Textプロパティ
- dbExpressやClientDataSetはFireMonkeyでも使用可能
  - データベースミドルウェアがWindowsのみの場合、dbExpressをWinows以外のOSで動作させることは不可。 → 3層構成が重要

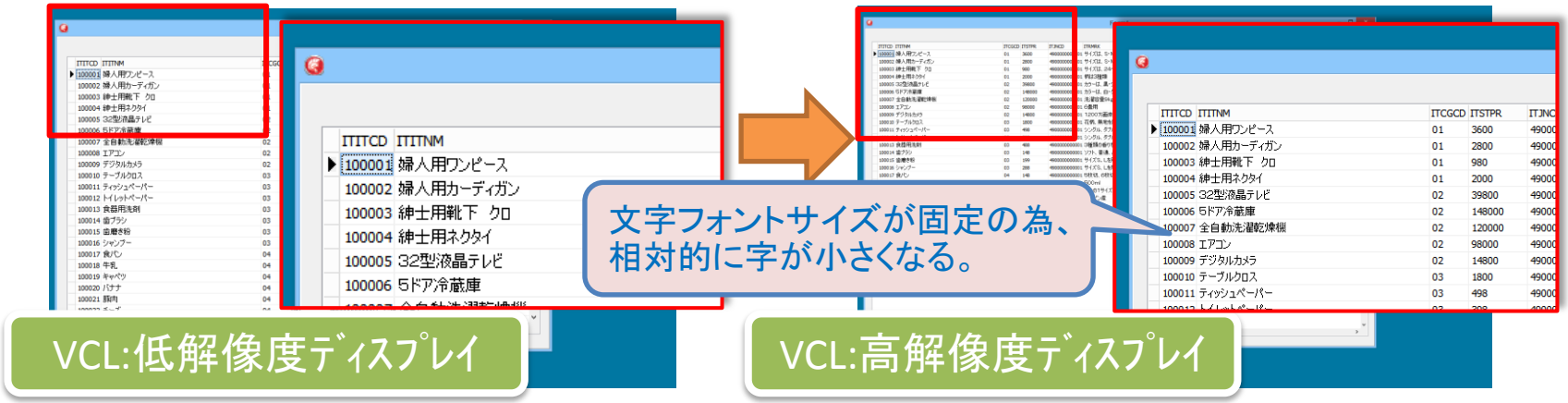
■ FireMonkeyの特長

- マルチデバイス（複数プラットフォーム）対応
  - ネイティブアプリ
    - ➢ Webアプリと比べ、操作性・レスポンス等優位
    - ➢ デバイスに搭載された各種機能にアプリからアクセス可能
  - 異なるプラットフォームでも単一のプロジェクト/ソースコードで開発可能
    - ➢ PCアプリ　：　Windows 32bit/64bit, MacOS
    - ➢ モバイルアプリ　：　iOS, Android



VCLとFireMonkeyとの違い

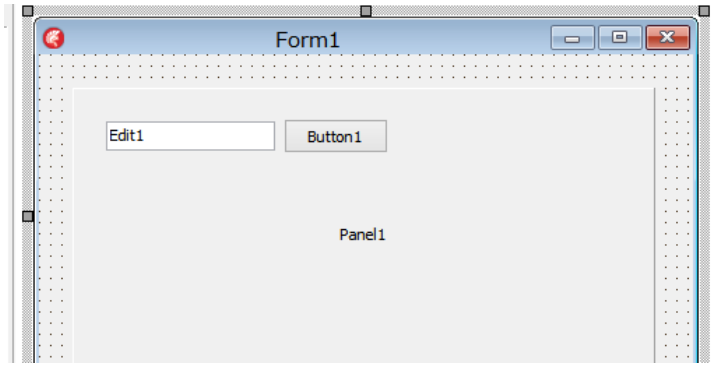
- FireMonkeyは高解像度対応が容易 (TScaledLayout)
  - VCLの場合、フォントやサイズがピクセル固定の為、ディスプレイが高解像度になると、フォームを大きくしても空白が広がるだけ。(相対的に字が小さくなる)

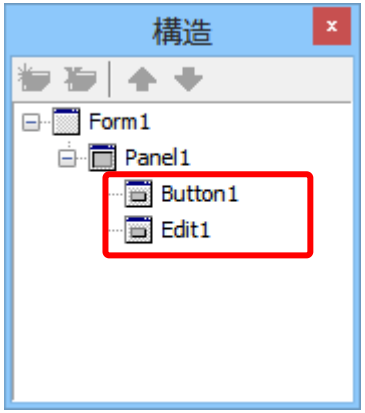


  - FireMonkey(FM)の場合、スケール変更が可能な為、高解像度ディスプレイでフォームを大きくした場合に、全体のサイズが広がる。(字が小さくならない)

