embarcadero®

Developer Camp

【T2】Delphiテクニカルセッション

こんなに簡単！Delphi によるiOS/Android 業務アプリケーション開発！

株式会社ミガロ.
吉原　泰介

ミガロ.について

株式会社ミガロ.

http://www.migaro.co.jp/

会社情報

所在地：本社　大阪市浪速区湊町2-1-57
難波サンケイビル13F

東京事業所　東京都港区麻布台1-4-3
エグゼクティブタワー麻布台11F

事業内容

IBM i 向けのソフトウェア・ツール販売および技術サポート

【開発ツール】

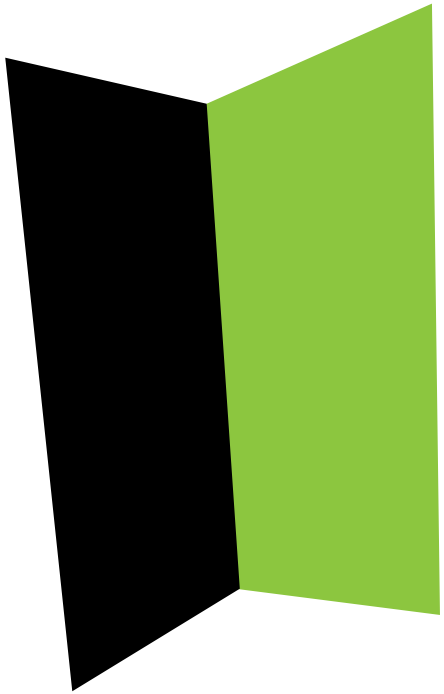
・Delphi/400

・JC/400

【スマートデバイス向けツール】

・Business4Mobile

- **Delphi/400**
- **DelphiをIBM i (AS/400)**に完全対応させたミドルウェア
- 国内約**700**社、全世界約**6,000**社の導入実績



アジェンダ

アジェンダ

1.企業導入が進むスマートデバイス

2.スマートデバイスアプリの種類

3.ネイティブアプリの開発
- 3-1. ネイティブアプリの開発環境
- 3-2. 簡単なネイティブアプリの開発
- 3-3. DBに接続するネイティブアプリの開発
- 3-4. ネイティブアプリの配布

4.まとめ

企業導入が進むスマートデバイス

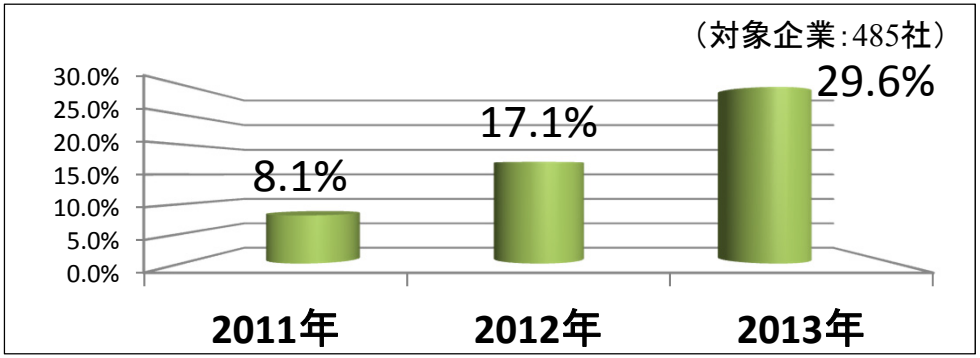
1.企業導入が進むスマートデバイス

- スマートデバイス企業導入率と導入OSの傾向

この1～2年でスマートデバイスの法人への導入が急速に進んできます。
また企業で導入されるスマートデバイスはiOS、Androidが主流になっている傾向です。

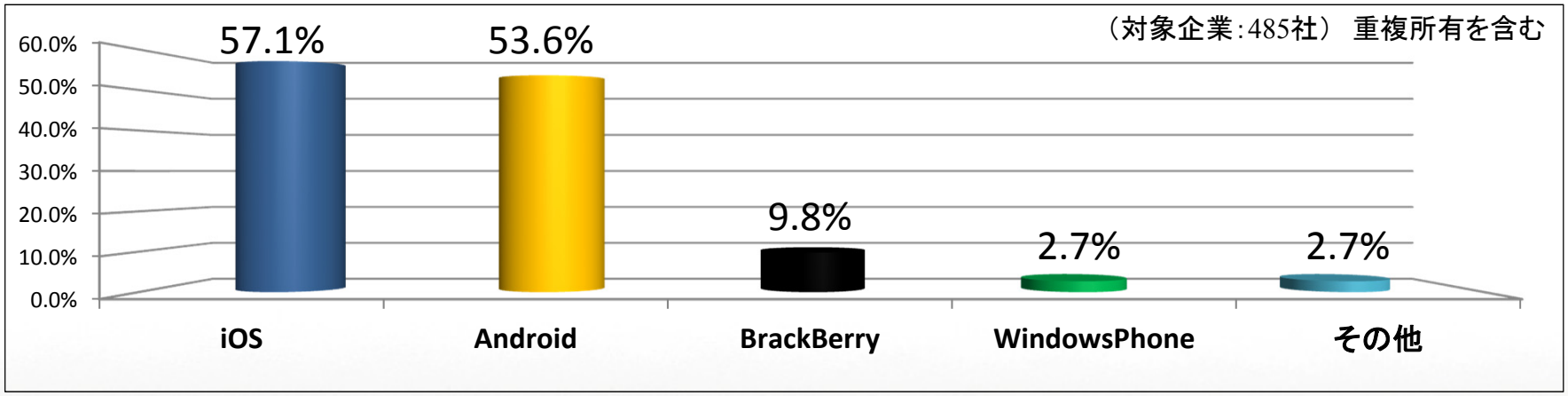
スマートデバイス企業導入率の遷移

2011年に比べると2013年では
導入率が3.5倍以上に増加



企業導入スマートデバイス比率（2013年）

スマートデバイスを導入している企業の大半が iOS 、Android を採用



1.企業導入が進むスマートデバイス

- スマートデバイスアプリの業務利用

スマートデバイス導入企業の多くは「業務効率化」を導入目的としています。

→ 今後はPCアプリケーション（Windows ）だけではなく、
スマートデバイスアプリも業務利用が増加

PCアプリケーションとスマートデバイスアプリケーションの特徴の違い

特徴が違う

PCアプリの特徴
細かいキーボード入力ができる
画面が大きく見やすい
ネットワークが安定している
携帯性が低い

スマートデバイスアプリの特徴
どこでもすぐに使用できる
タッチで感覚的に操作できる
カメラ、GPS、センサー等が利用できる
細かいキーボード入力は不向き

スマートデバイスアプリの種類

2.スマートデバイスアプリの種類

- スマートデバイスで利用されるアプリケーション
  スマートデバイス（iOS、Android）で利用することができるアプリは、
  大きく2種類のアプリケーションに分かれます。
  （ハイブリッドアプリケーションは折衷形式）

ネイティブアプリケーション

Webアプリケーション

2.スマートデバイスアプリの種類

ネイティブアプリケーション

ネイティブアプリケーションは、スマートデバイス端末上で動作して
デバイス機能と連携ができるアプリケーションです。
App Store や PlayStore といったストアや、社内向けに公開した
Webサーバ等からインストールして利用することができます。

一般的特徴

・デバイス機能（カメラ、GPS等）が利用できる
・レスポンスが良い
・オフラインでも利用できる

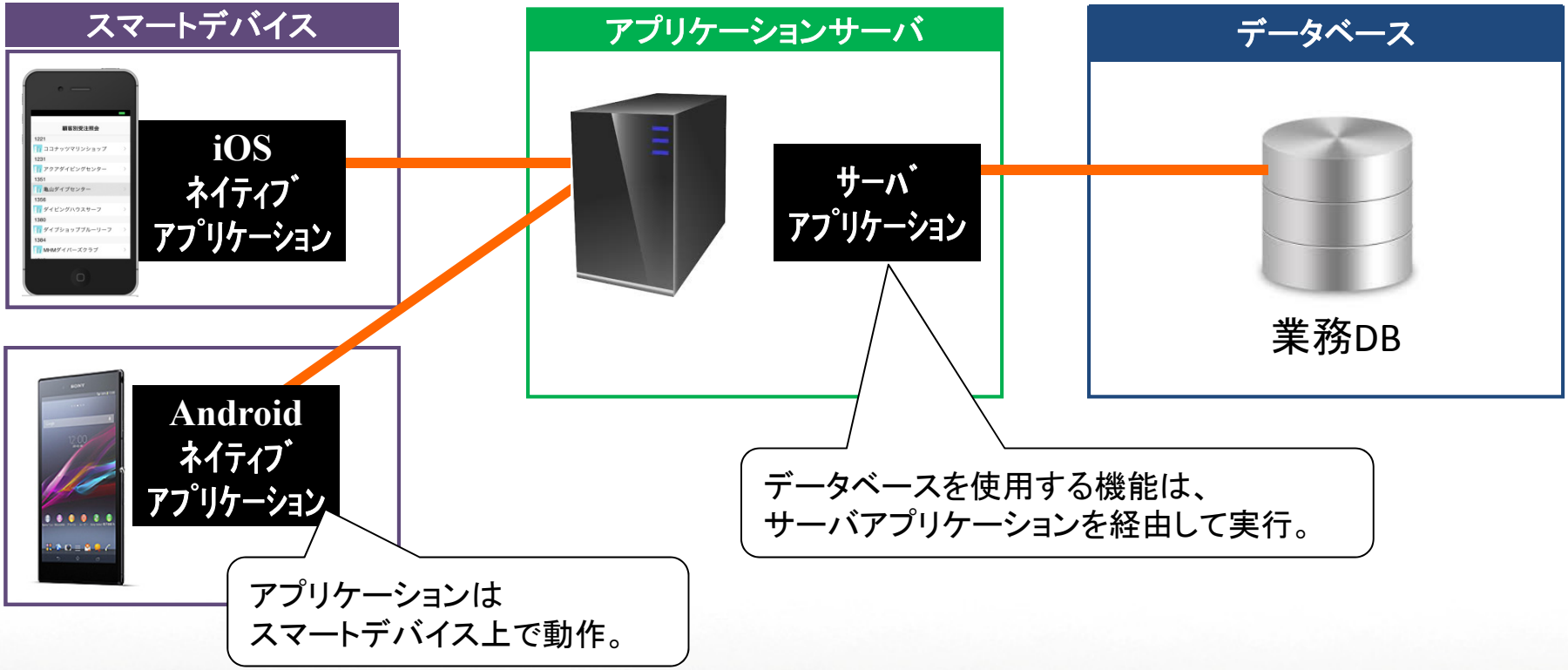
開発言語例

・iOS：　　Objective-C
・Android：　Java等

2.スマートデバイスアプリの種類

ネイティブアプリケーション

ネイティブアプリケーションの環境構成



2.スマートデバイスアプリの種類

Webアプリケーション

Webアプリケーションは、Webサーバ上で動作するプログラムを
PC同様にブラウザから利用できるアプリケーションです。
スマートデバイス端末にアプリケーションはインストールされないため、
ブラウザのブックマーク等を使って利用することができます。

一般的特徴

・ブラウザで実行するため、プラットフォームを問わず
　汎用的に開発・利用できる。
・インストールが不要なため、利用が容易。

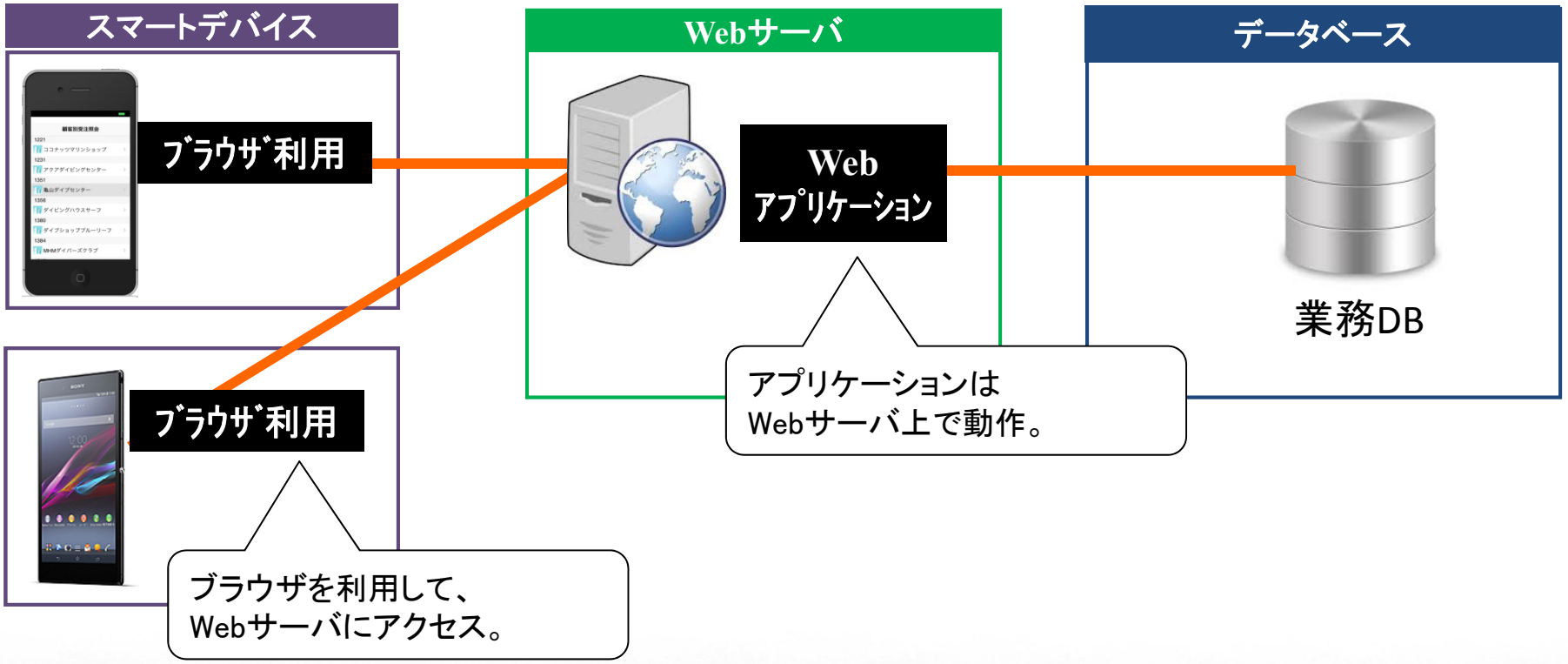
開発言語例

・HTML5、javaScript、 PHP、java等

2.スマートデバイスアプリの種類

Webアプリケーション

Webアプリケーションの環境構成



2.スマートデバイスアプリの種類

- ネイティブ / Webアプリの特徴

	ネイティブ	Web
開発言語	iOS:Objective-C Android:Java	HTML javascript等
開発生産性	△	○
デバイス機能	◎	△
パフォーマンス	◎	○
オフライン動作	◎	×
配布	△	◎

多くの開発言語習得が必要

言語によって開発環境も煩雑になるため、大変

2.スマートデバイスアプリの種類

- ネイティブ / Webアプリの特徴（Delphi）

	ネイティブ	Web
開発言語	Delphi	Delphi
開発生産性	◎	◎
デバイス機能	◎	△
パフォーマンス	◎	○
オフライン動作	◎	×
配布	△	◎

開発言語を
Delphiで統一できる

Delphiの開発機能は
生産性が高い

2.スマートデバイスアプリの種類

・Delphiネイティブアプリケーションの強み

DelphiスキルでiOS / Android ネイティブ開発ができる

開発言語はDelphiだけで iOS / Android のネイティブ開発ができます。
またコンパイルの設定切り替えだけで、1つのプログラムから
iOS / Android 両方のデバイスに対応できます。
XE7では、デバイス毎に細かい画面変更管理も可能です。(FireUI)

従来と同じ手法でネイティブ開発ができる

コンポーネントで画面設計して、イベントでプログラムコーディングする
従来の開発手法でネイティブアプリケーションが開発できます。

デバイス連携機能を簡単に開発することができる

スマートデバイス連携機能(カメラやGPS等)を専用コンポーネントで、
簡単に開発することができます。

2.スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携
  ネイティブアプリケーションではカメラ連携、バーコード連携、GPS連携
  といったデバイス連携機能をアプリケーションへ簡単に実装できます。