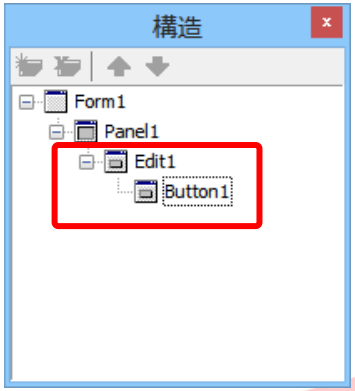
VCLとFireMonkeyとの違い

- コンポーネントの親子関係が自由
  - VCLの場合、TPanelやTScrollbox等コンテナコンポーネントのみ親子関係が成立。



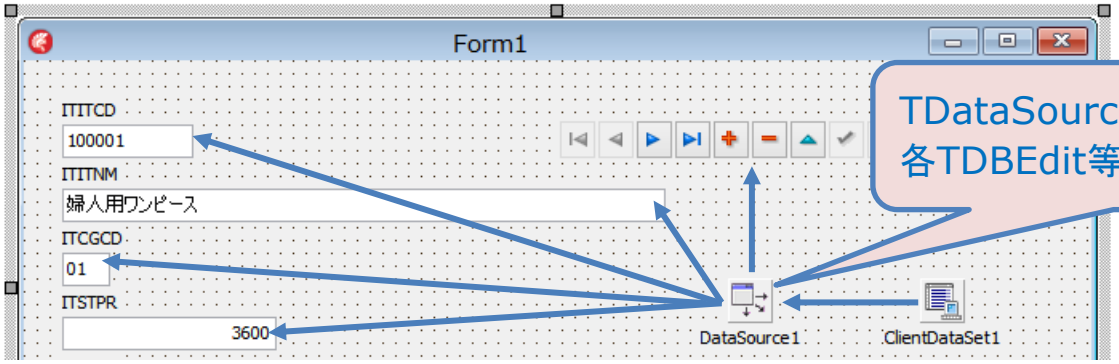


  - FireMonkeyの場合、ビジュアルコンポーネントの親子関係が自由に設定可能。

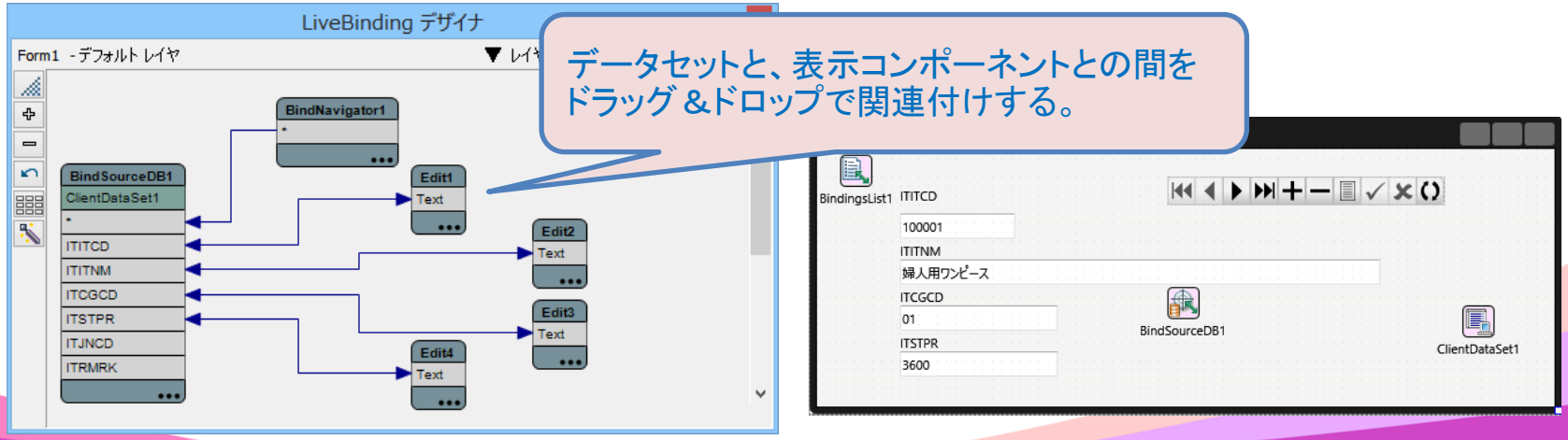


VCLとFireMonkeyとの違い

- TDBxxxxコンポーネントの有無
  - VCLの場合、TDBEditやTDBGrid等を使用してデータベース連結が可能。

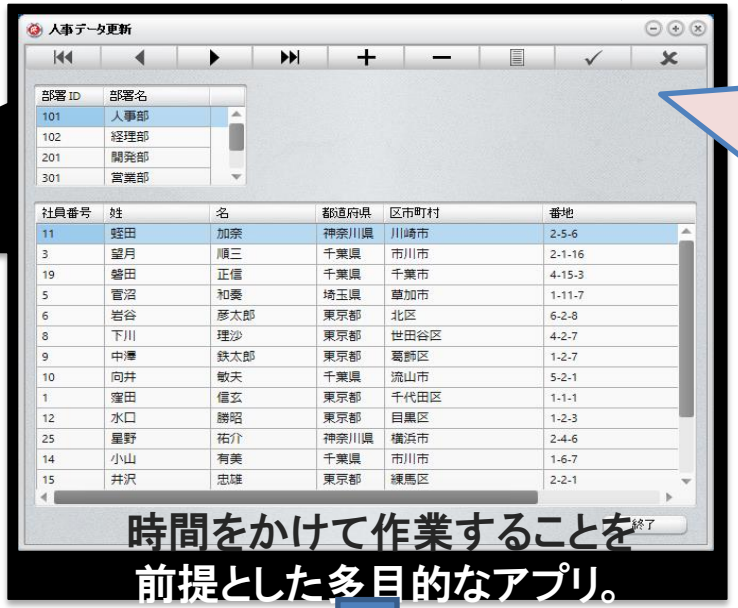


  - FireMonkeyの場合、TEditやTStringGrid等と項目コンポーネントとを LiveBinding で関連付け。



■ PCアプリとモバイルアプリの違い

PC

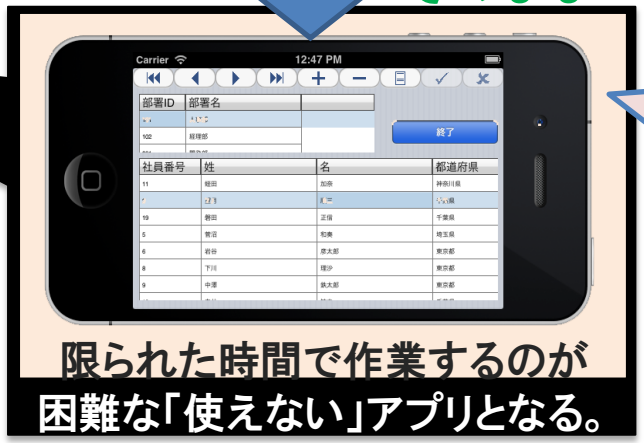


時間をかけて作業することを
前提とした多目的なアプリ。

オフィスでの使用が前提
- キーボード
- マウス
- 十分な画面サイズ
- リッチな通信環境

モバイル

そのまま「モバイル化」しても...



限られた時間で作業するのが
困難な「使えない」アプリとなる。

いつでも・どこでもが前提
- ソフトキーボード
- タッチ
- 限られた画面サイズ
- 不安定な通信環境

PCアプリの全ての機能をそのままモバイルで実現しても、結局使い勝手が悪い！

■ モバイルに最適なアプリとは？

■ 限られた画面サイズ/入力方法

- ☑ 1画面に情報を詰め込みすぎない
  (1機能＝1画面)
  - ➢ 部門選択 → 社員選択 → 社員編集
- ☑ 必要最小限の入力にする工夫
  - ➢ モバイルから編集が必要な物を取捨選択

■ デバイス機能の活用

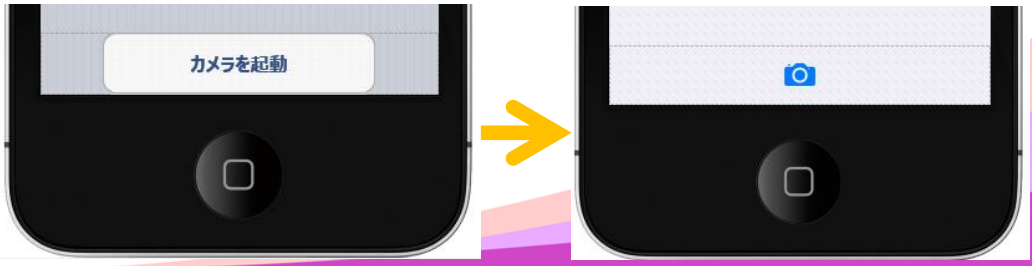
- ☑ カメラによる画像撮影
- ☑ マップやGPS、音声による入力省力化
- ☑ 電話やメールとの連携

モバイルアプリでは、
何を**選び**、何を**捨てる**か、
何を**新しい価値**として採用するか？を
明確にすることが重要！

■ モバイル画面設計のポイント

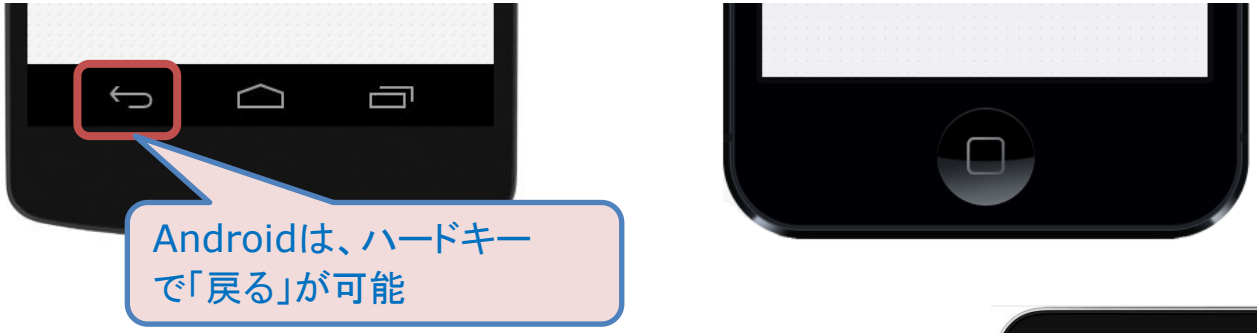
1. 使用するデバイスや画面サイズが限定できるか？
   1. 業務で使用するデバイスが限定できれば画面設計はシンプル。
      例）iPad専用アプリであれば 論理サイズ 768dp × 1024dpで固定できる。
      iPad/iPad2　：　1024ﾋﾟｸｾﾙ×768ﾋﾟｸｾﾙ
      iPad3/iPadAir：　2048ﾋﾟｸｾﾙ×1536ﾋﾟｸｾﾙ
   2. 機種が限定できない場合でも、論理サイズで設計する事で
      ある程度の解像度の違いは吸収可能。
   3. 実行時の縦横切り替えを許可するか、固定化するかを決定する。

2. シンプルな表現を意識する。
   1. テキスト/ボタンのサイズは大きくする。
   2. 表示する情報量は最低限にする。
      例）メール一覧：受信日時　【当日】時分のみ　【当年】月日のみ
   3. 表記をシンプルにする。
      例）アイコン表示



■ モバイル画面設計のポイント

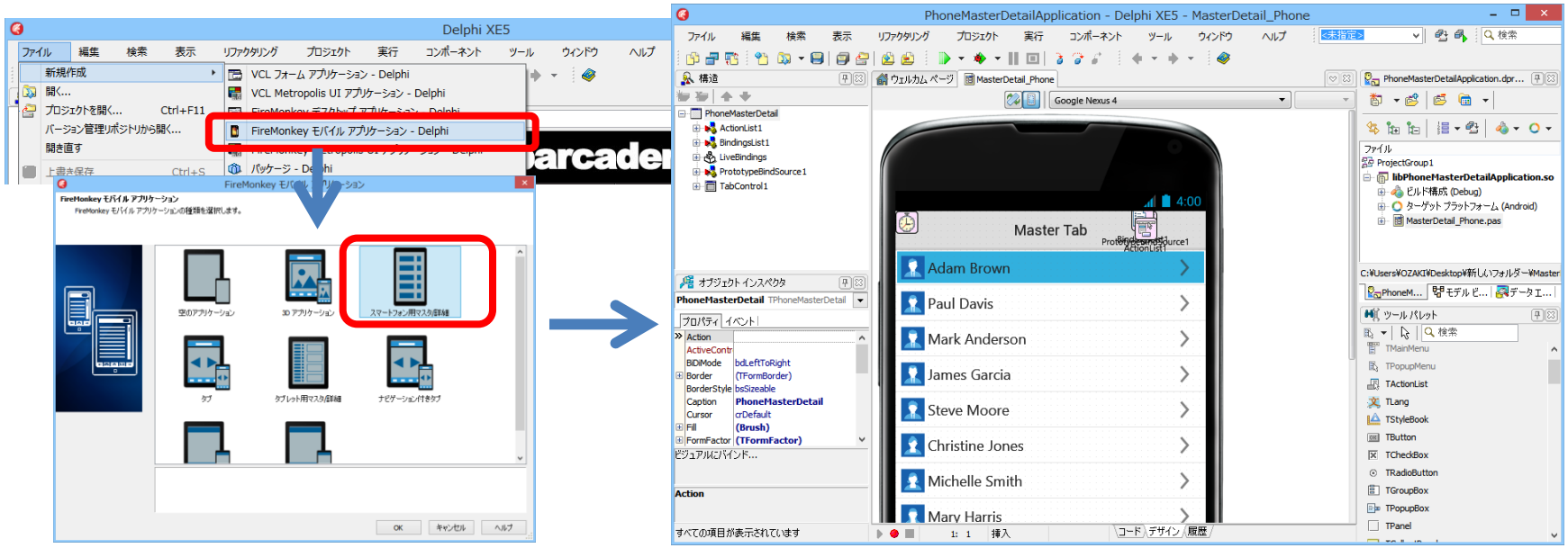
3. デバイスの違いを意識する
   1. iOSには「戻る」ボタンが無い。iOSの場合には画面上に「戻る」ボタンを配置する。



4. コンポーネントの配置に留意する。
   1. よく使用する機能は、下に配置する。
   2. 逆に「削除」等、誤タップをぎたいものは上は配置。
   3. 片手で操作しやすいエリアを意識する。
   4. ソフトキーボードの出現を考慮する。

■ iOS / Android デバイスの違いの考慮

- 新規作成から[スマートフォン用マスタ/詳細]を作成



- 実行結果
  - [iOSとAndroid]

■ iOS / Android デバイスの違いの考慮

- ソースを確認

```
procedure TPhoneMasterDetail.FormCreate(Sender: TObjec
begin
  { This defines the default active tab at runtime }
  TabControl1.ActiveTab := TabItem1;
{$IFDEF ANDROID}
  { This hides the toolbar back button on Android }
  BackButton.Visible := False;
{$ENDIF}
end;
```

【条件付コンパイル】
Androidの場合のみ有効

値	ターゲットプラットフォーム
IOS	iOS
ANDROID	Android
MSWINDOWS	Windows
MACOS	Mac OS X

```
procedure TPhoneMasterDetail.FormKeyUp(Sender: TObject; var Key: Word;
  var KeyChar: Char; Shift: TShiftState);
begin
  if Key = vkHardwareBack then
  begin
    if TabControl1.ActiveTab = TabItem2 then
    begin
      ChangeTabAction1.Tab := TabItem1;
      ChangeTabAction1.ExecuteTarget(Self);
      Key := 0;
    end;
  end;
end;
```

Androidの場合、
「戻る」キー押下時に、処理を実行。
（「戻る」キーの無いiOSでは
この部分は、実行されない。）

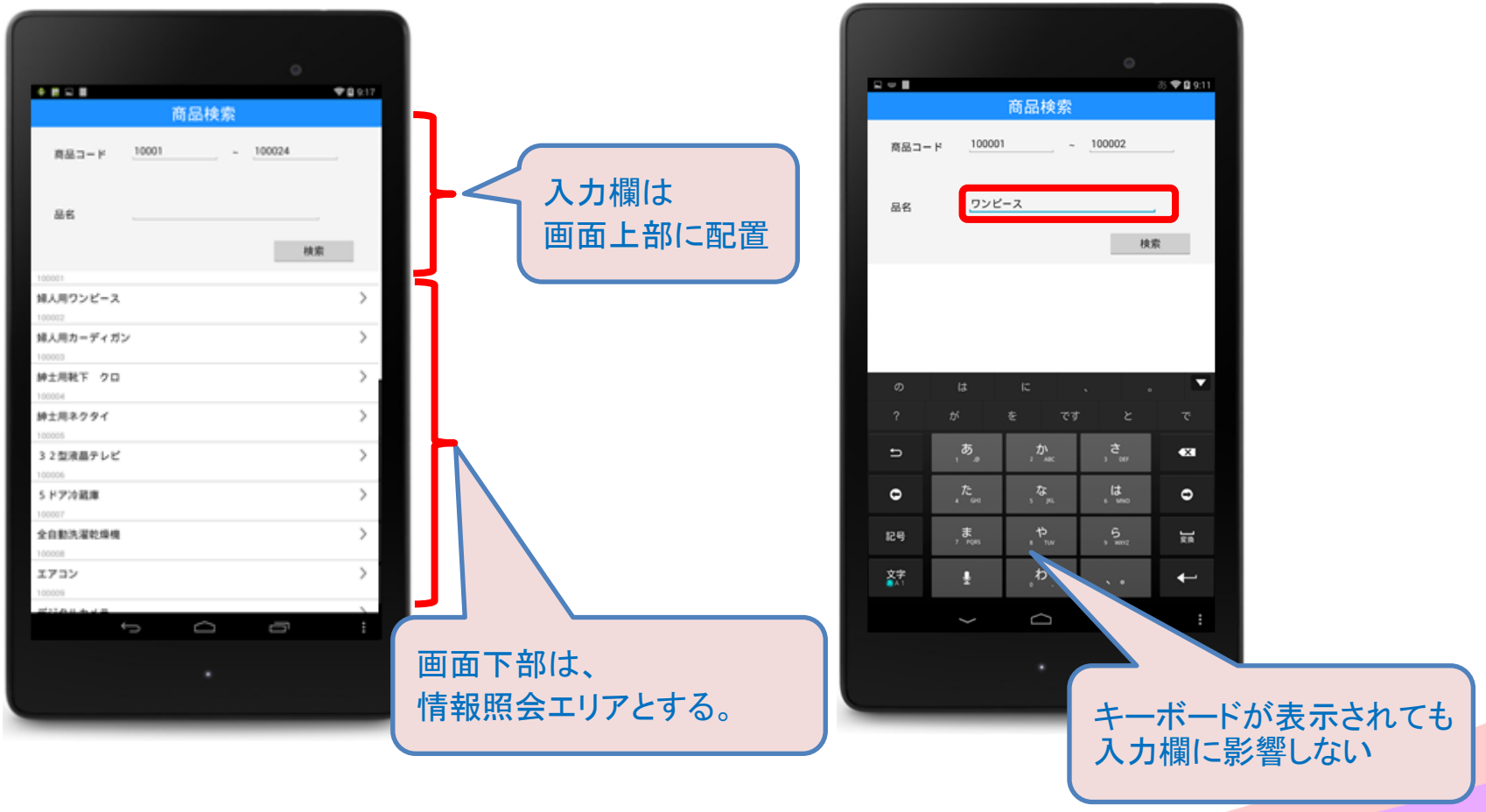
■ ソフトキーボードの考慮

- 一般的にモバイルデバイスは、キーボードを持たない為入力が必要な場面になるとソフトキーボードを使用する。

ソフトキーボードを使用する際の考慮点は？

■ ソフトキーボードの考慮

- 可能な限り、入力欄が必要な箇所は画面上部に配置する。



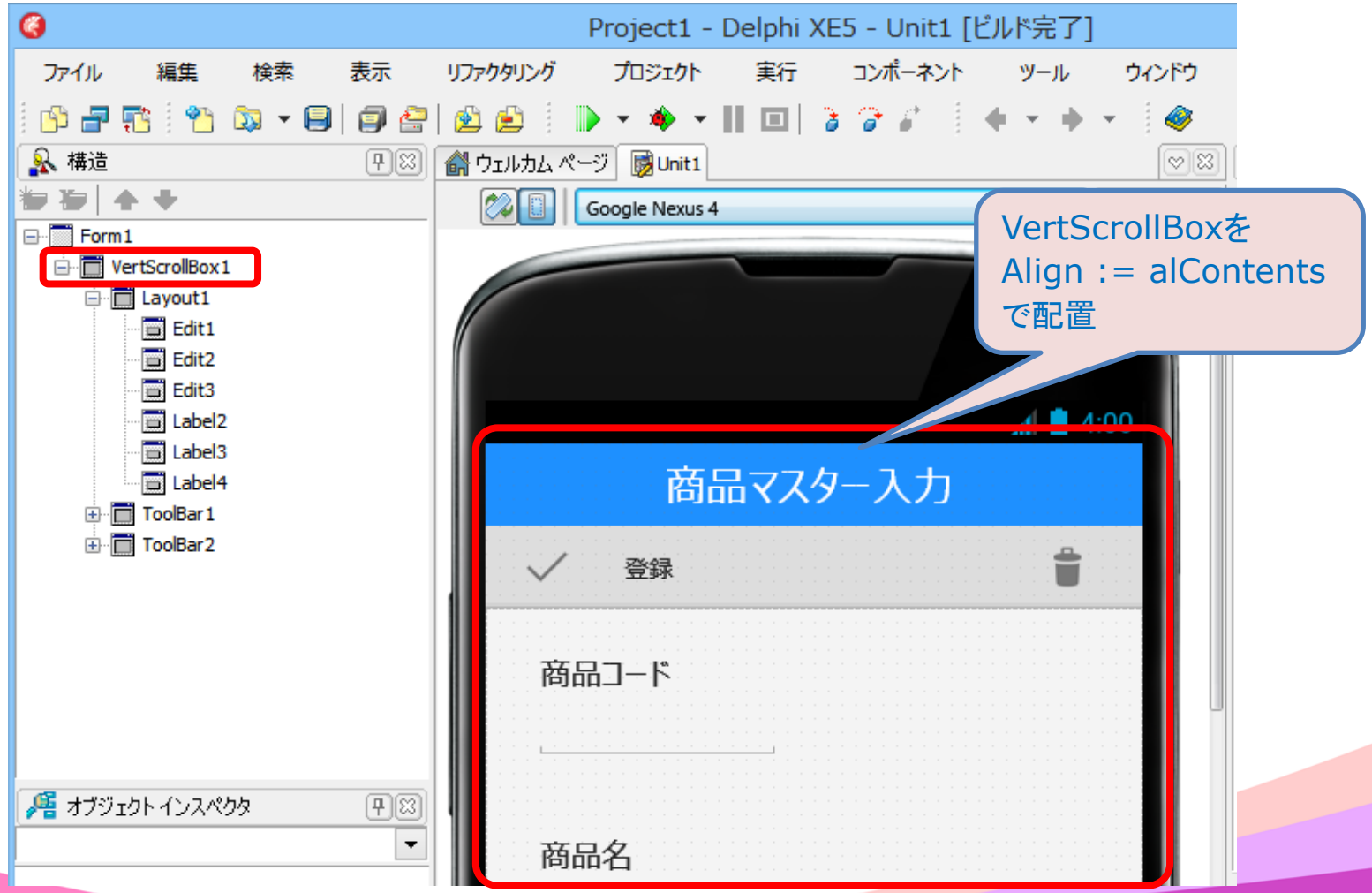
■ ソフトキーボードの考慮

- 入力画面等で、画面下部にも入力エリアがある場合

下部に入力欄がある場合は、どのような制御を行えばよいか？

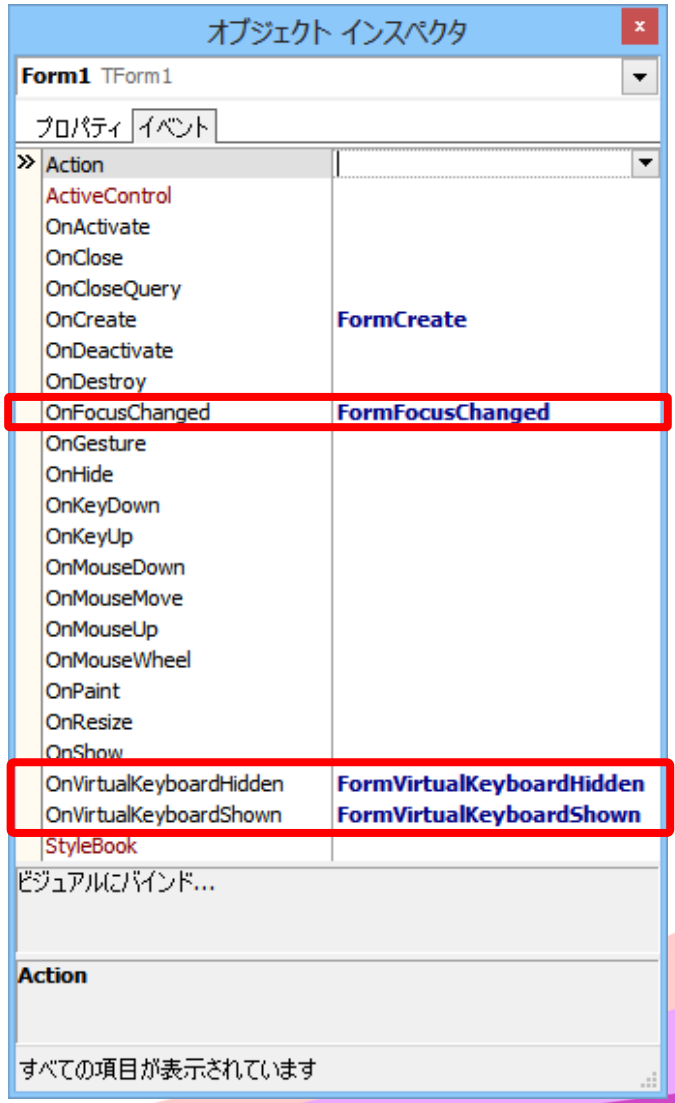
■ ソフトキーボードの考慮

- VertScrollBoxの利用



■ ソフトキーボードの考慮

- 実装の考え方
  - ソフトキーボード出現/終了のイベント(OnVirtualKeyboardShown/Hidden)
    - ➢ キーボードの出現位置を取得
  - フォーカス変更時(OnFocusChanged)
    - ➢ キーボードの出現時、キーボードの上端とフォーカスのあるコンポーネントの下端が被る位置となる場合に、差分に相当する分だけ、画面をスクロールする。
- Delphi XE5サンプルプログラム有
  - ➢ C:¥Users¥Public¥Documents¥RAD Studio¥12.0¥Samples¥FireMonkeyMobile¥Delphi¥ScrollableForm¥ScrollableFormDemo.dpr