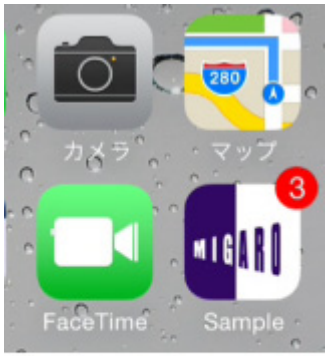
カメラ撮影

録音・再生

バーコード読取

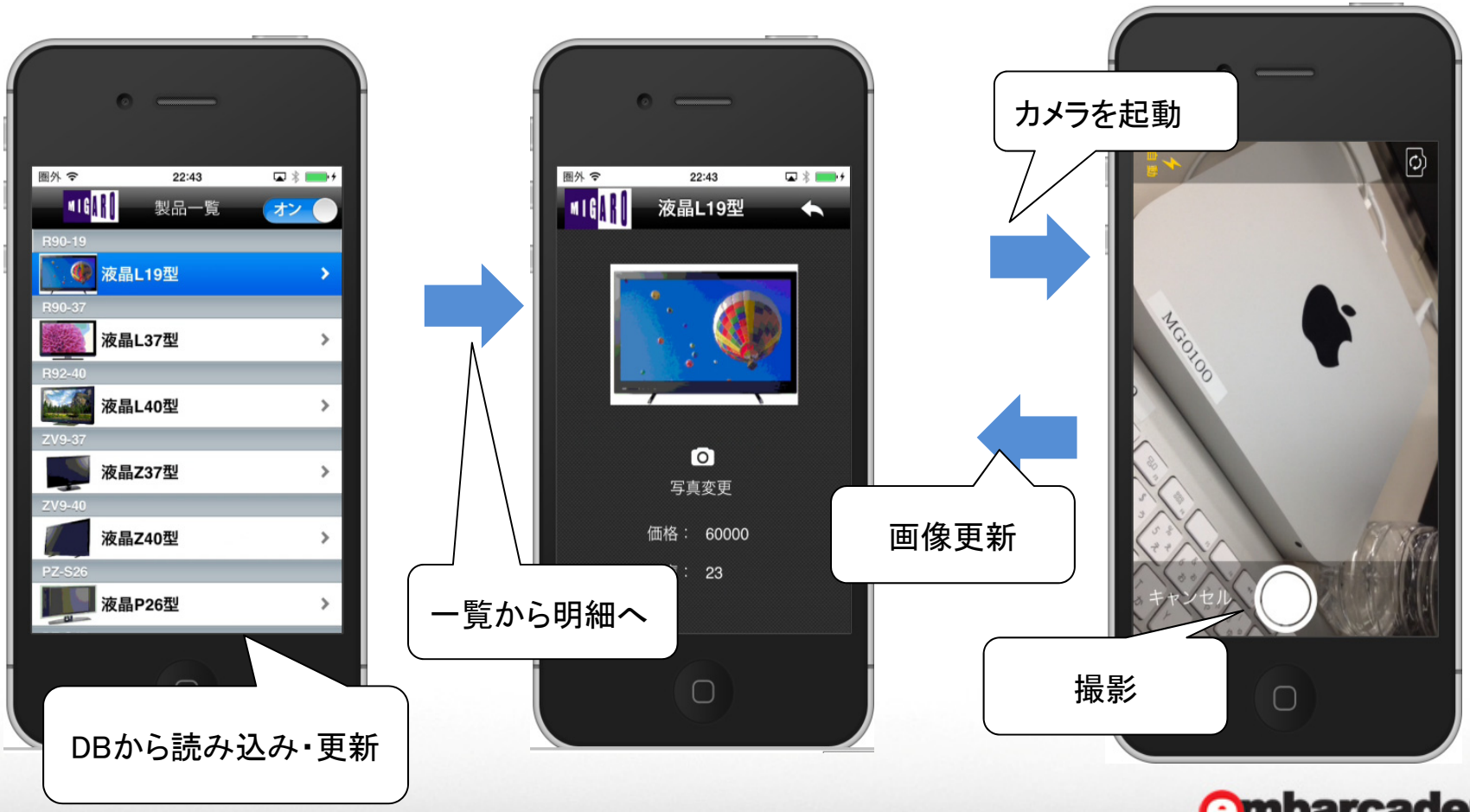
GPS位置情報取得

通知etc



2.スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携例1
  カメラ機能を連携したネイティブアプリケーション



2.スマートデバイスアプリの種類

- ネイティブアプリのデバイス機能連携例2
  バーコード、QRコード読取りを活用したネイティブアプリ

2.スマートデバイスアプリの種類

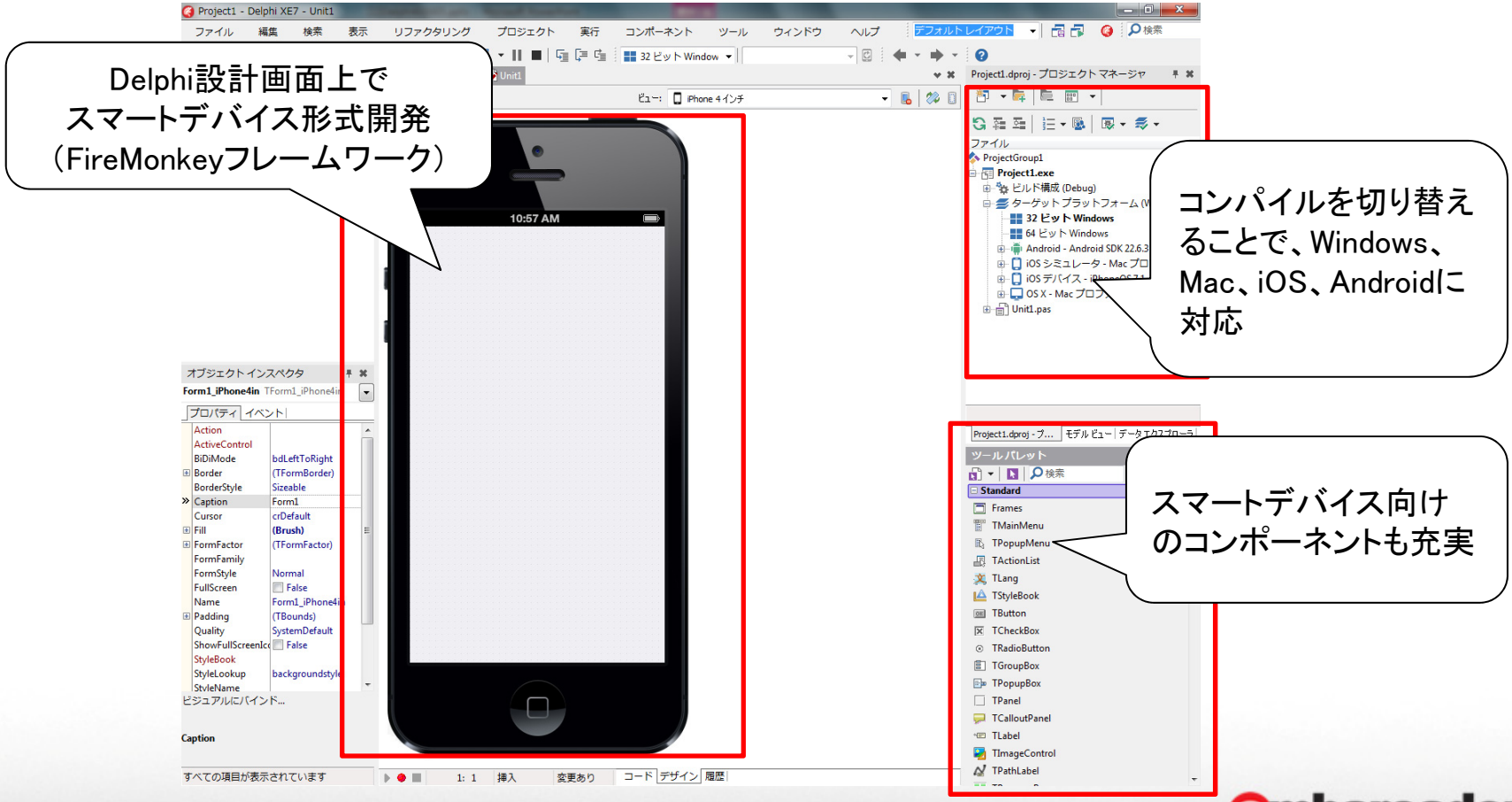
- ネイティブアプリのデバイス機能連携例3
  GPSを使って地図連携を活用したネイティブ

ネイティブアプリの開発

3-1. ネイティブアプリの開発環境

- Delphi のネイティブアプリケーション開発機能

  Delphi では、従来のWindowsアプリケーション向けVCL開発機能に加え、FireMonkeyフレームワークを利用することで、 Mac、iOS、Androidなどのマルチデバイス開発が可能です。

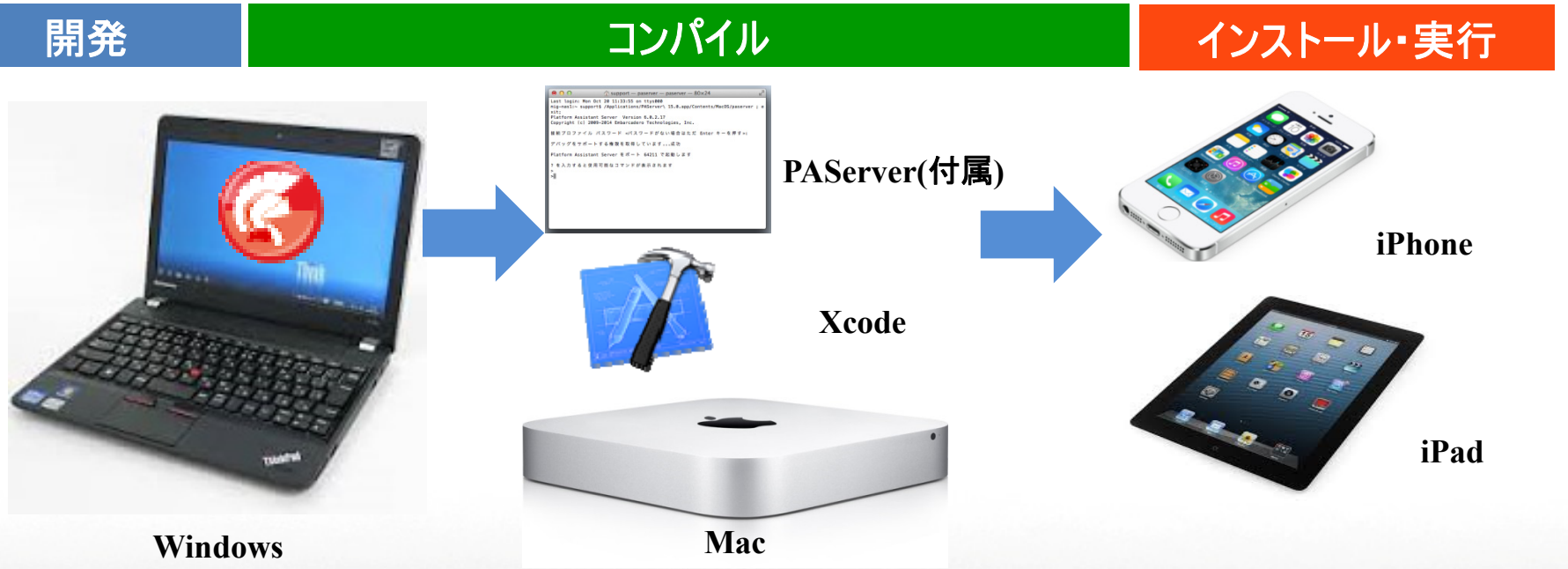


3-1. ネイティブアプリの開発環境

- iOSネイティブアプリ開発に必要な環境
  - Windows端末（Delphi）
  - Mac端末（OSX 10.8～10.9）
  - iOS Developer Program(Xcode,配布）
  - iOS実機(iPhone、iPad等 iOS7～7.1）

8はupdate1?

Mac上にWindows仮想環境を作成して、1台で構成することも可能です



3-1. ネイティブアプリの開発環境

- Androidネイティブアプリ開発に必要な環境
  - Windows端末（Delphi）
  - Android実機
    （Android 2.3.3以降のARM7 + NEON対応デバイス）

開発	コンパイル	インストール・実行

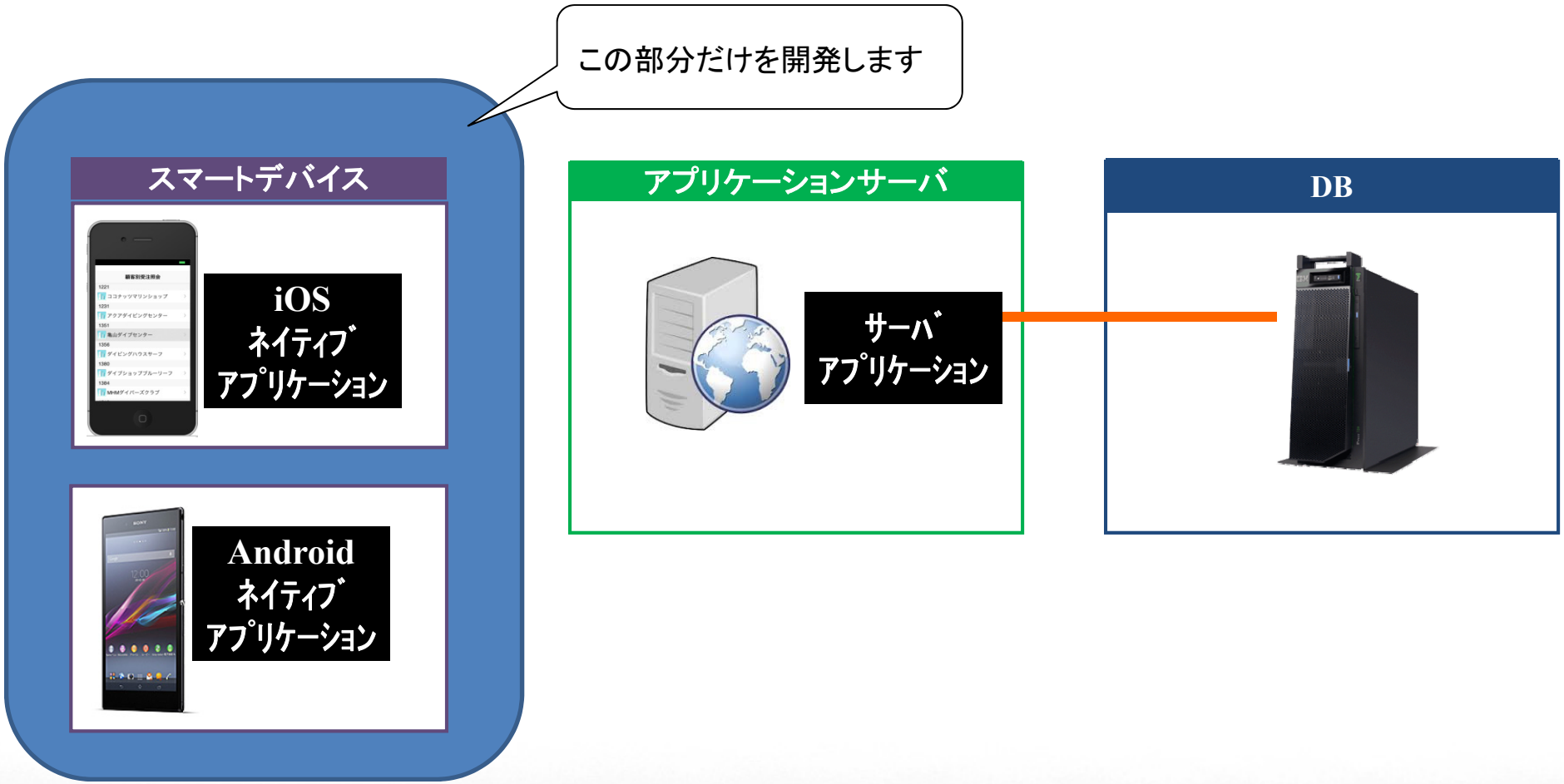
Windows

android SDK

Android
Phone/Tablet

3-2. 簡単なネイティブアプリの開発

- 簡単なネイティブアプリの開発



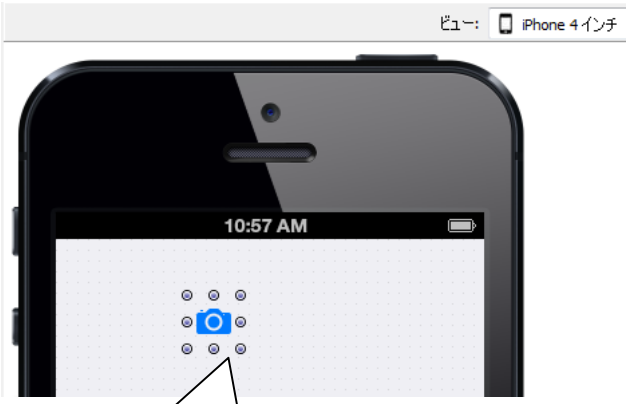
3-2. 簡単なネイティブアプリの開発

- デモで開発するネイティブアプリケーション
  スマートデバイス機能（カメラ）を連携したアプリケーションを開発

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリケーション開発の流れ
  ネイティブアプリケーションの開発は、従来のC/S、Web開発手法と同じ