■ ソフトキーボードの考慮

- 実行例
  - 画面下部コンポーネント選択時、画面がスクロール

３．DataSnapを使用した3層アプリの開発手法

Delphi/400 C/Sアプリケーション

Windows C/Sアプリ

Windows PC

Delphi/400
C/Sアプリ

CO400クライアント
モジュール

データベース接続用の
クライアントライブラリを使用

※IBMiに限らず、Oracleや
SQLServerも同様

DB I/O

IBM i

CL　RPG

DB2/400

モバイルアプリ

iOS / Android

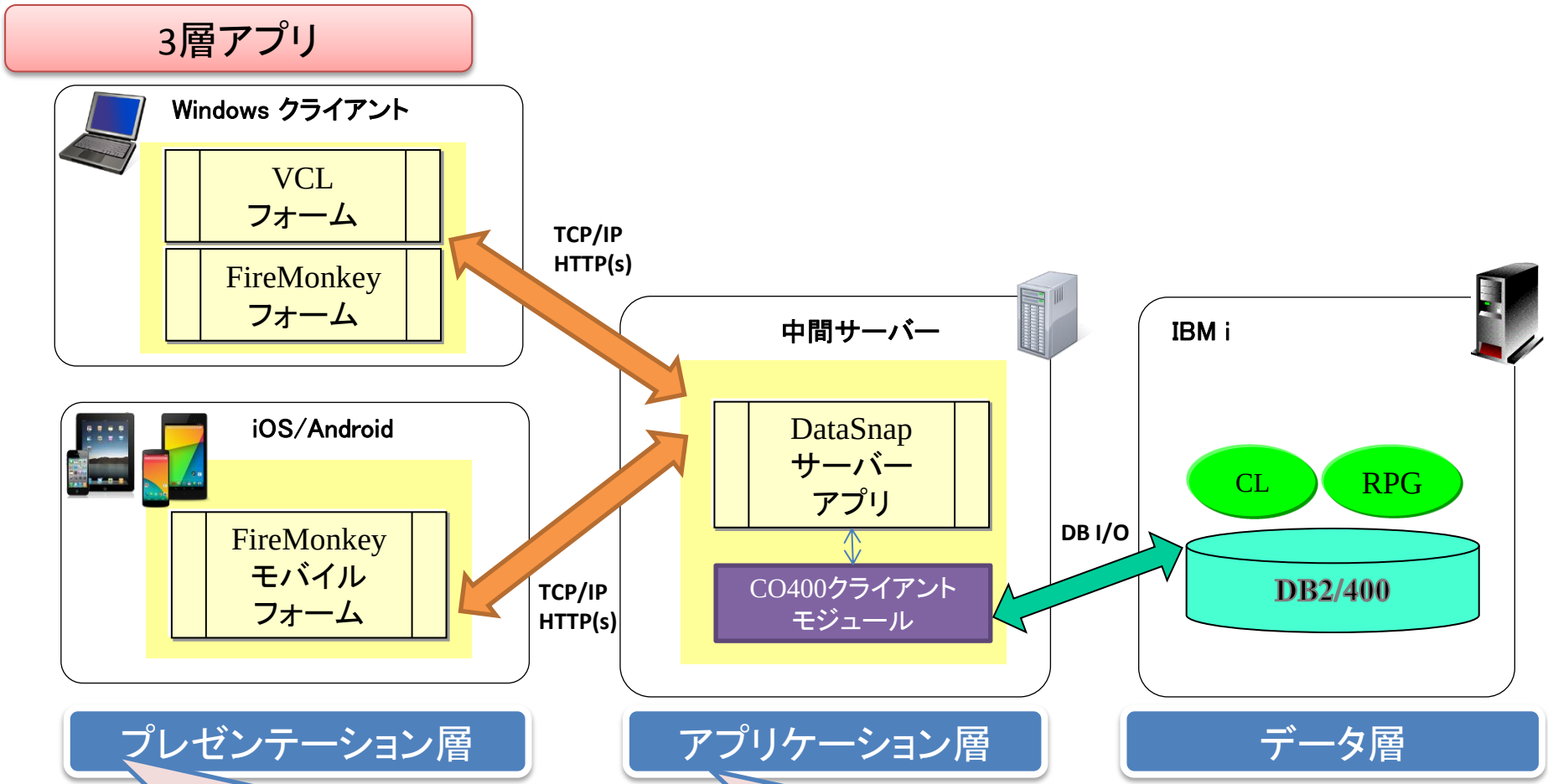
Delphi
モバイルアプリ

CO400クライアント
モジュール

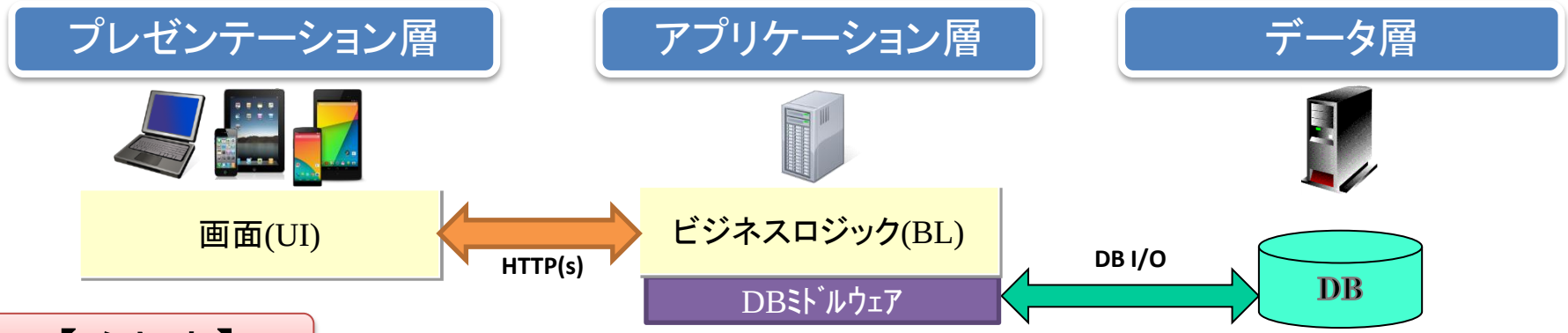
モバイルデバイス用
クライアントライブラリは
一般的に用意されていない

※IBMiに限らず、Oracleや
SQLServerも同様

DataSnapによるアプリの3層化



DataSnapによる３層化のメリットと留意点



【メリット】

- ビジネスロジック(BL)の仕様変更時、画面(UI)に影響しないものは、[アプリケーション層]のプログラムのみ変更すれば良い。
- 異なる画面(UI)であっても同じビジネスロジック（BL）が使用できる。
- プレゼンテーション層のプログラムはDBミドルウェア等が不要な為、ネイティブアプリであっても導入が比較的容易。（アプリのインストールのみで良い）
- [プレゼンテーション層]と[アプリケーション層]の通信にSSLが使用できる。

【留意点】

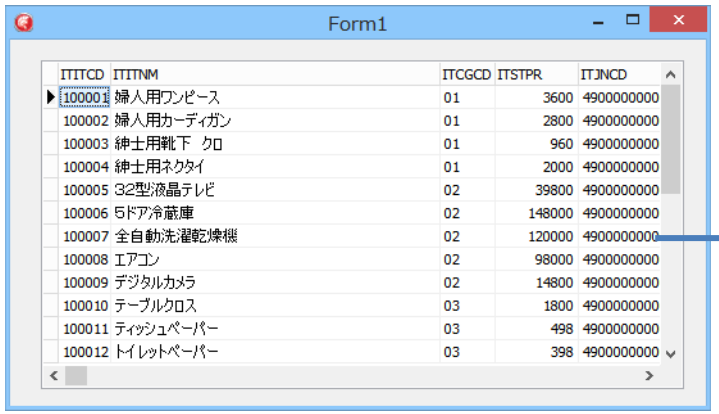
- ビジネスロジックにBDEは使用不可。（dbExpressを使用）
- QTEMPの使用は不可。（ファイルメンバーにより代替）

DataSnapアクセスの基本

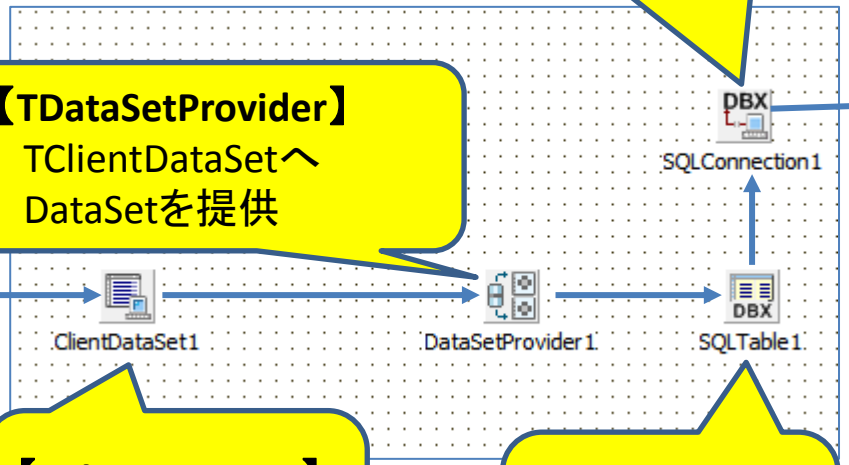
- C/Sアプリの場合 (dbExpress)

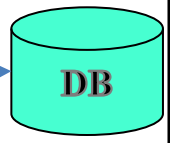
Delphi/400
C/Sアプリ

TForm



TDataModule





【TSQLConnection】
データベース接続を管理

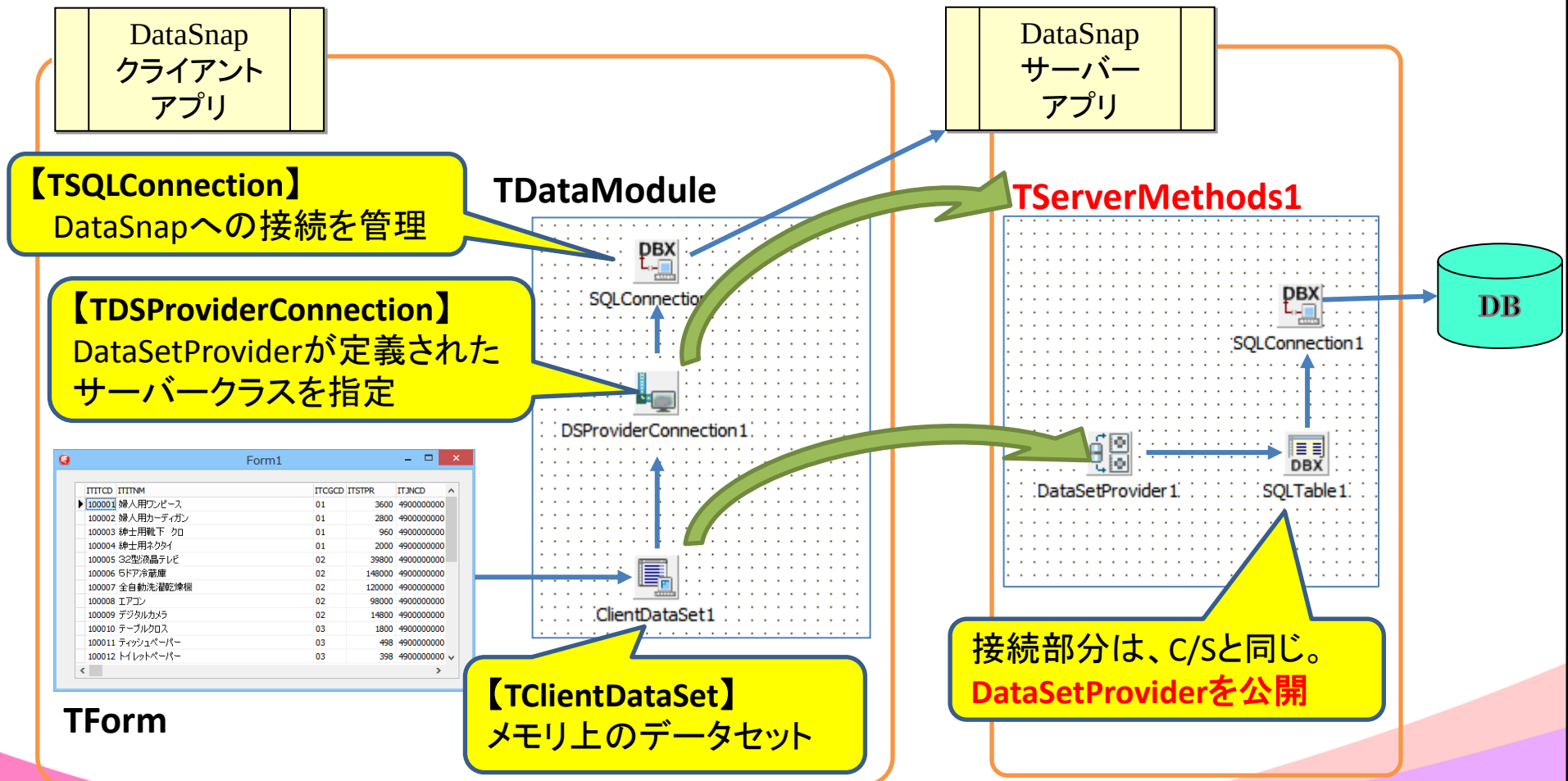
【TDataSetProvider】
TClientDataSetへ
DataSetを提供

【TClientDataSet】
メモリ上のデータ
セット

【TSQLTable】
【TSQLQuery】
データ取得

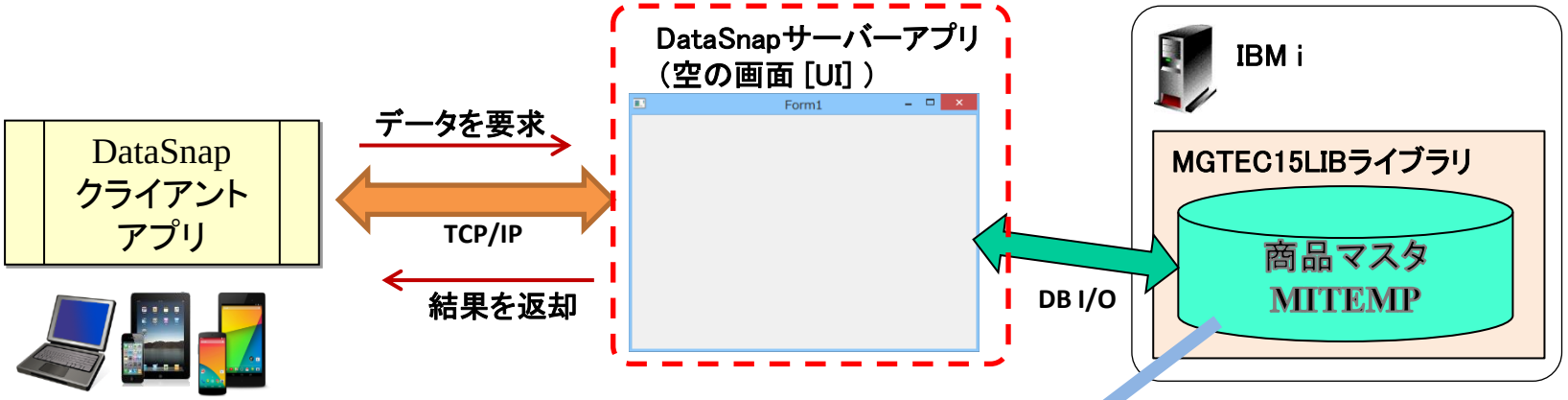
■ DataSnapアクセスの基本

- DataSnapの場合



DataSnapアプリの作成

- DataSnapサーバーを経由した商品マスタの参照
  - IBMi(AS/400)にDB接続
  - クライアントからTCP/IPにて接続を受け付け
  - クライアントからの要求に対し [商品マスタ(MITEMP)]より データを抽出し、クライアントへ結果を受け渡す



DSPFMT　　レ コ ー ド 設 計 書　　日付 14/10/22 時刻 14:15:01

物理ファイル	MGTEC15LIB/MITEMP	様式名	MITEMR	レコード長	270
様式記述	商品マスタ				

5= 詳細

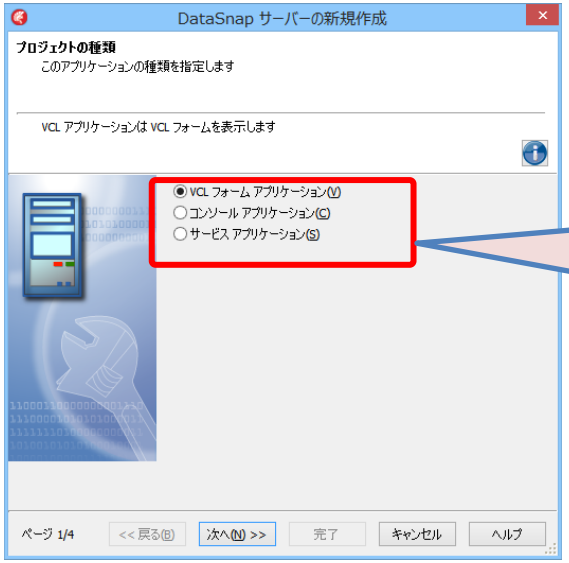
選択	項目名	桁数	属性	キー順	開始	終了	テキスト記述／欄見出し
_	ITITCD	6	A	1 ANN	1	6	商品コード
_	ITITNM	42	O		7	48	商品名称
_	ITCGCD	2	A		49	50	商品カテゴリ
_	ITSTPR	9 0	P		51	55	標準単価
_	ITJNCD	13	A		56	68	ＪＡＮコード
_	ITRMRK	202	O		69	270	商品備考