① コンポーネントの配置 → ② イベントにプログラミング → ③ コンパイル/インストール

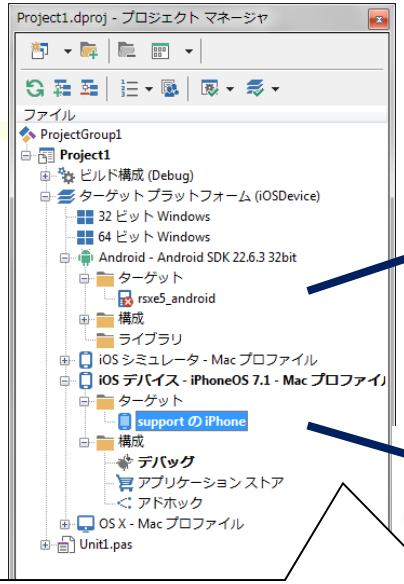


コンポーネントを使って
画面設計

```
implementation

{$R *.fmx}
{$R *.iPhone4in.fmx IOS}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Button1.Text := 'クリックしました';
end;

end.
```

Delphi言語で
プログラミング

コンパイル　　インストール

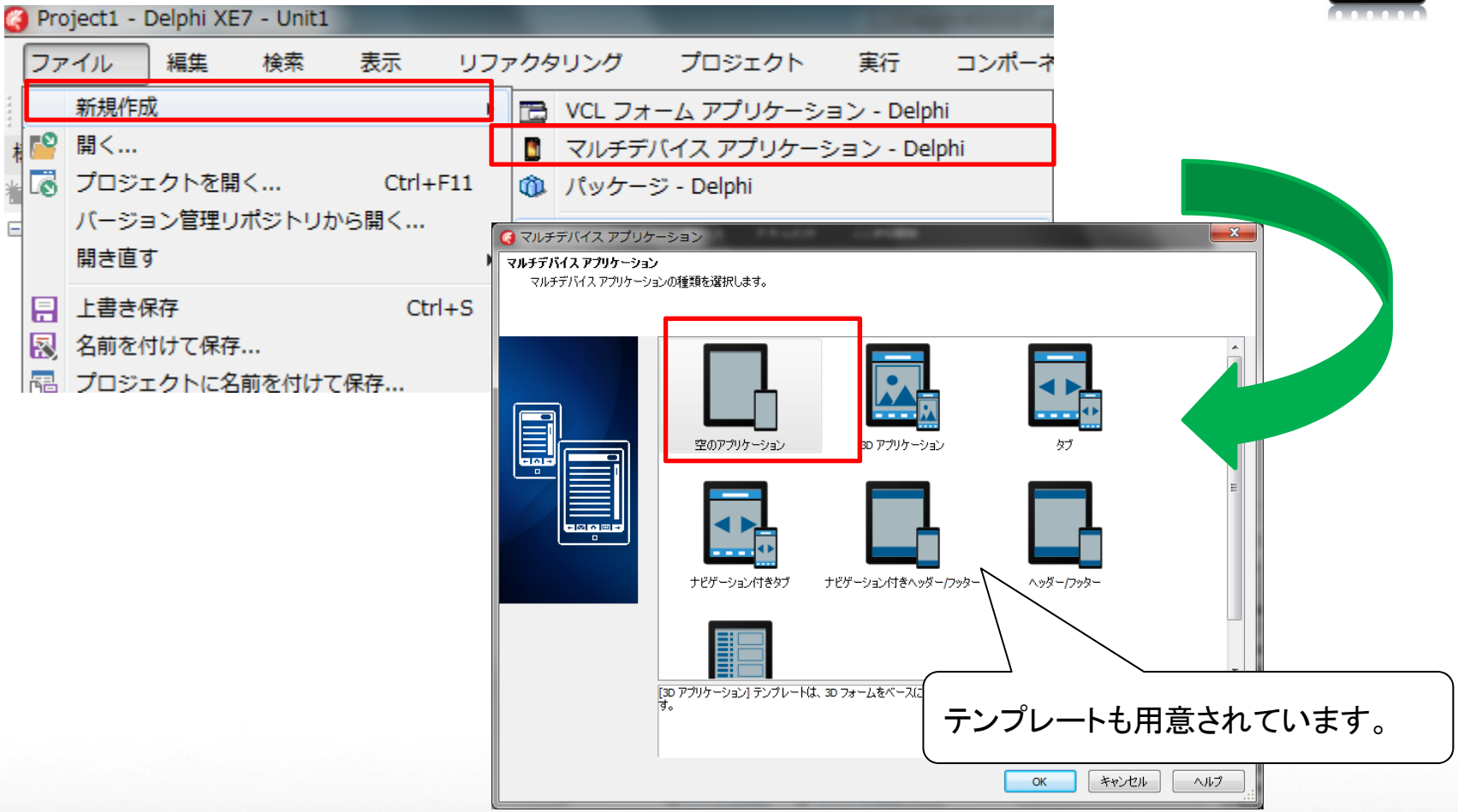


ターゲットのスマートデバイスに
コンパイルしてインストール

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順1

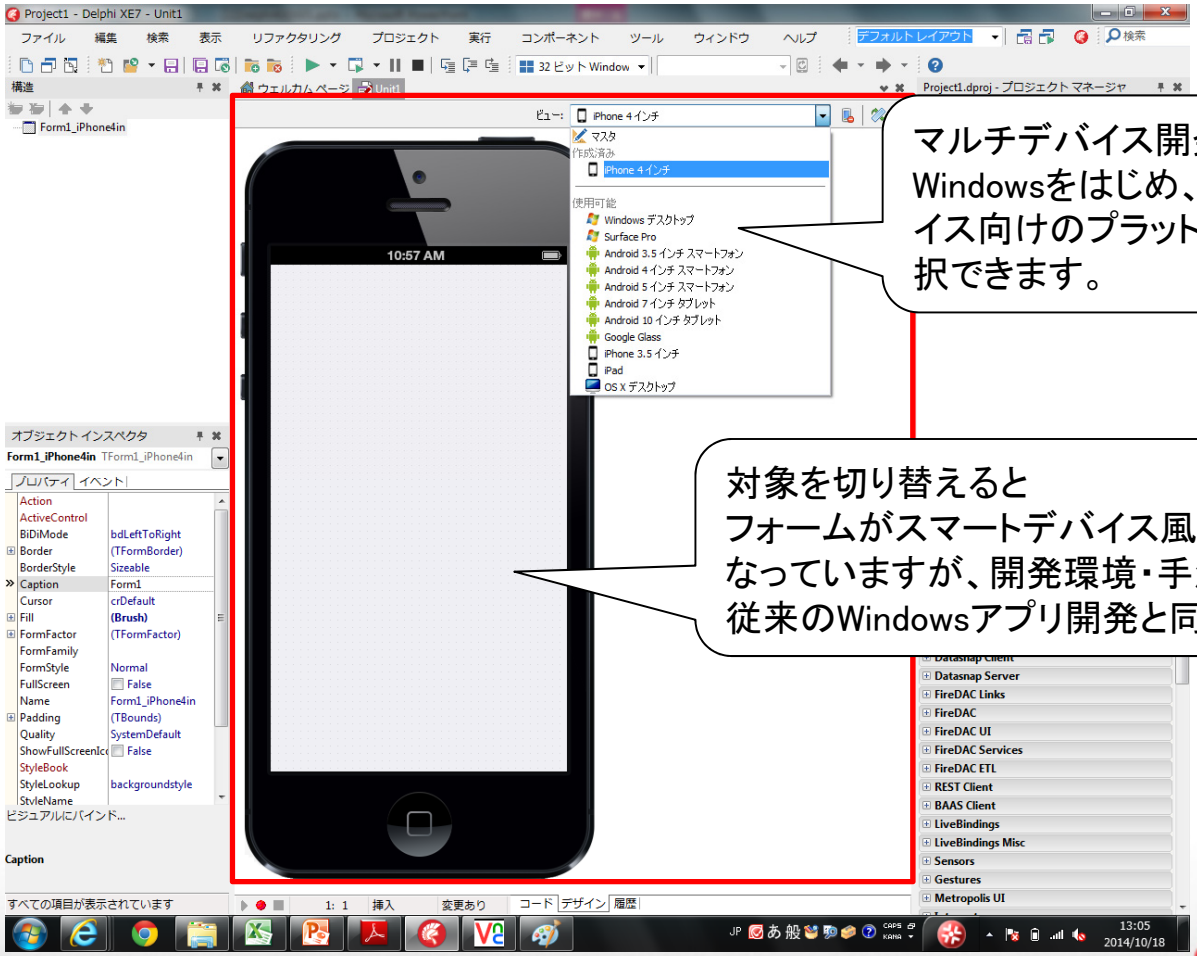
新規作成よりマルチデバイスアプリケーションを選択



3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順2

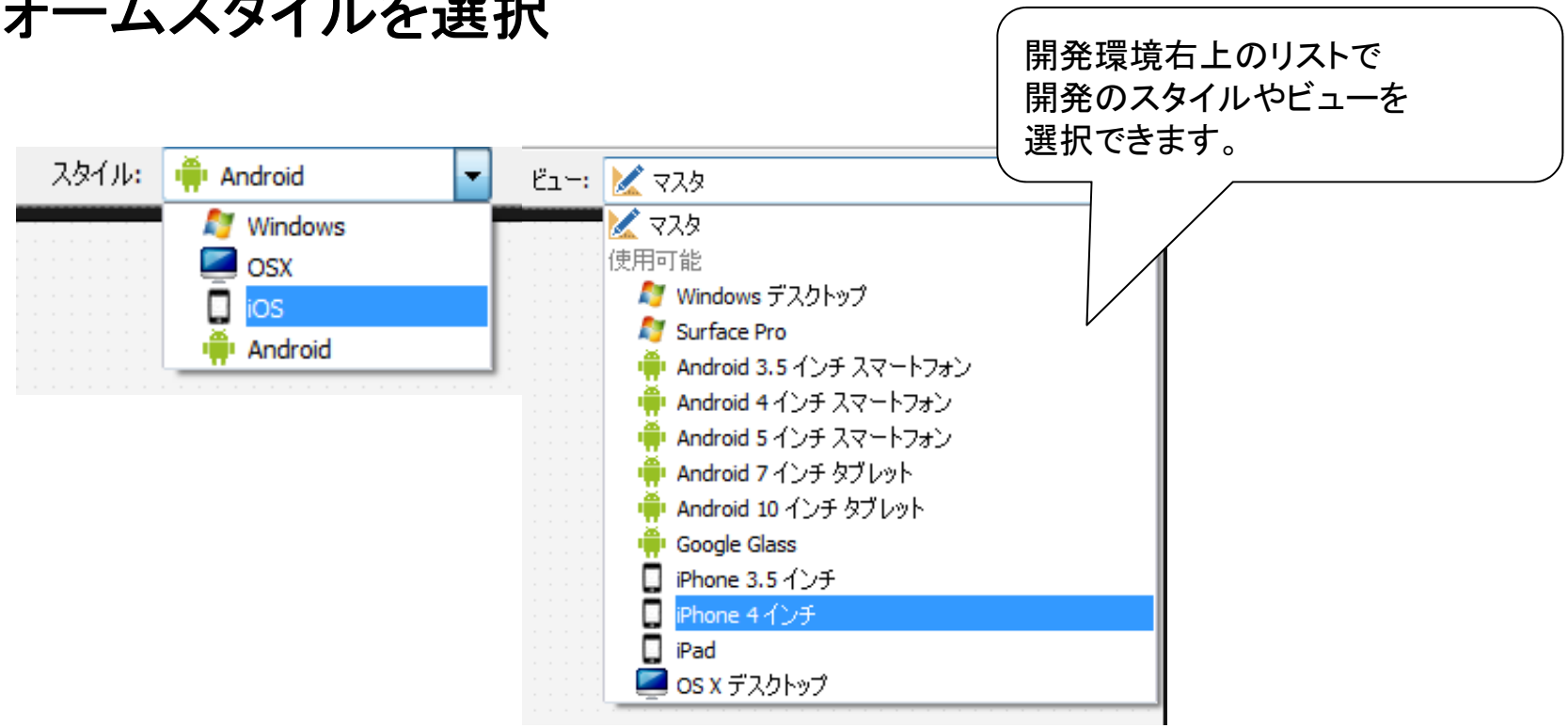
ネイティブアプリ開発画面



3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順3

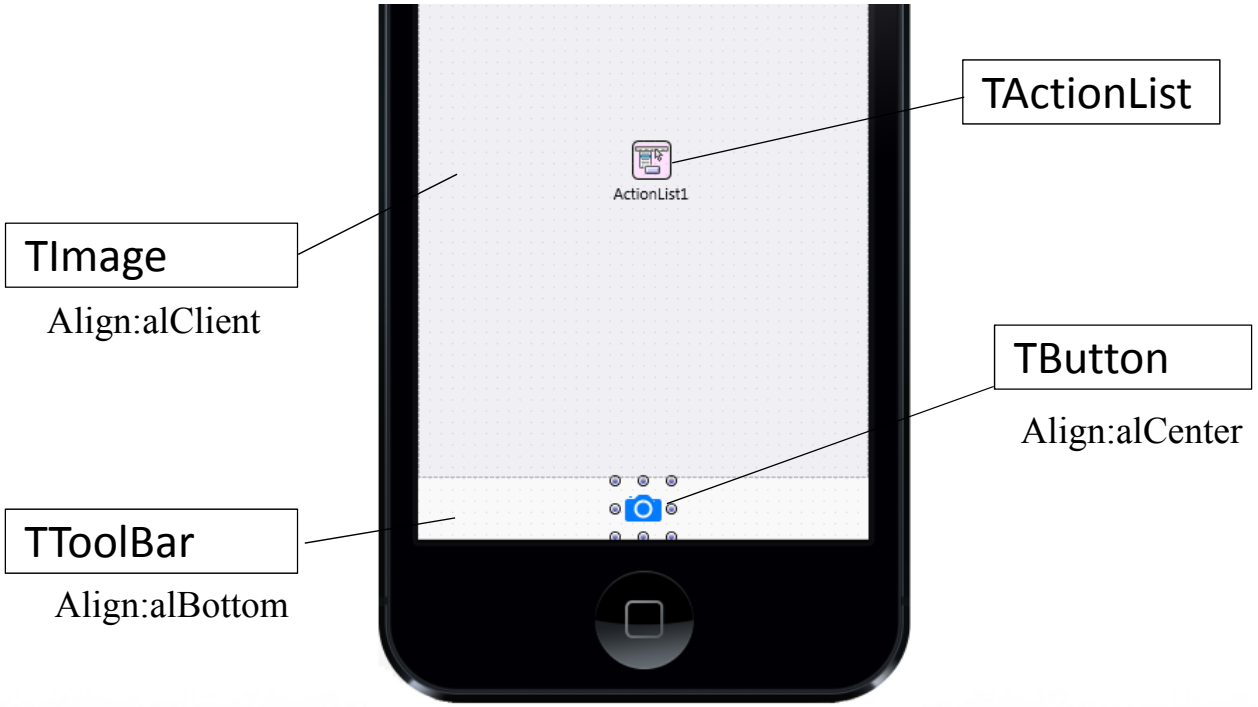
フォームスタイルを選択

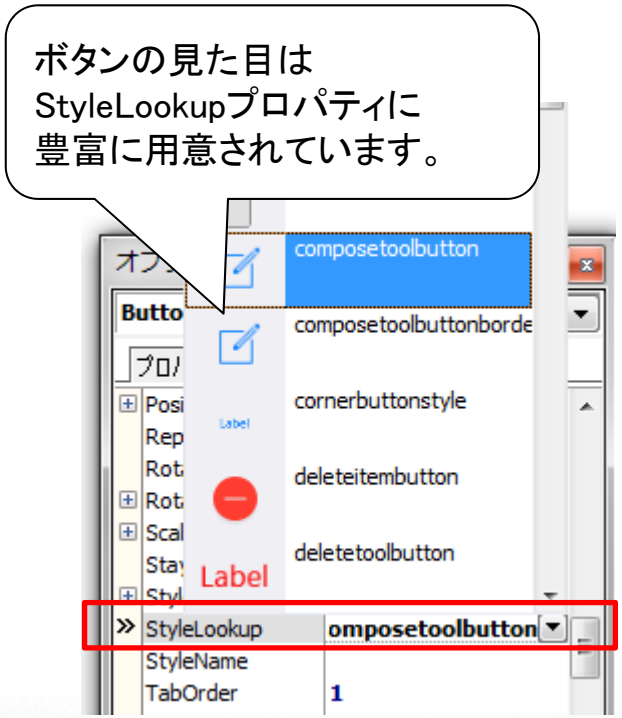


3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順4

フォームに次のコンポーネントを配置
TToolBar、TButton、TImage、TActionList

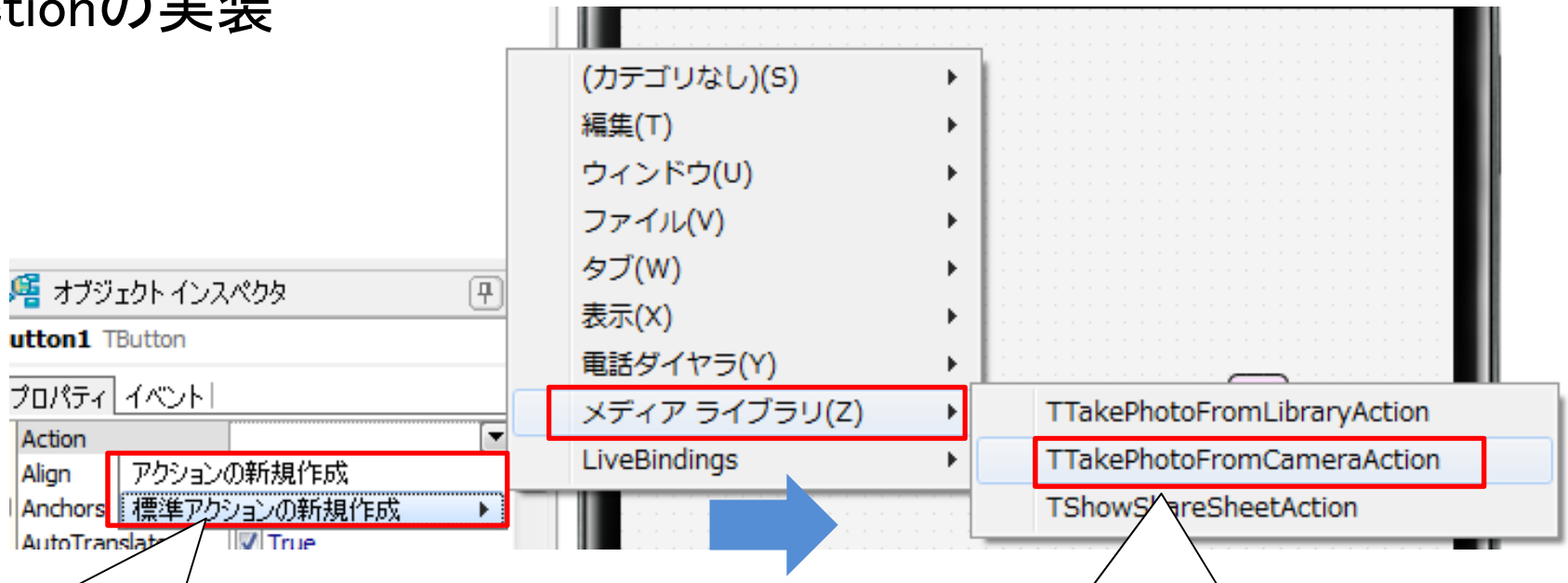




3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順5

Actionの実装



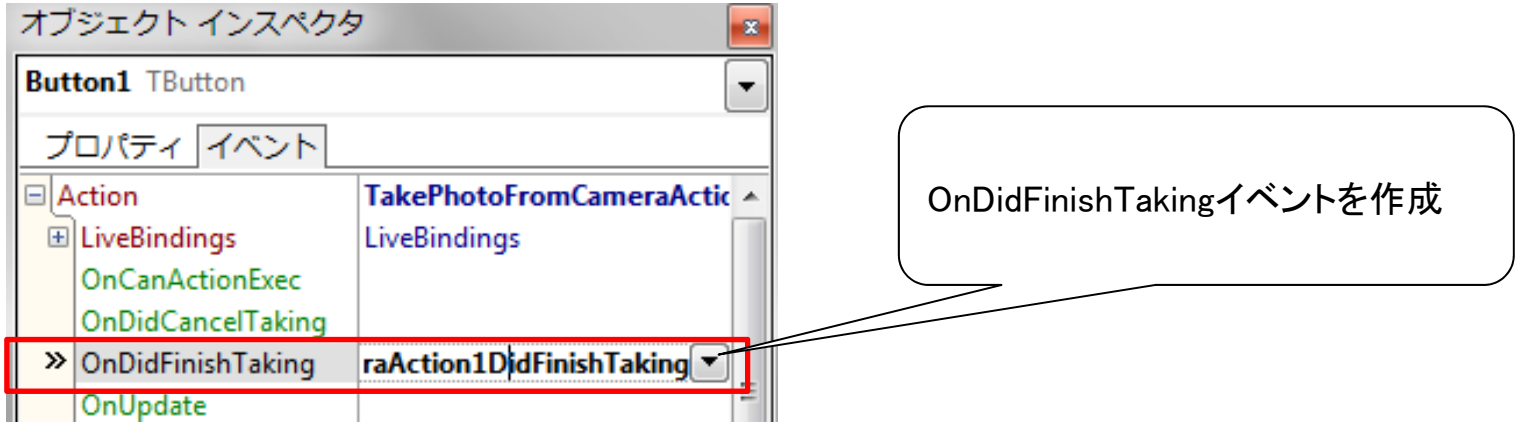
ButtonのActionプロパティで
標準アクションの新規追加

TTakePhotoFromCameraAction
を選択

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順5

Actionのイベントにプログラムを実装



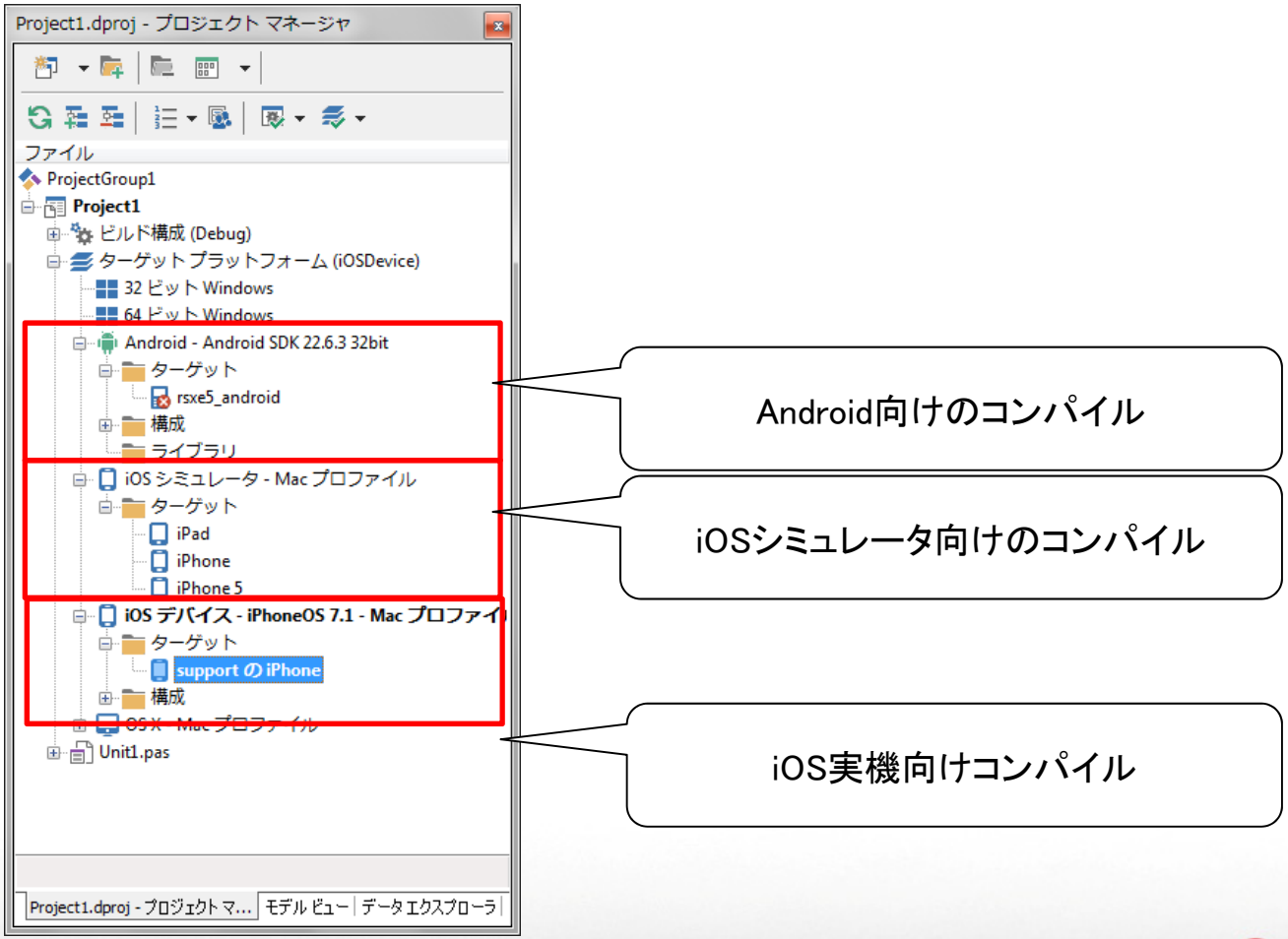
OnDidFinishTaking処理（カメラ撮影終了処理）

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
  Image1.Bitmap.Assign(Image);
end;
```

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順6

iOS実機向けにコンパイル



3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順7

iOSアプリケーションの実行

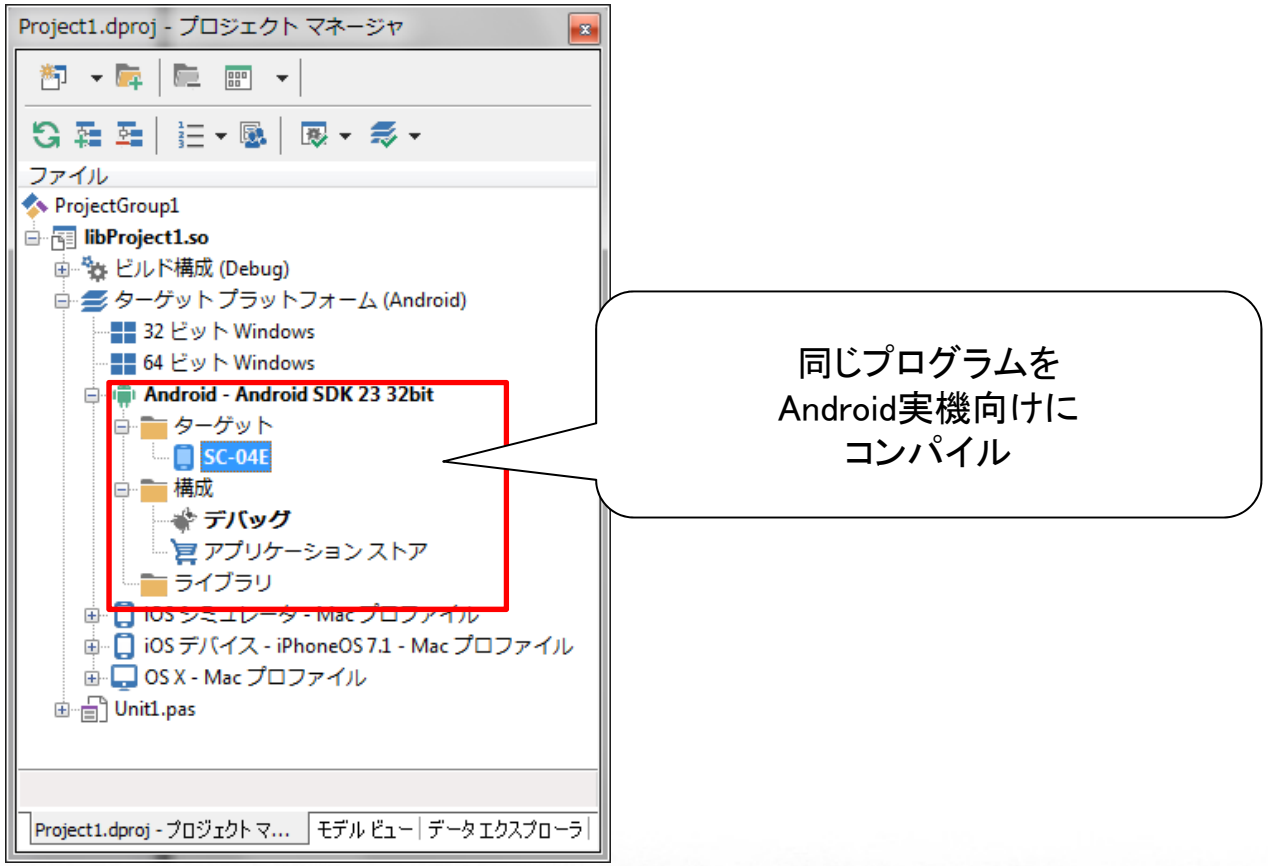
スマートデバイスの機能を使えば、カメラ撮影を連携したアプリケーションも
PCやWebのアプリケーションと比べて、簡単に実現ができます。

35

3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順8

Android実機向けにコンパイル



3-2. 簡単なネイティブアプリの開発

- ネイティブアプリ開発手順9

Androidアプリケーションの実行

1つのプログラムからiOS、Androidのネイティブアプリケーションを開発できます。

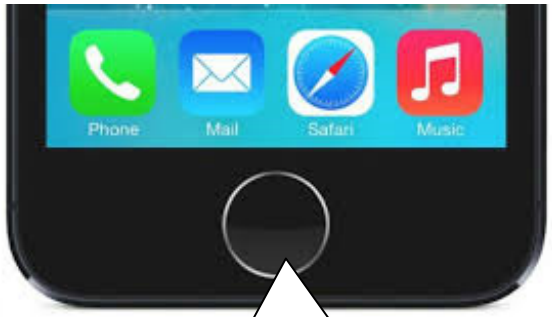
3-2. 簡単なネイティブアプリの開発

- 補足：iOSとAndroidの違い1

ハードウェアキーの違い

Androidには「戻るボタン」や「メニューボタン」が物理的に存在しますが、
iOSには「ホームボタン」しかありません。
例えばiOSで「戻るボタン」が前提のアプリを作成してしまうと意図した画面遷移操作が行えなくなります。そのため、OS・ハードの違いを把握した画面設計は非常に重要となってきます。

iOS



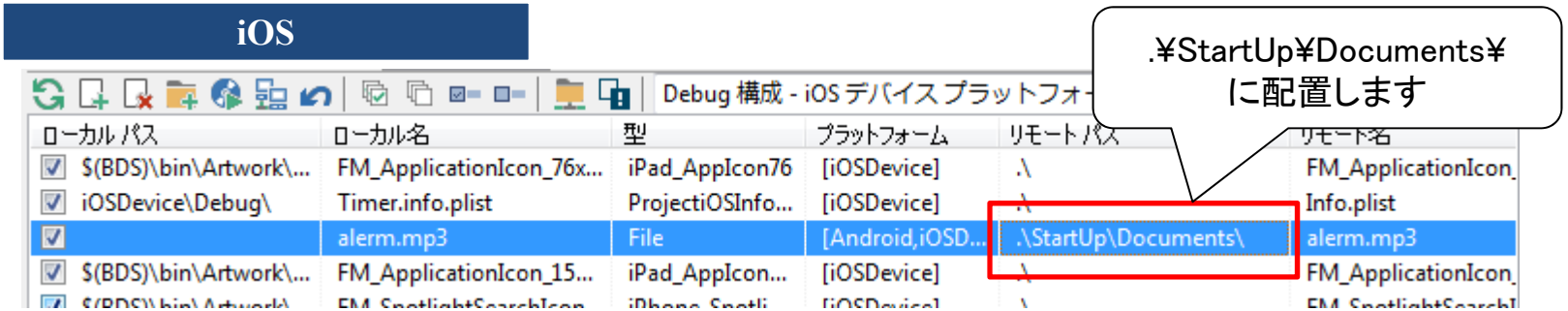
ホームボタン
しかない

Android

ホームボタン、メニューボタン、
戻るボタンがある

3-2. 簡単なネイティブアプリの開発

- 補足：iOSとAndroidの違い2
  ファイル配置の違い（プロジェクト|配置から設定）
  音源ファイルや動画ファイルなど、アプリケーション内で固有で持ちたい場合、配置（保存）先のパスはプラットフォームによって異なります。

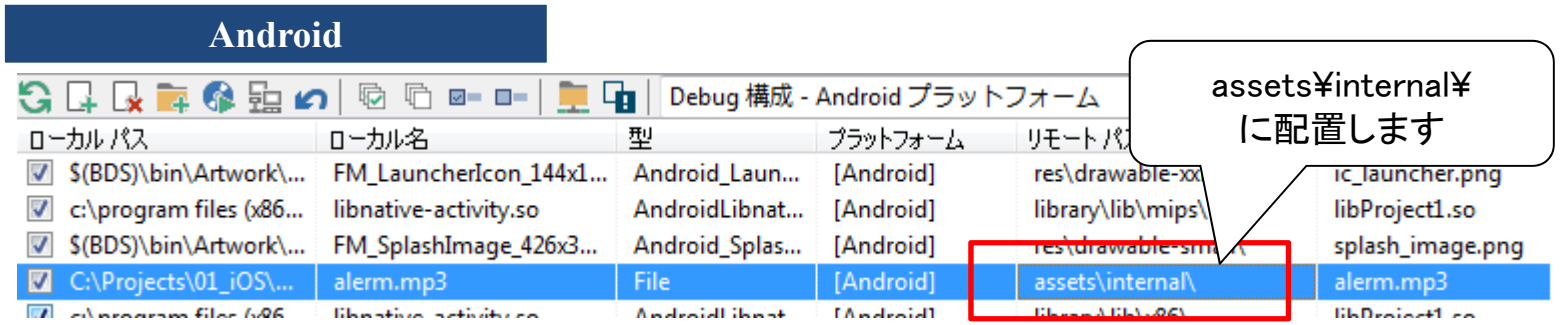


プログラム上でのネイティブファイルパス指定例（iOS）