DataSnapアプリ
サーバー作成手順　：　P34-36
クライアント作成手順：　P37-39

■ DataSnapサーバー　作成手順（1）

- [ファイル]→[新規作成]→[その他]より「DataSnap Server」を選択

- [DataSnapサーバー]ウィザードを使用して定義を行う
  - （ステップ1） プロジェクトの種類

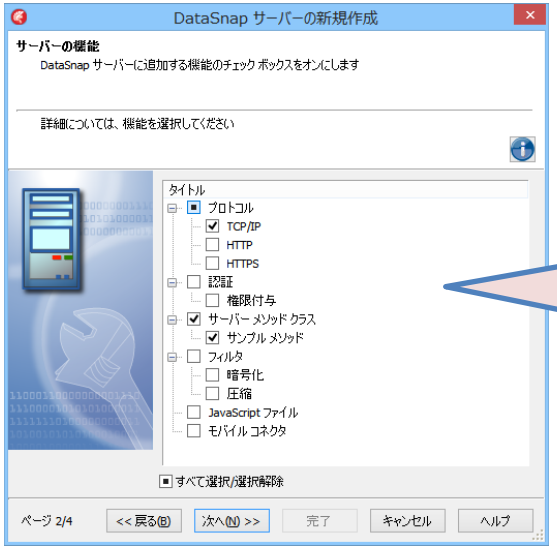


開発時は、通常「VCLフォームアプリケーション」を選択。（単体で実行可能な形式）

実際のサーバーで稼働させる場合は、「サービスアプリケーション」を選択する。

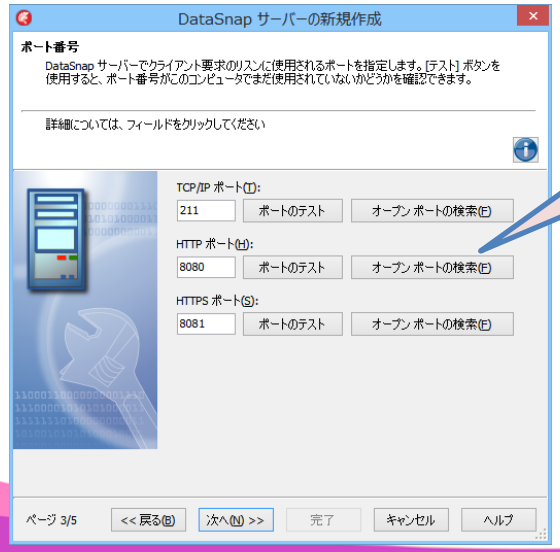
DataSnapサーバー　作成手順（2）

- （ステップ2） サーバーの機能



通信に使用するプロトコルや、認証機能の有無等を定義。
プロトコル： TCP/IP、HTTP(s)
認証 ： DataSnapサーバーにユーザー/パスワードを設定。

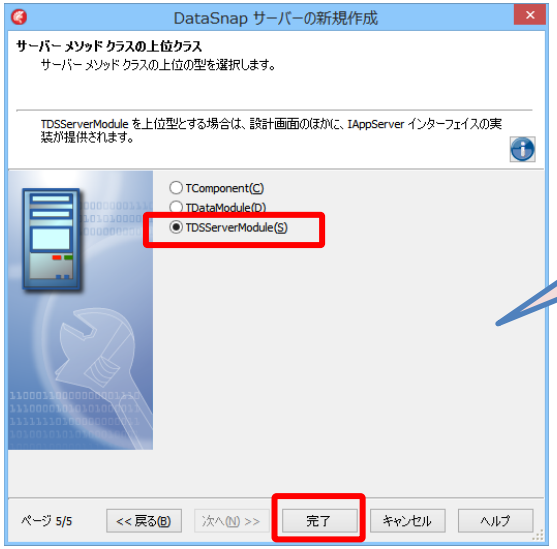
- （ステップ3） ポート番号



プロトコルごとに利用するポート番号を指定。

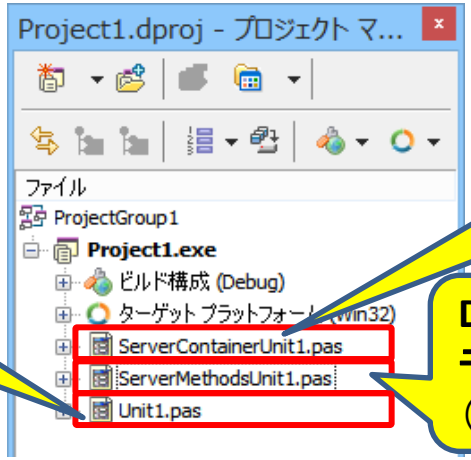
■ DataSnapサーバー　作成手順（3）

- （ステップ4） サーバーの機能



全てコードのみ（コンポーネントを使用しない場合）は、[TComponent]で良いが、通常は、[TDSServerModule]を選択する。

- アプリケーション雛形生成



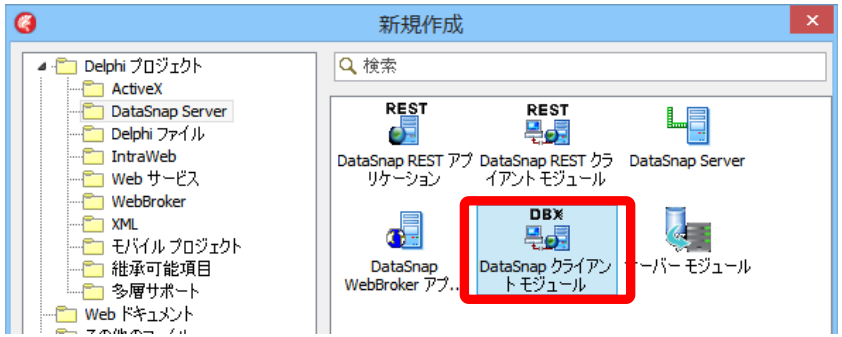
DataSnapサーバーの管理用モジュール（プロジェクトの種類により異なる）

表示フォーム（VCLアプリの場合のみ）

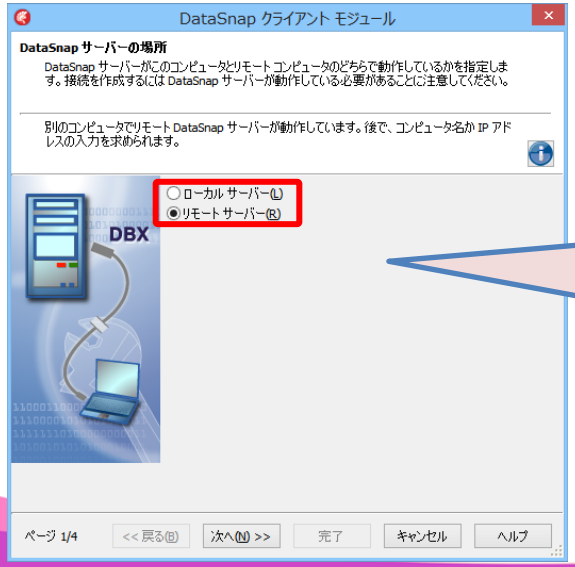
DBアクセス処理やサブルーチンを記述するモジュール（プロジェクトの種類が異なっても同じ）

■ DataSnapクライアント　作成手順（1）

- VCL あるいは　FireMonkey (デスクトップ / モバイル)アプリケーション作成
- [ファイル]→[新規作成]→[その他]より[DataSnap クライアントモジュール]を選択



- [DataSnapクライアントモジュール]ウィザードを使用して定義を行う
  - （ステップ1） DataSnapサーバーの場所

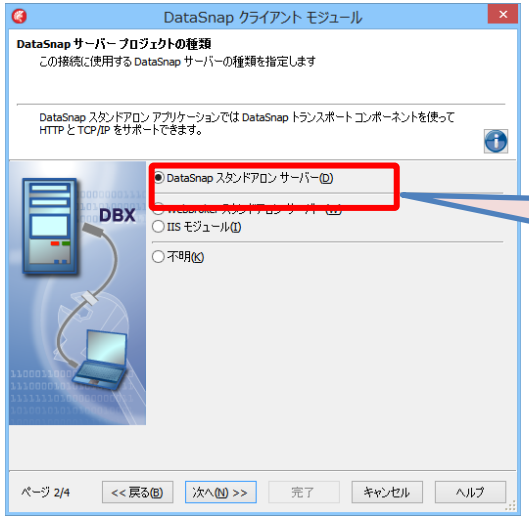


通常は、「リモートサーバー」を指定。

DataSnapサーバとクライアントを同一PCで実行する場合は、「ローカルサーバー」でよい。

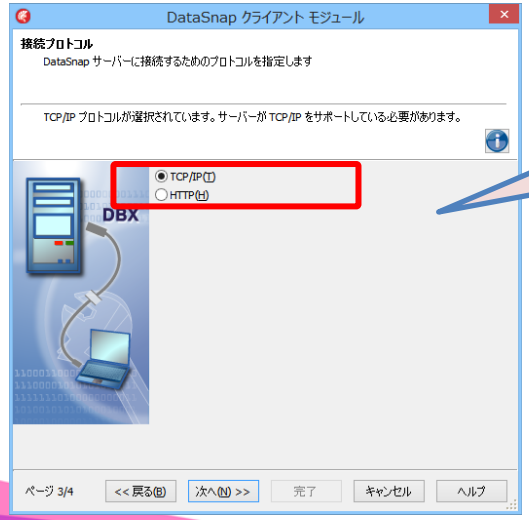
DataSnapクライアント　作成手順（2）

- （ステップ2） DataSnapサーバープロジェクトの種類



DataSnapスタンドアロンサーバーを指定。

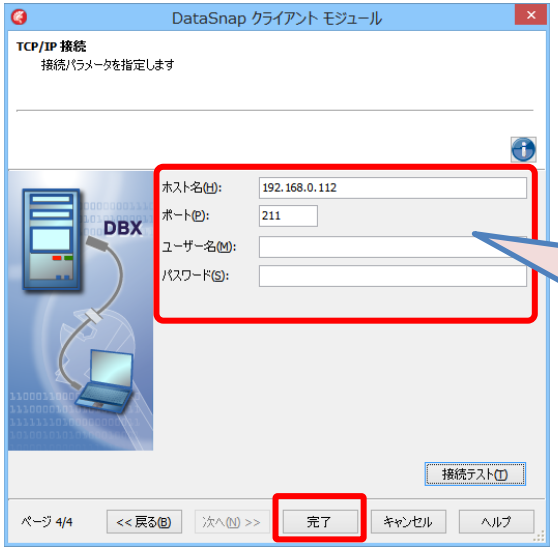
- （ステップ3） 接続プロトコル



サーバー作成時に指定したプロトコルを指定。

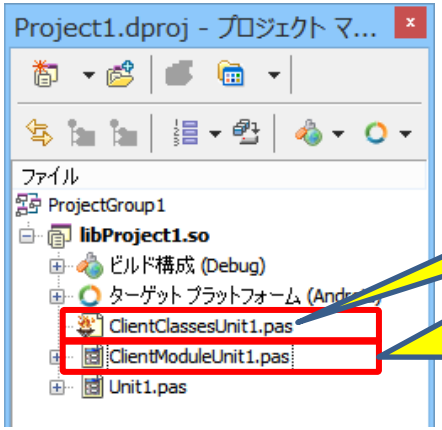
■ DataSnapクライアント　作成手順（3）

- （ステップ4） サーバーの機能



DataSnapの接続情報を登録して、接続テストを行う。
【TCP/IPの場合】
ホスト名： DataSnapサーバーのIPアドレス
ポート ： 使用するTCPポート番号
ﾕｰｻﾞｰ/ﾊﾟｽﾜｰﾄﾞ： 認証ありの場合指定。

- アプリケーション雛形生成



DataSnapサーバーで定義された
サブルーチンの呼び出し定義（自動生成）

DataSnapサーバーへ接続してTClientDataSet等を定義するモジュール

IBMiデータの更新方法

- (その1) データセットを直接操作
  - dbExpressでのデータセット編集処理 （C/Sアプリでの例）

```
procedure TdmMain.DataUpdate(strCode, strName: String);
begin
  with ClientDataSet1 do
  begin
    //クライアントデータセットの編集
    Append;
    FieldByName('ITITCD').AsString := strCode;
    FieldByName('ITITNM').AsString := strName;
    Post;

    //編集内容の適用
    ApplyUpdates(-1);
  end;
end;
```

dbExpressは単方向データセットの為、C/Sアプリの場合でも、通常ClientDataSetを使用して処理を行う。

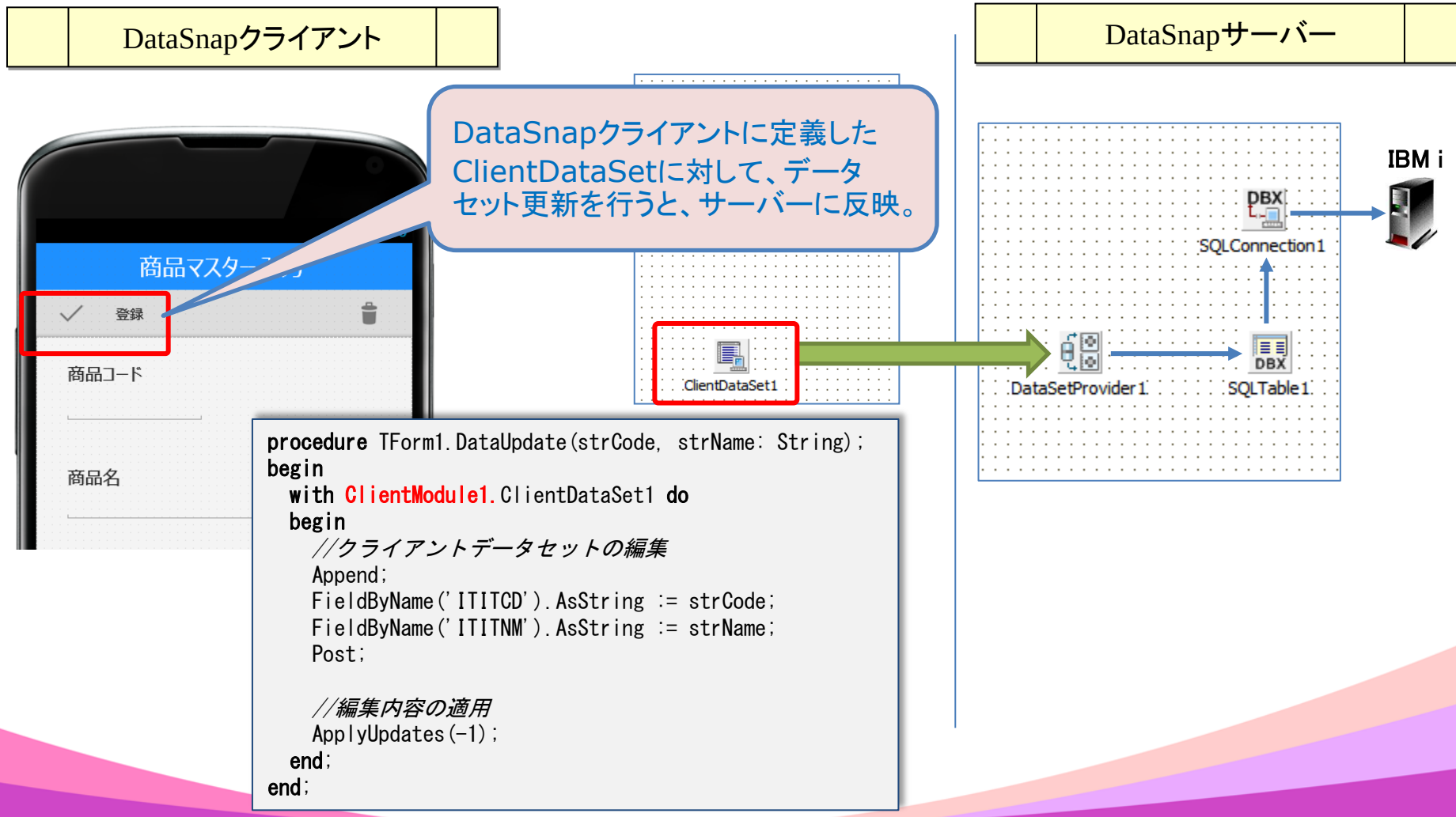
AppnedやEditでデータセットを編集し、Postで確定する。

データセットの編集内容をデータベースサーバーに適用。

DataSnapクライアントアプリもTClientDataSetを使用する為、C/Sアプリと同じ手順でデータセットの操作・更新が可能！

DataSnapでのデータセット更新

- DataSnapクライアントから直接TClientDataSetに更新。




```
procedure TForm1.DataUpdate(strCode, strName: String);
begin
  with ClientModule1.ClientDataSet1 do
  begin
    //クライアントデータセットの編集
    Append;
    FieldByName('ITITCD').AsString := strCode;
    FieldByName('ITITNM').AsString := strName;
    Post;

    //編集内容の適用
    ApplyUpdates(-1);
  end;
end;
```

IBMiデータの更新方法

- (その2) SQLを使用した更新処理
  - dbExpressでのSQL編集処理 （C/Sアプリでの例）

```
procedure TdmMain.DataSQLUpdate(strCode, strName: String);
begin
  with SQLQuery1 do
  begin
    //SQL作成
    SQL.Text := 'INSERT INTO MITEMP (ITITCD, ITITNM) ' +
                'VALUES (:ITITCD, :ITITNM) ';
    //値のセット
    ParamByName('ITITCD').AsString := strCode;
    ParamByName('ITITNM').AsString := strName;
    //SQL実行
    ExecSQL;
  end;
end;
```

dbExpressでSQLを実行する場合は、TSQLQueryを使用する

SQL文を作成し、パラメータなどをセットする。

更新SQLを実行。

DataSnapクライアントにはTClientDataSetしかない為、SQLは発行不可。

どうすれば良いか？

DataSnapへのサブルーチン呼出

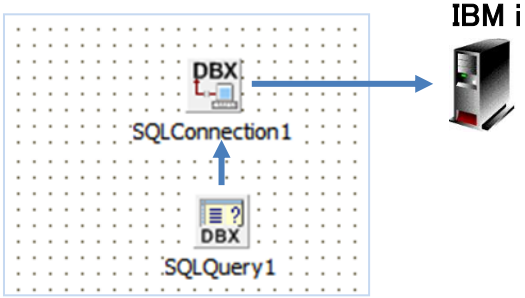
- DataSnapサーバーに用意したサブルーチンを
  クライアントから呼び出す。

	DataSnapクライアント	

	DataSnapサーバー	

DataSnapサーバーの
[DataSQLUpdate]サブルーチンを
呼び出せばよい。

[DataSQLUpdate]サブルーチン




```
procedure TServerMethods1.DataSQLUpdate(strCode, strName: String);
begin
  with SQLQuery1 do
  begin
    //SQL作成
    SQL.Text := 'INSERT INTO MITEMP (ITITCD, ITITNM) ' +
                'VALUES (:ITITCD, :ITITNM) ';
    //値のセット
    ParamByName('ITITCD').AsString := strCode;
    ParamByName('ITITNM').AsString := strName;
    //SQL実行
    ExecSQL;
  end;
end;
```

ServerMethodsユニットの定義

- DataSnapサーバー側は、サブルーチンを追加すれば良い

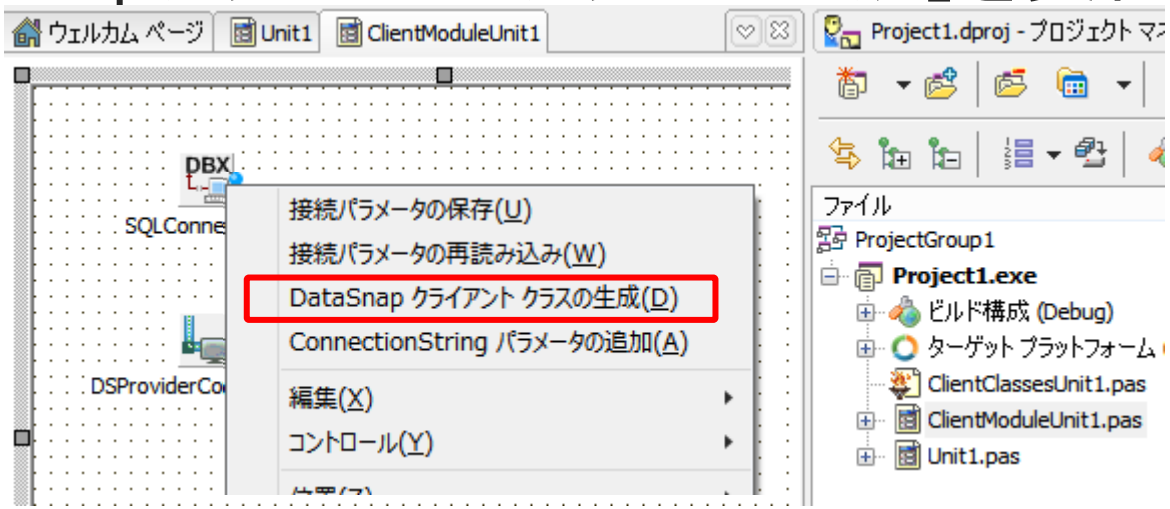
```
unit ServerMethodsUnit1;

interface

uses System.SysUtils,  …  ;

type
  TServerMethods1 = class(TDSServerModule)
  private
    { private 宣言 }
  public
    { public 宣言 }
    function EchoString(Value: string): string;
    function ReverseString(Value: string): string;
    procedure DataSQLUpdate(strCode, strName: String);

  end;
```