```
GetHomePath +PathDelim + ‘ Documents ’ + PathDelim + ‘ ファイル名'
```

3-2. 簡単なネイティブアプリの開発

- 補足：iOSとAndroidの違い2
  ファイル配置の違い（プロジェクト|配置から設定）



プログラム上でのネイティブファイルパス指定例（Android）

```
TPath.GetDocumentsPath +‘ ファイル名 ’

補足）SDカードなど外部ストレージで扱う場合
TPath.GetSharedDocumentsPath  +‘ ファイル名'
```

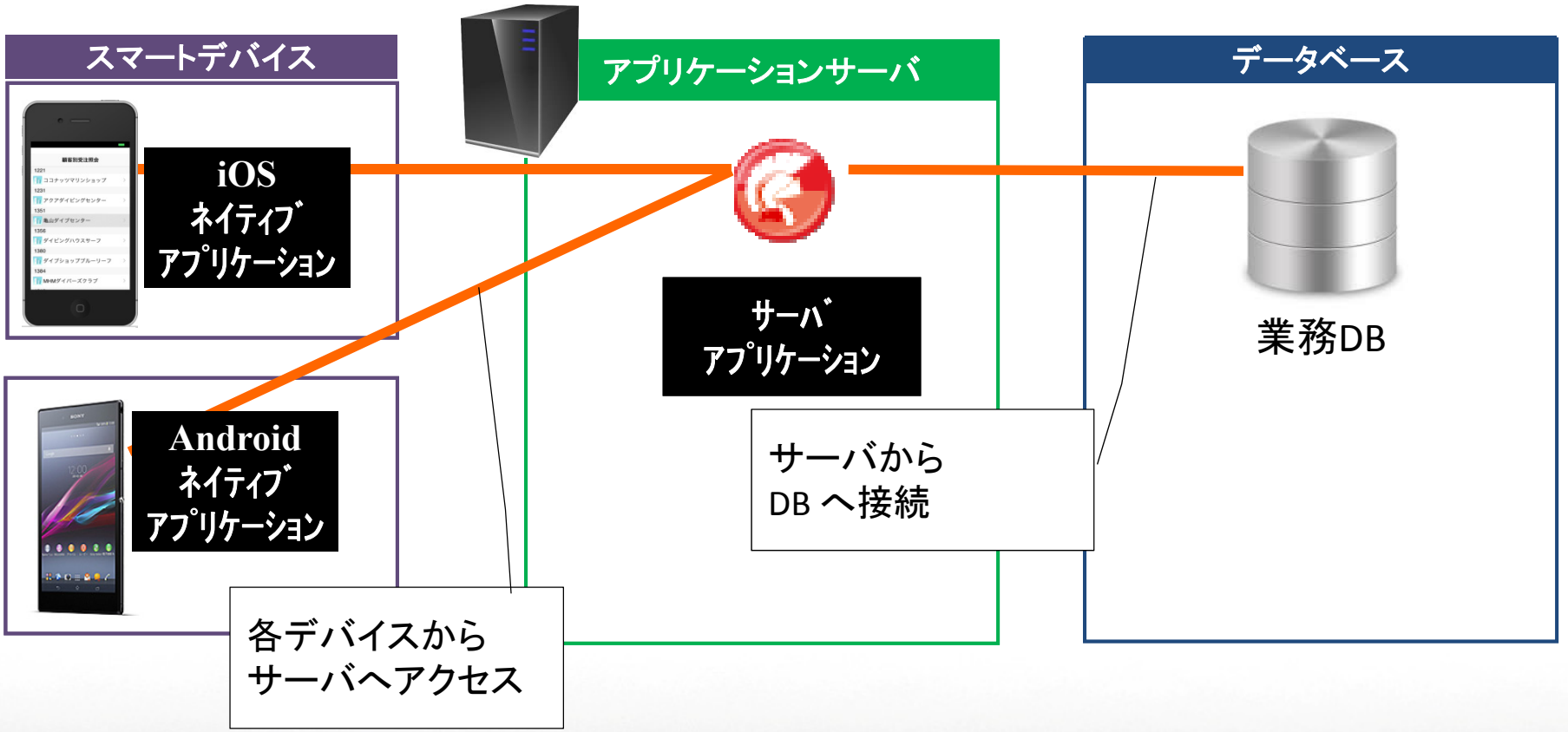
外部ストレージに保存しているファイルは
PCとのUSB転送などで便利です

40

3-3. DBに接続するネイティブアプリの開発

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリの仕組み
  ネイティブアプリケーションからデータベースに接続する仕組みは、
  Webアプリケーションに近い、サーバを経由した3階層方式になります。



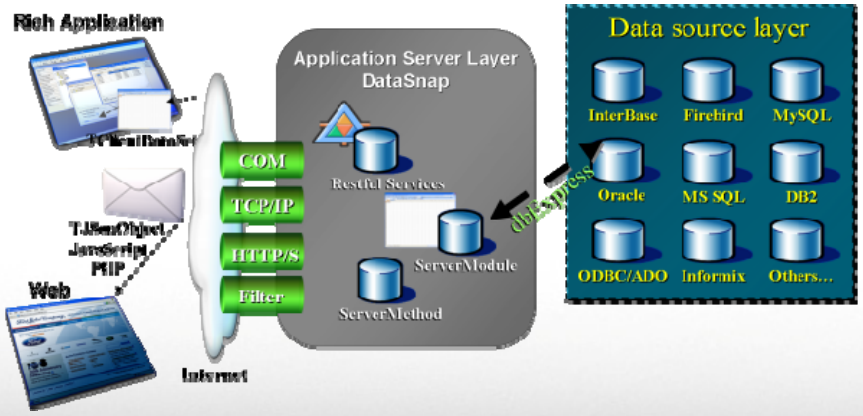
3-3. DBに接続するネイティブアプリの開発

- サーバアプリケーションとは？

  アプリケーションサーバ（中間サーバ）からデータベースに接続、処理を行うアプリケーションです。
  ネイティブアプリケーションは、サーバアプリケーションを経由して
  データベースにアクセスすることができます。

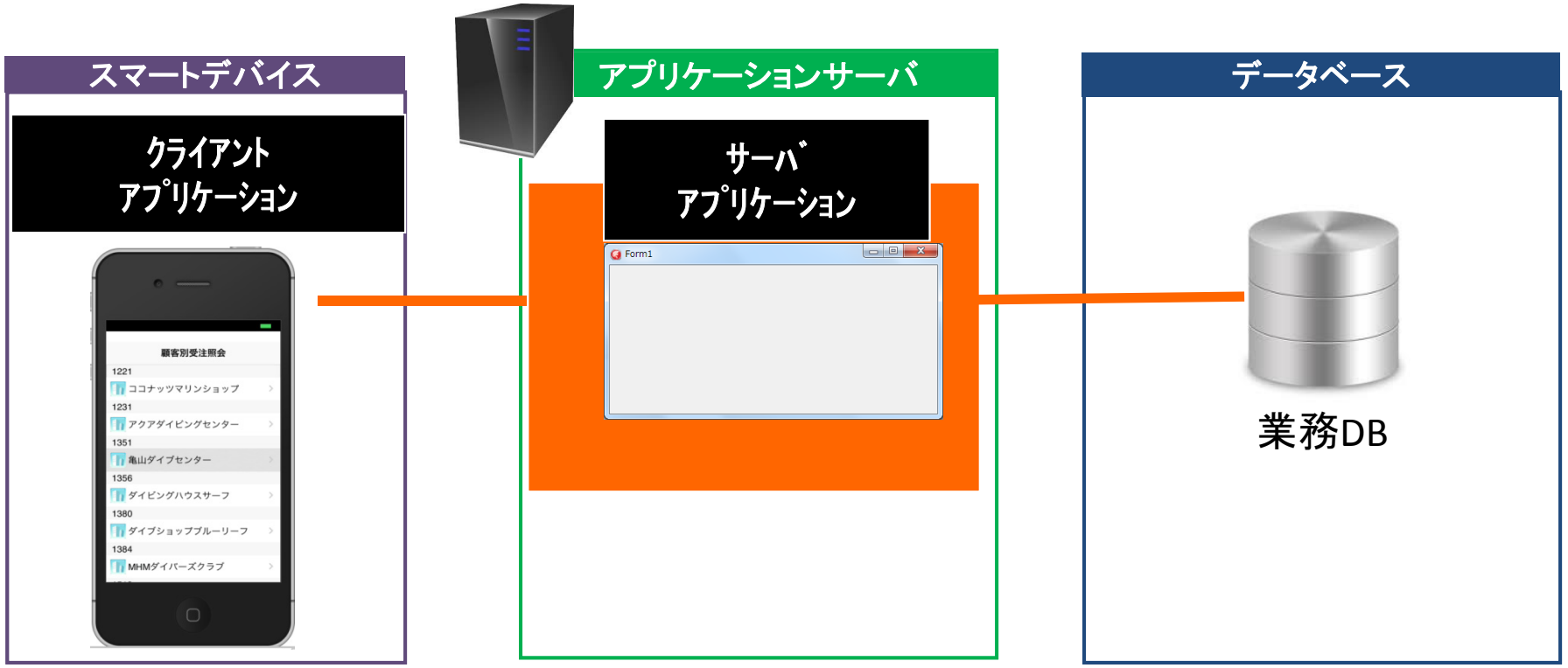
Delphiで開発するサーバアプリケーション

DelphiではサーバアプリケーションをDataSnapで簡単に開発できます。
『DataSnap』はサーバアプリケーション専用の開発機能です。
サーバアプリケーションは、FireDac、dbExpress等のDBコンポーネント
を設定したり、関数をプログラミングすることで機能を実装できます。



3-3. DBに接続するネイティブアプリの開発

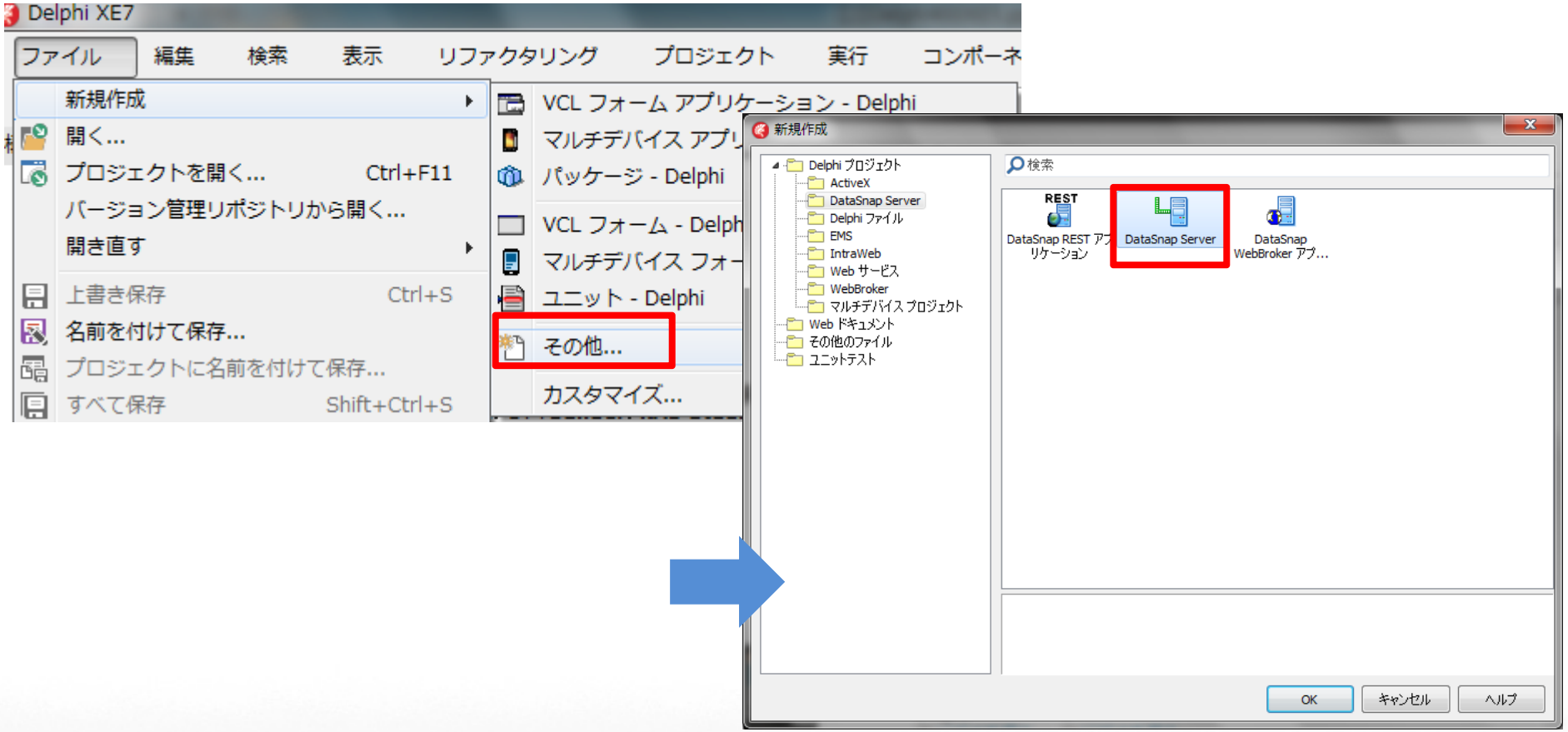
- サーバアプリの開発



3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順1

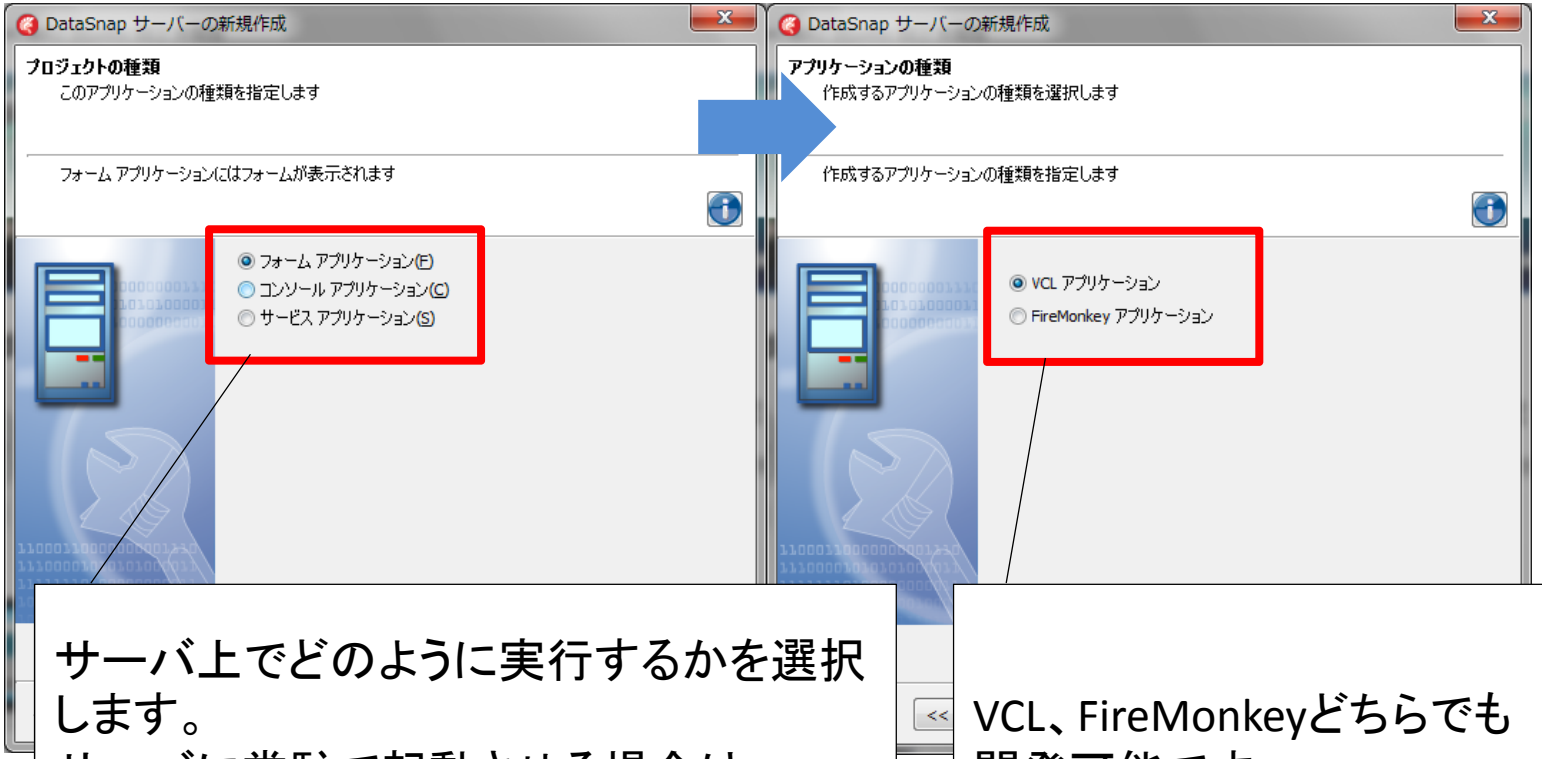
メニューの[ファイル|新規作成|その他]から『DataSnapServer』を選択



3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順2

DataSnapServerの形式を選択



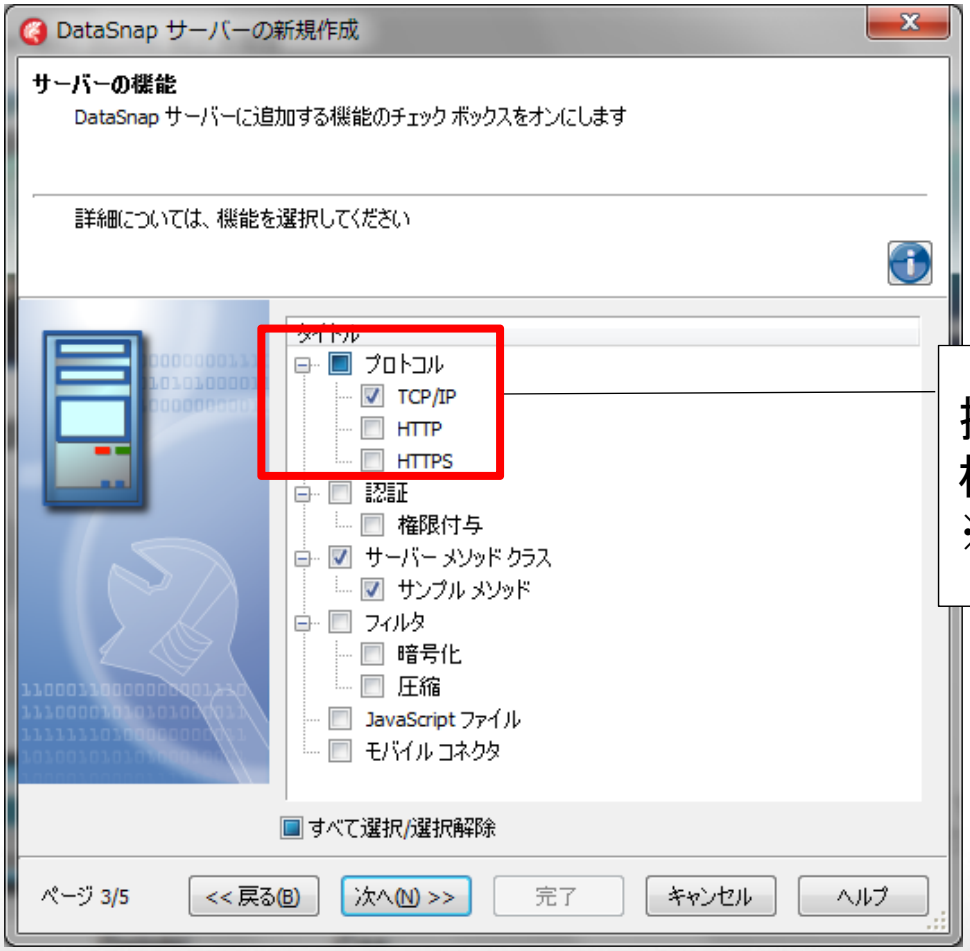
サーバ上でどのように実行するかを選択します。
サーバに常駐で起動させる場合は
サービスアプリケーションを使います。

VCL、FireMonkeyどちらでも
開発可能です。

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順3

通信プロトコルを設定



接続通信の設定
標準はTCP/IPを使います。
※HTTP経由での通信も可能です。

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順4

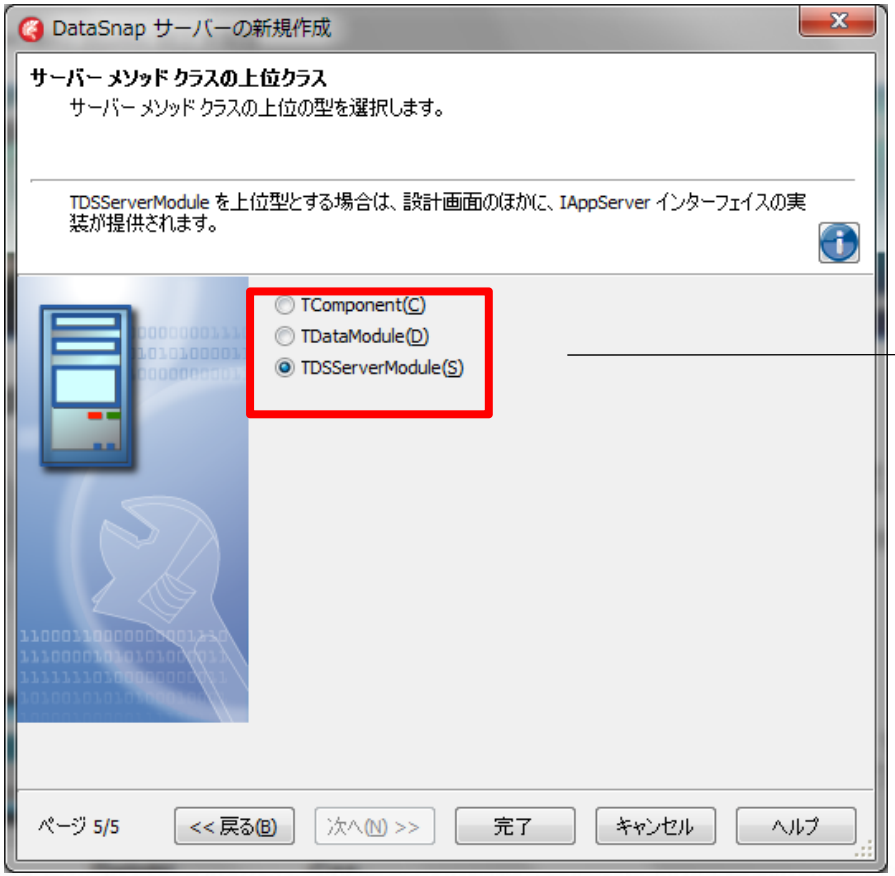
使用するポート番号を指定

サーバ接続に使用するポート番号を指定します。（デフォルト211）

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順5

サーバーメソッドの上位クラスを選択

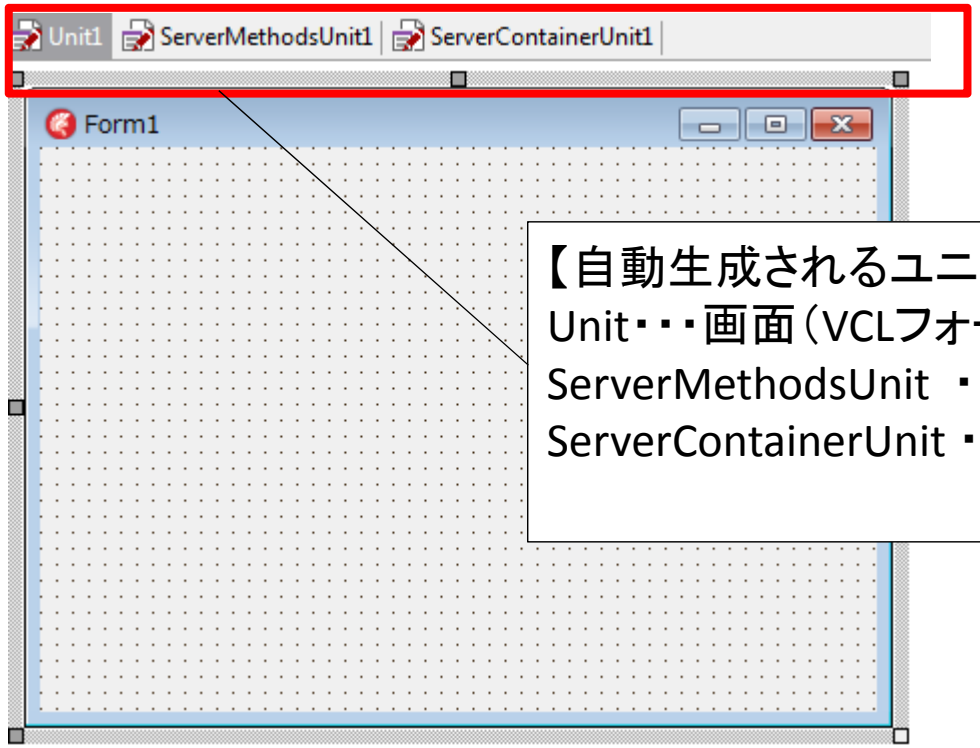


データセットを公開する場合は
TDSServerModuleを選択します。

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順7

選択した構成によってモジュールが生成