DataSnapサーバーにサブルーチンを追加した場合、

DataSnapクライアントはどうすればよいか？

■ DataSnapクライアントクラスの生成

- DataSnapクライアントのSQLConnectionより、「DataSnapクライアントクラスの生成」を実行

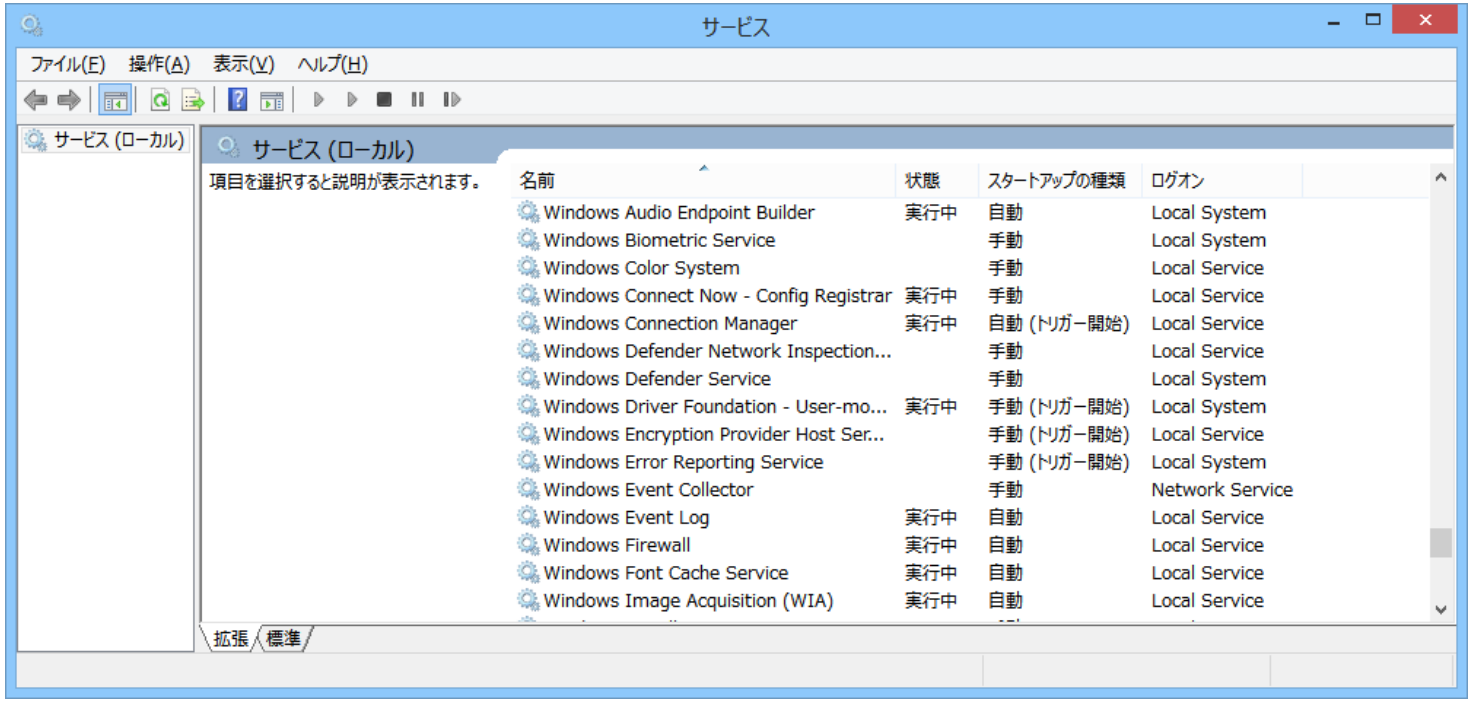


- 実行後、下記のようにサブルーチンを利用可能

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //サーバーの更新処理を実行
  ClientModule1.ServerMethods1Client.DataSQLUpdate(Edit1.Text,
    Edit2.Text);
end;
```

DataSnapサービスアプリ

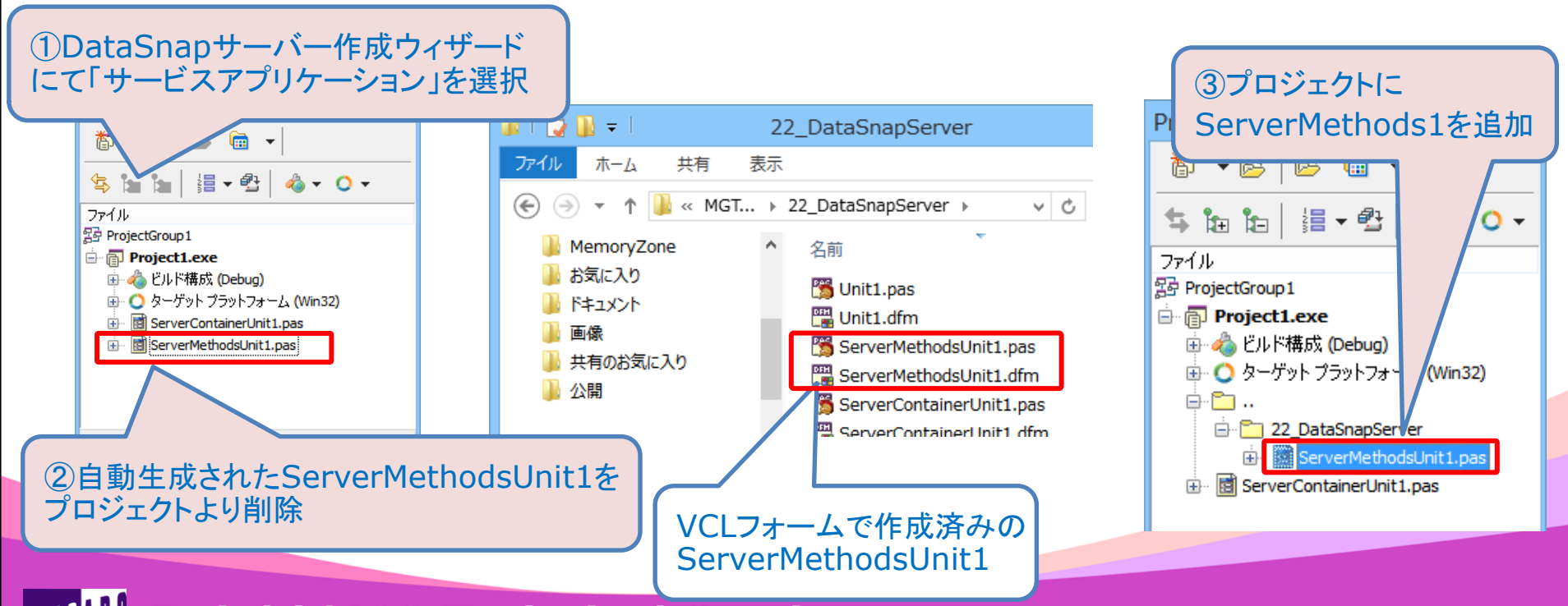
- サービスアプリとは？
  - ログオンせずとも実行可能。
  - Windows動作中であれば、バックグラウンドで常に実行可能。



DataSnapサーバーアプリは、最終的にサービスアプリに変更可能！

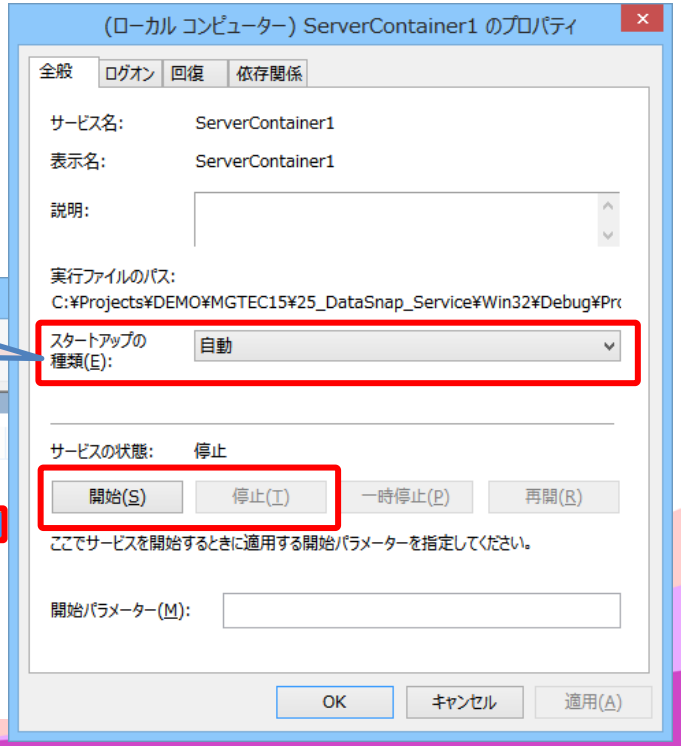
■ DataSnapサービスアプリ

- 変更手順
  - VCLフォームアプリとして作成したDataSnapサーバーアプリケーションと異なるフォルダに、「サービスアプリケーション」として新規作成。
  - 自動生成された[ServerMethodsUnit1.pas]を削除。
  - VCLフォームとして作成したDataSnapサーバーアプリケーションの[ServerMethodsUnit1.pas]をプロジェクトに追加。



■ DataSnapサービスアプリ

- サービスへの登録/解除方法
  - コマンドプロンプトより操作する。
    - サービス登録　：　作成exeを　[/install]オプションをつけて実行
    - サービス解除　：　作成exeを　[/uninstall]オプションをつけて実行



４．まとめ

■ まとめ

- マルチデバイス対応画面レイアウト設計
  - **FireMonkey**
    - マルチデバイス　ネイティブアプリ
    - 単一プロジェクトで複数プラットフォーム対応
    - スケールの変化に対応した画面が作成可能
    - LiveBindingによるデータセットと画面との連結
    - PCアプリとモバイルアプリ
    - モバイル画面の留意点
- 3層アプリの開発手法
  - **DataSnap**
    - 2層アプリと3層アプリの違い
    - DataSnapサーバー/クライアント作成方法
    - 2つの更新処理の実装方法
    - サービスアプリ

ご清聴ありがとうございました