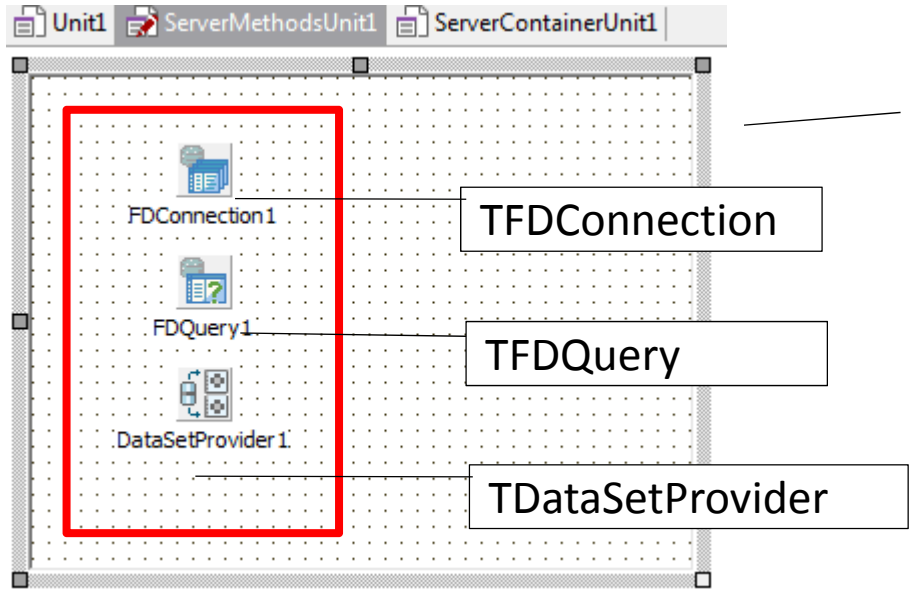


【自動生成されるユニット】
Unit・・・画面（VCLフォームアプリケーション時のみ）
ServerMethodsUnit ・・・サーバが提供する機能
ServerContainerUnit ・・・通信を制御する機能

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順8

ServerMethodsに次のコンポーネントを配置
TFDConnection、TFDQuery、TDataSetProvider



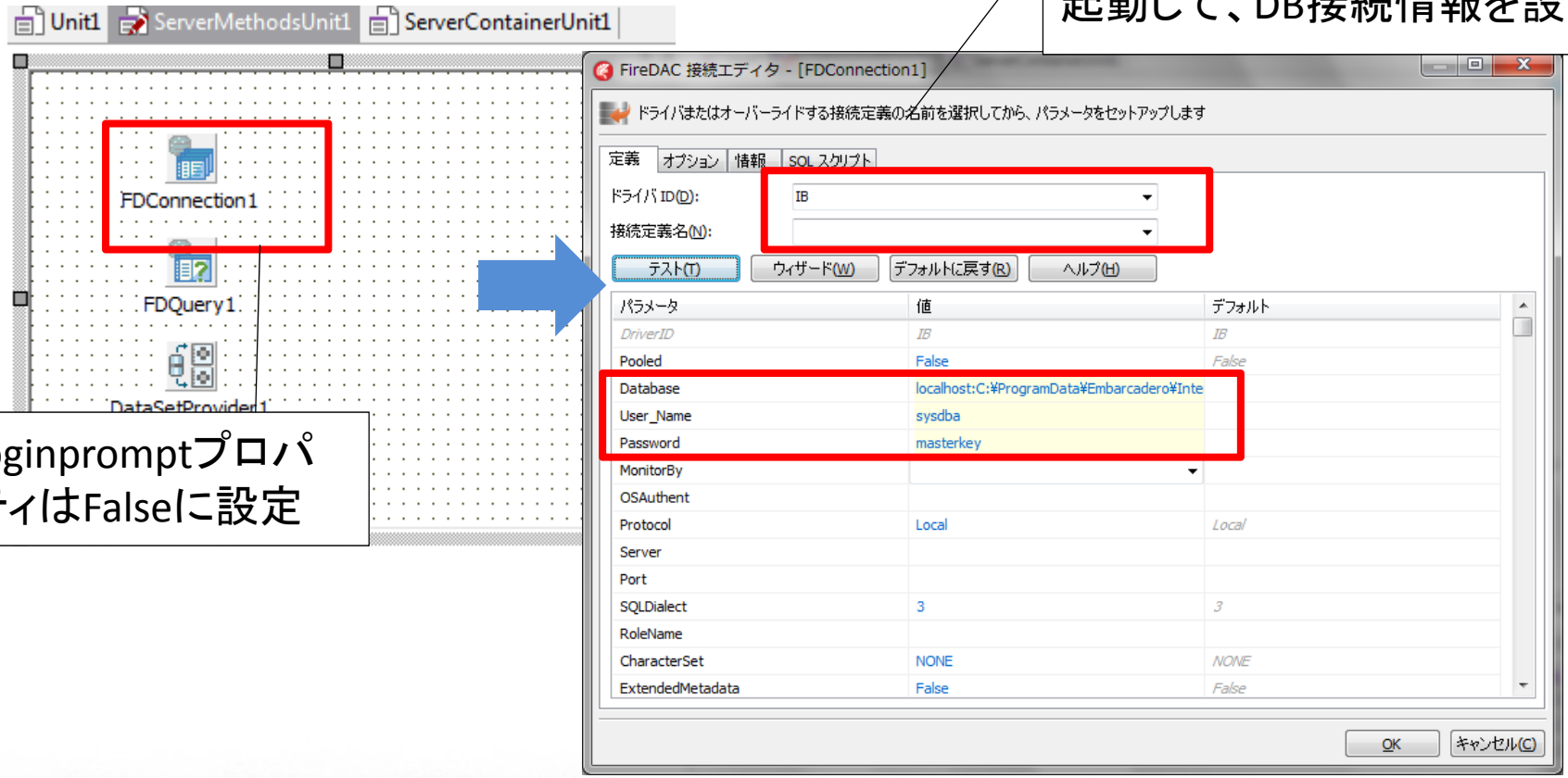
今回はInterBaseへSQL実行する
サーバアプリケーションの機能だけ
実装しています。
(dbExpressでも可能です)

3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順9

TFDConnectionコンポーネントの設定

右クリックから接続エディタを起動して、DB接続情報を設定

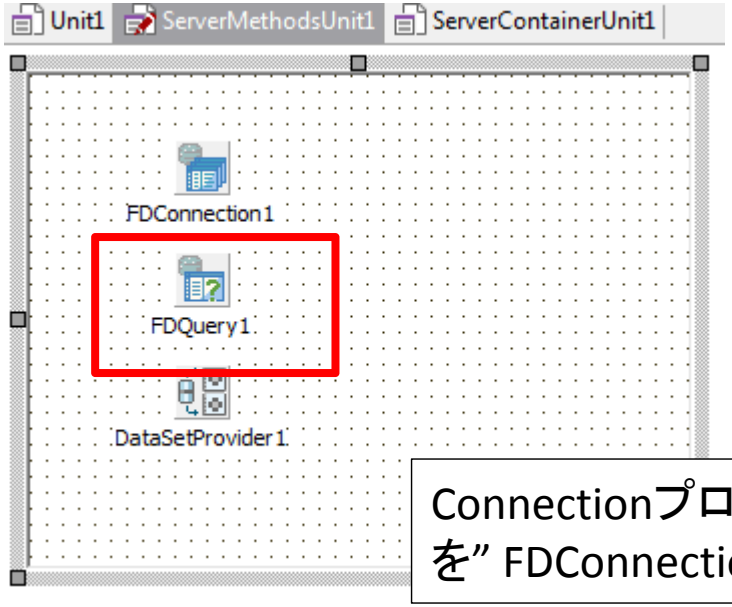


loginpromptプロパティはFalseに設定

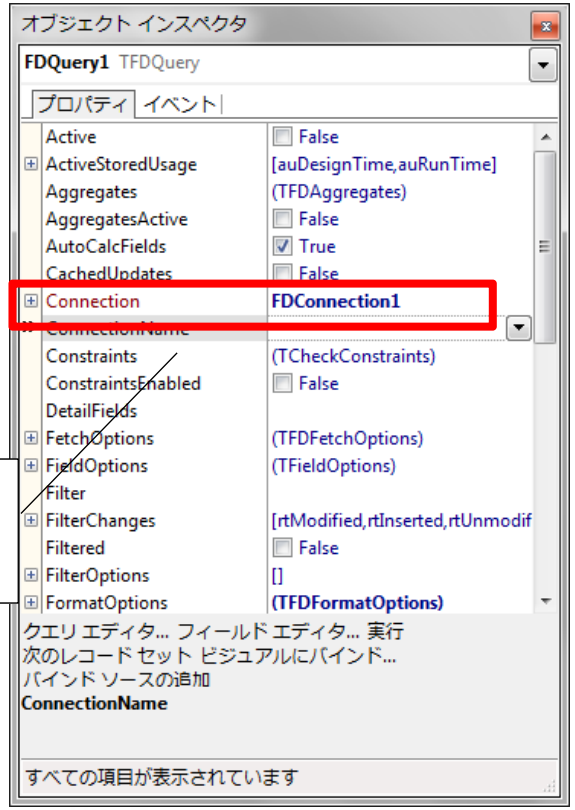
3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順10

TFDQueryコンポーネントの設定



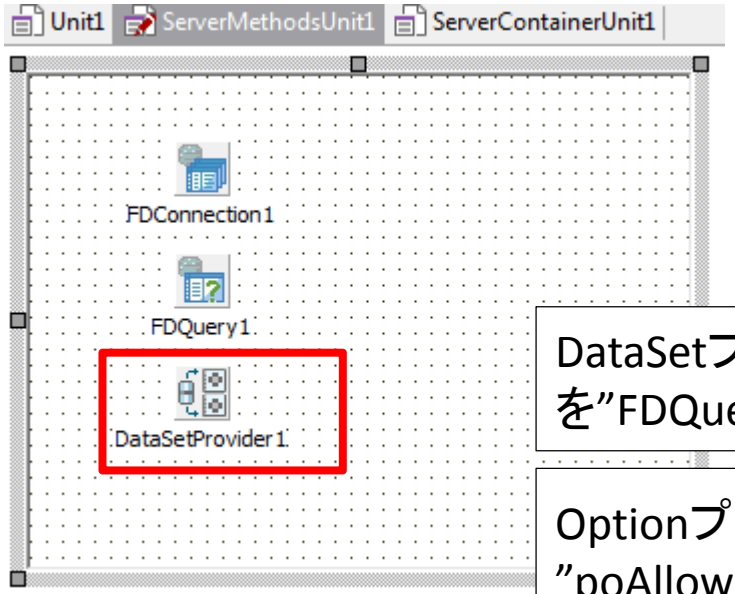
Connectionプロパティ
を” FDConnection1”に設定



3-3. DBに接続するネイティブアプリの開発

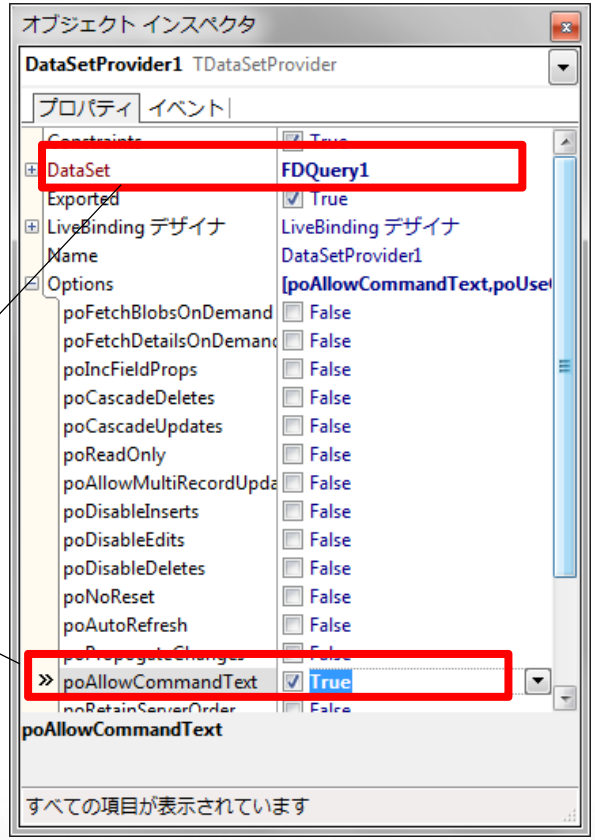
- サーバアプリの開発手順11

TDataSetProviderコンポーネントの設定



DataSetプロパティ
を”FDQuery1”に設定

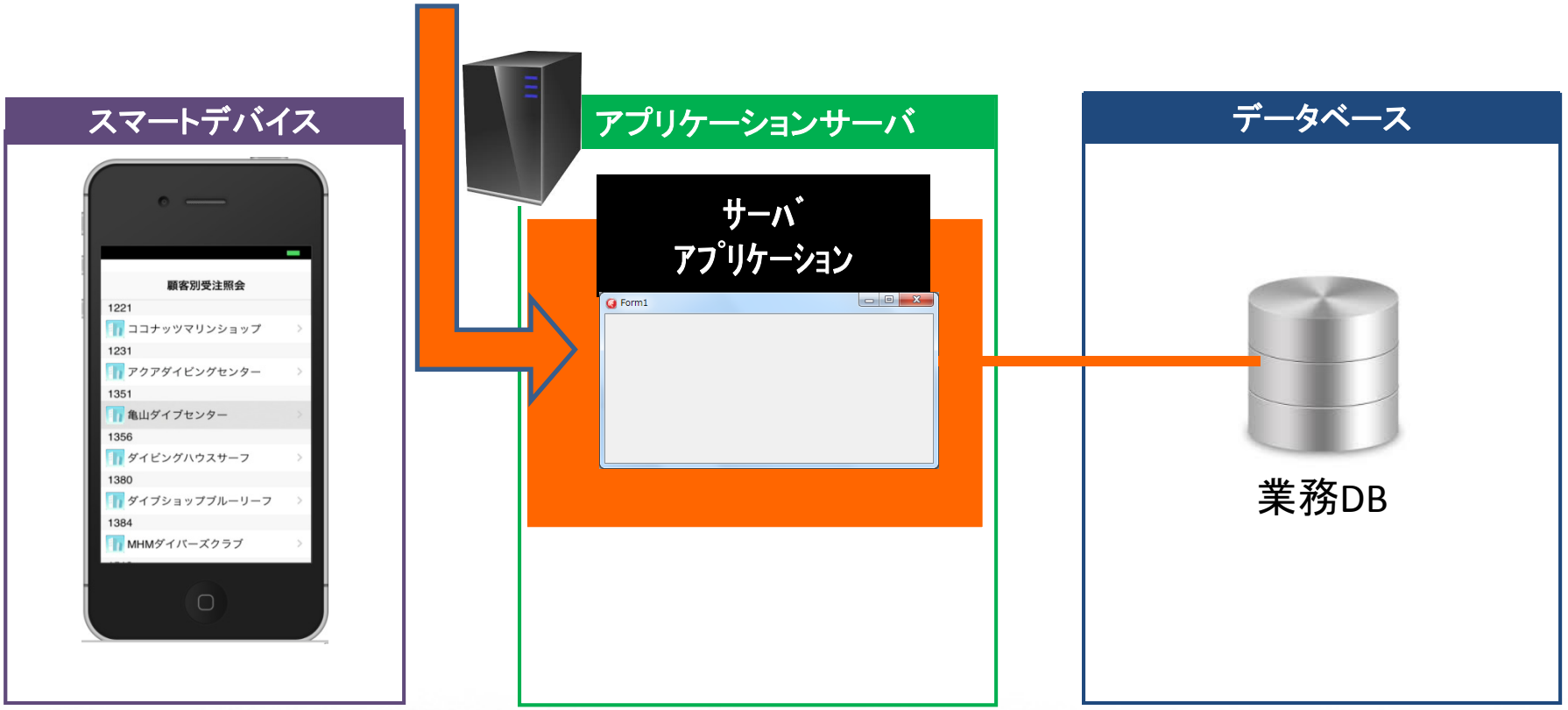
Optionプロパティの
”poAllowCommandText”を
“True”に設定



3-3. DBに接続するネイティブアプリの開発

- サーバアプリの開発手順12

完成したらコンパイルして、アプリケーションサーバ上で起動しておきます。

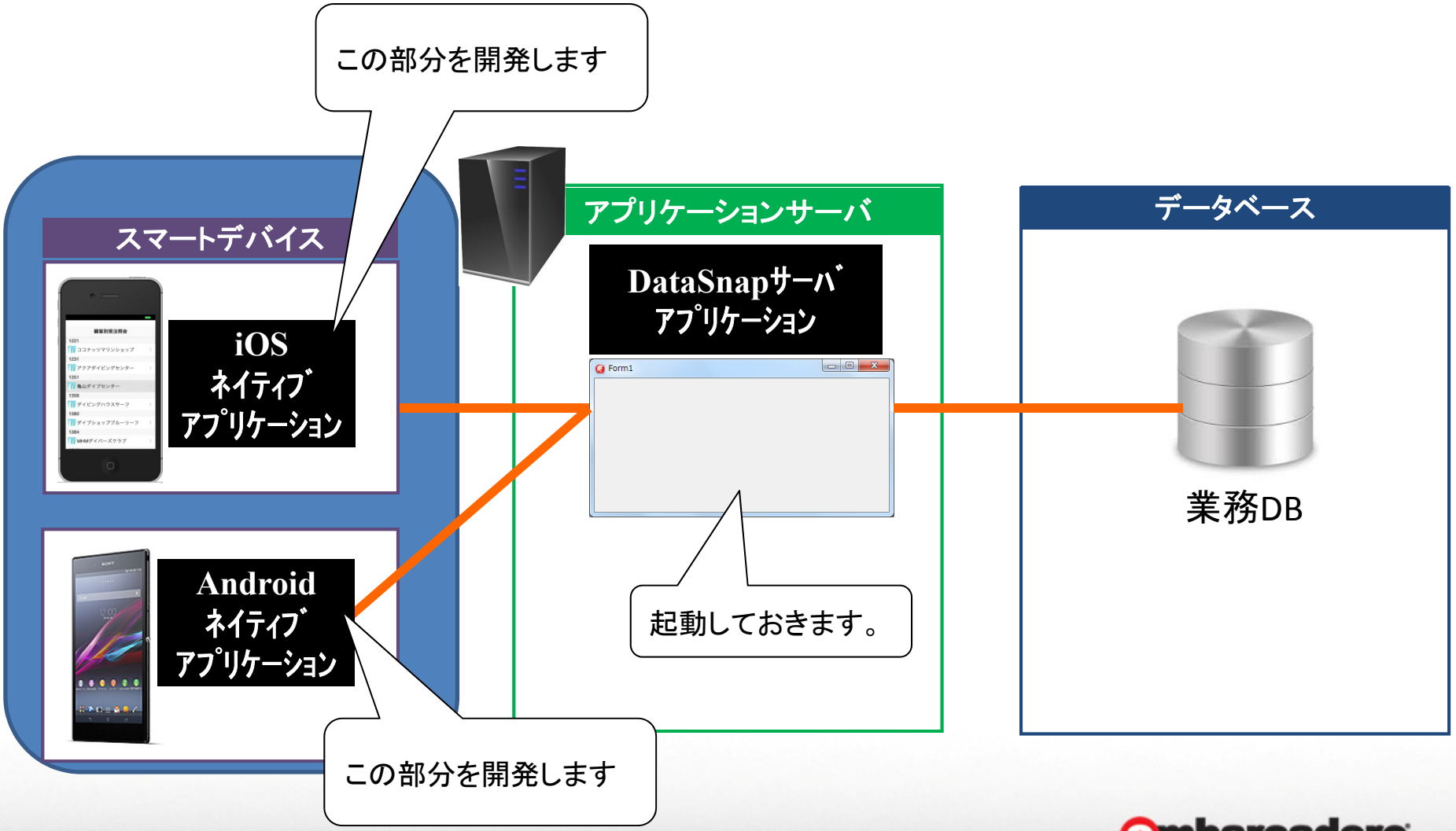


3-3. DBに接続するネイティブアプリの開発

- デモで開発するネイティブアプリケーション

3-3. DBに接続するネイティブアプリの開発

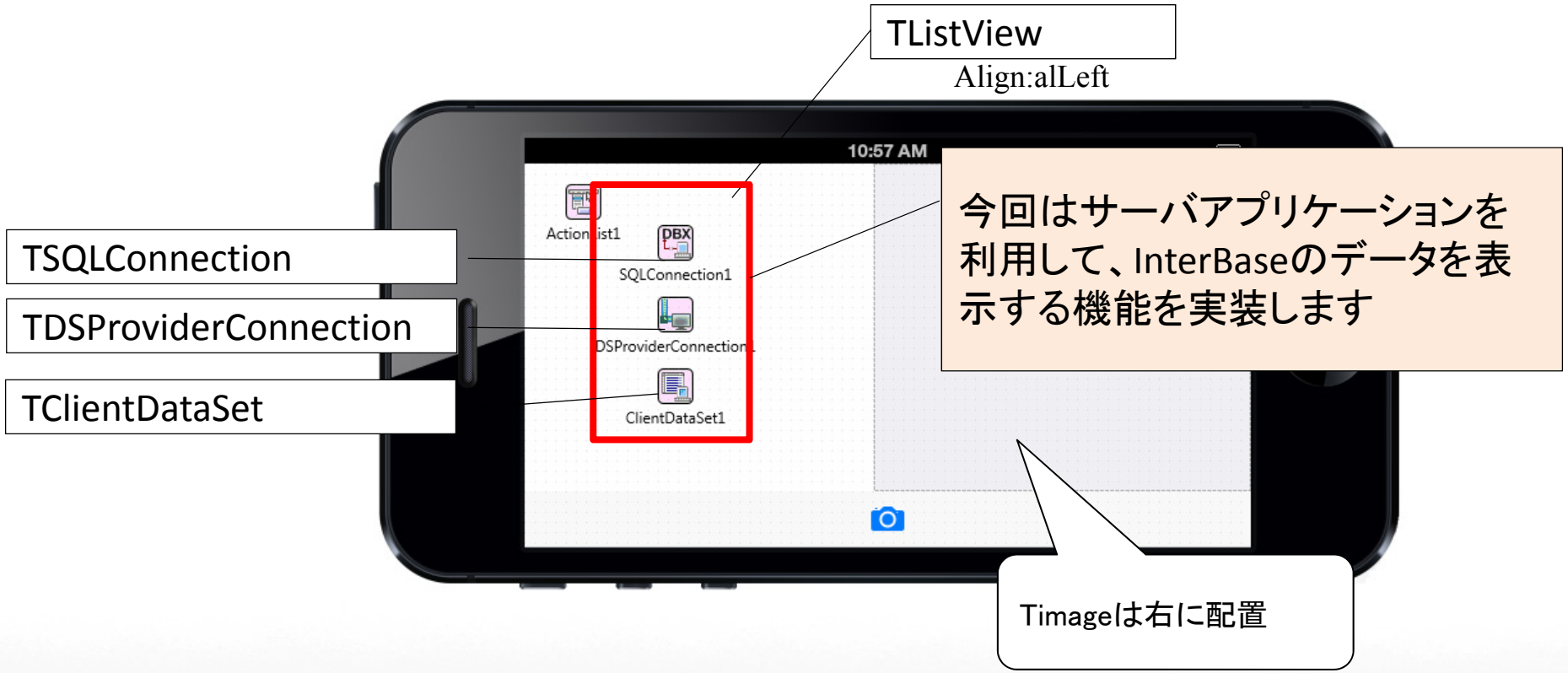
- 業務DBに接続するネイティブアプリ



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順1

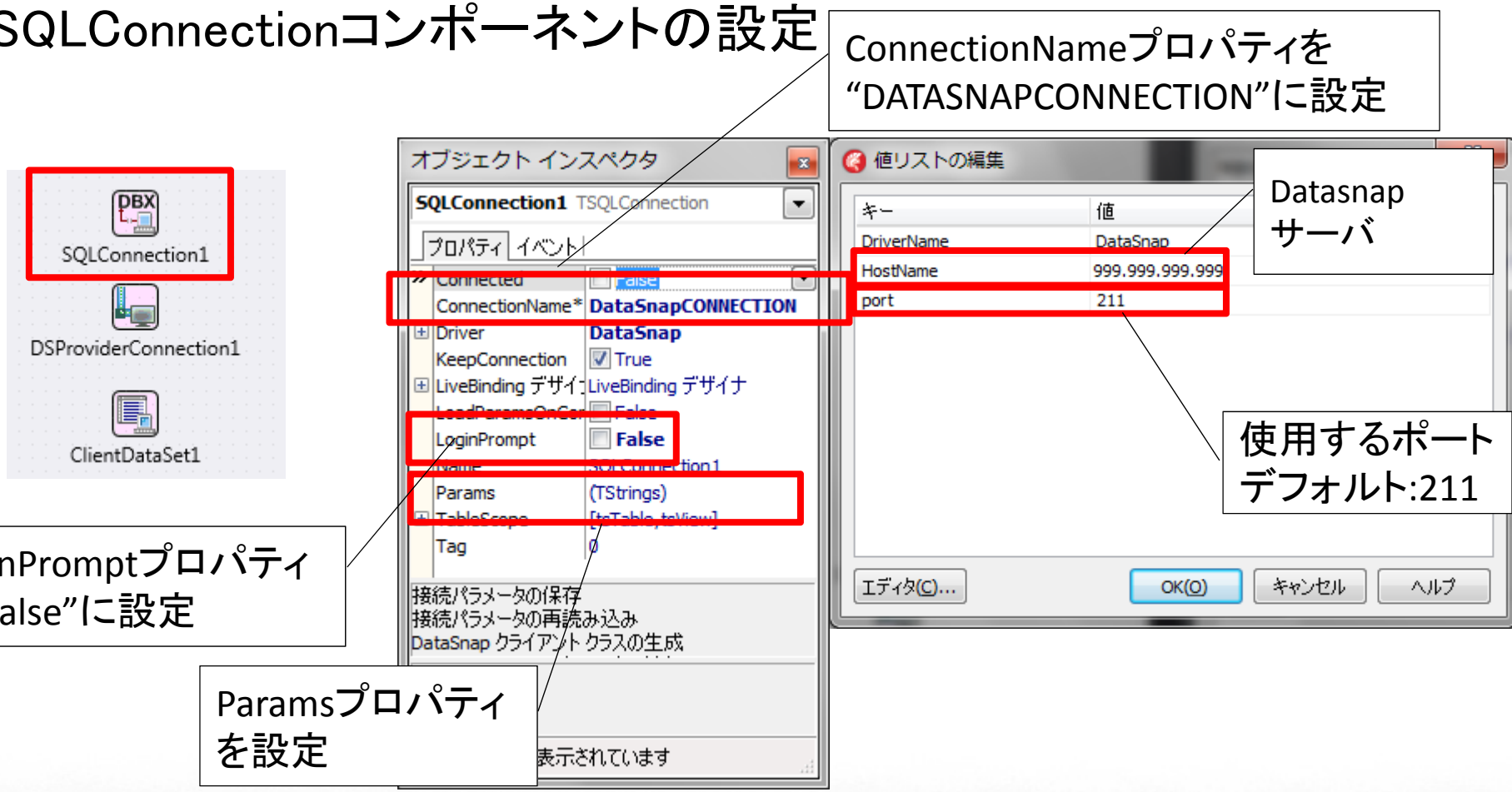
先ほど3−2.で作成した写真撮影のアプリをカスタマイズします。
TListView、TSQLConnection、TDSProviderConnection、TClientDataSetを追加配置



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順2

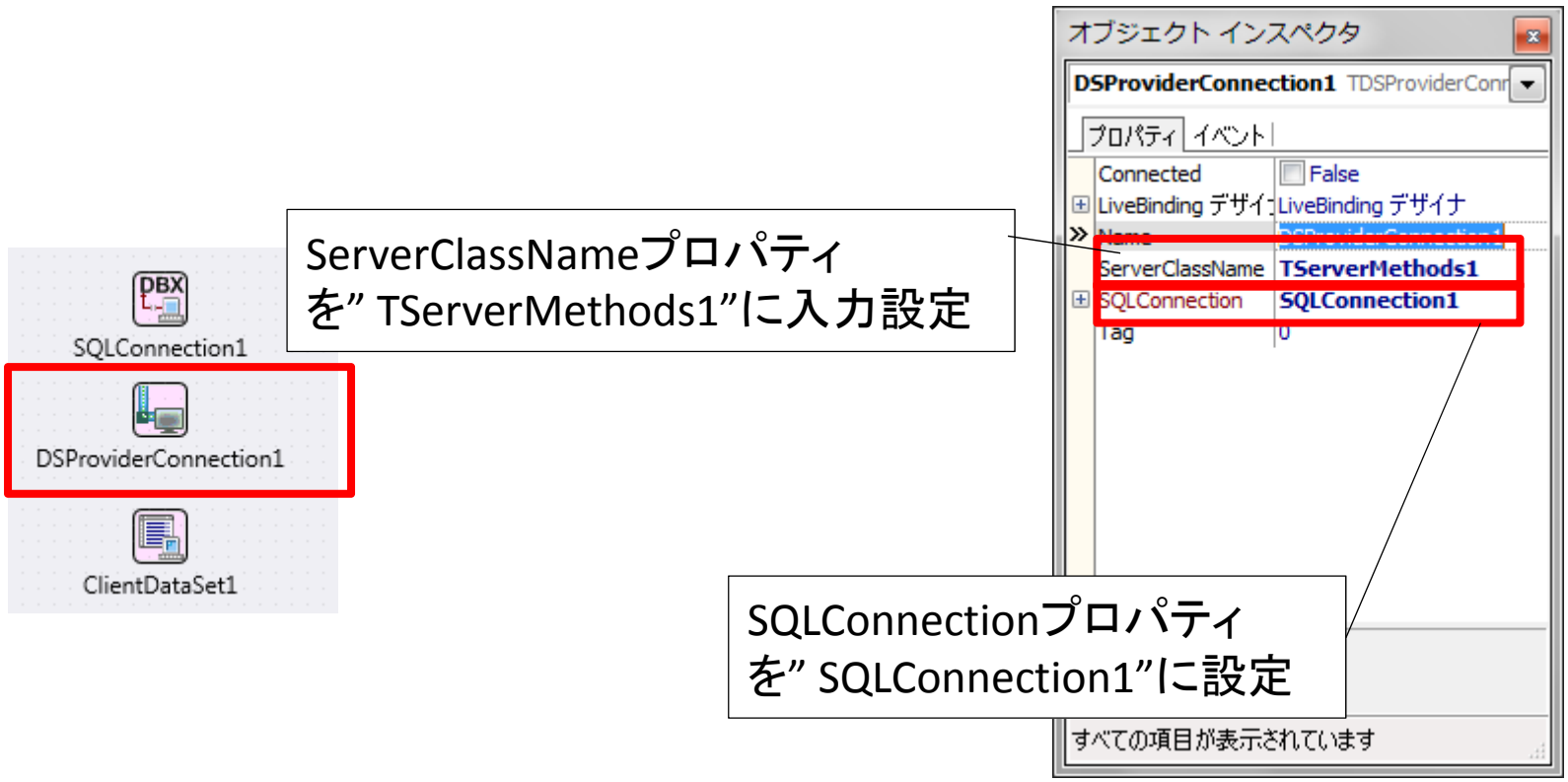
SQLConnectionコンポーネントの設定



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順3

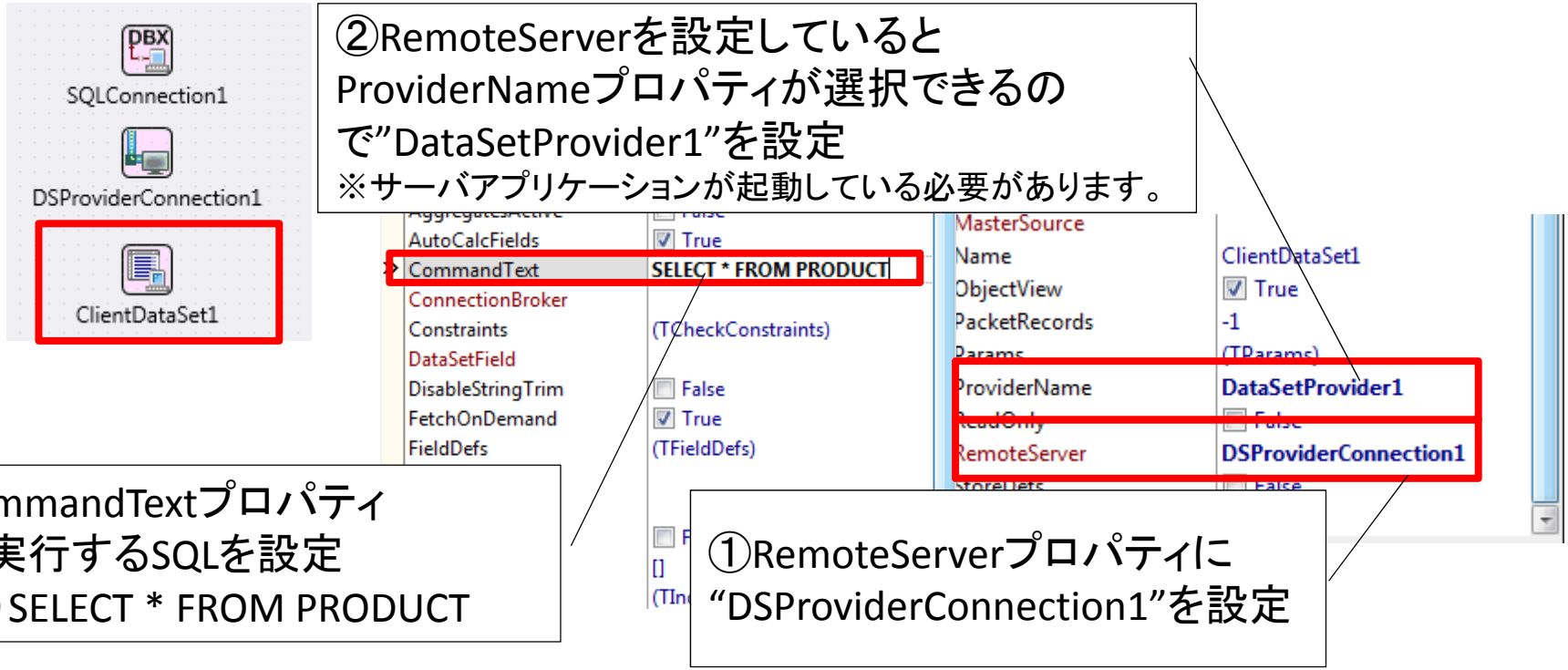
DSProviderConnectionコンポーネントの設定



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順4

ClientDataSetコンポーネントの設定

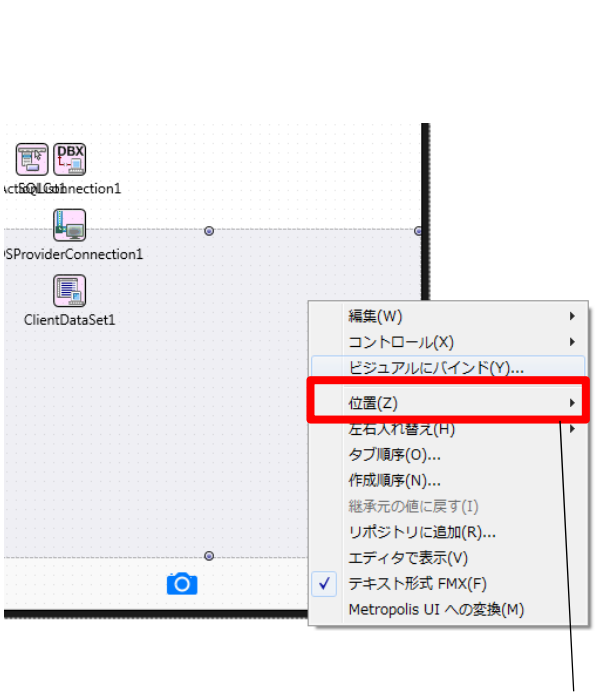


設定ができたら「すべてのフィールドの追加」で取り込み

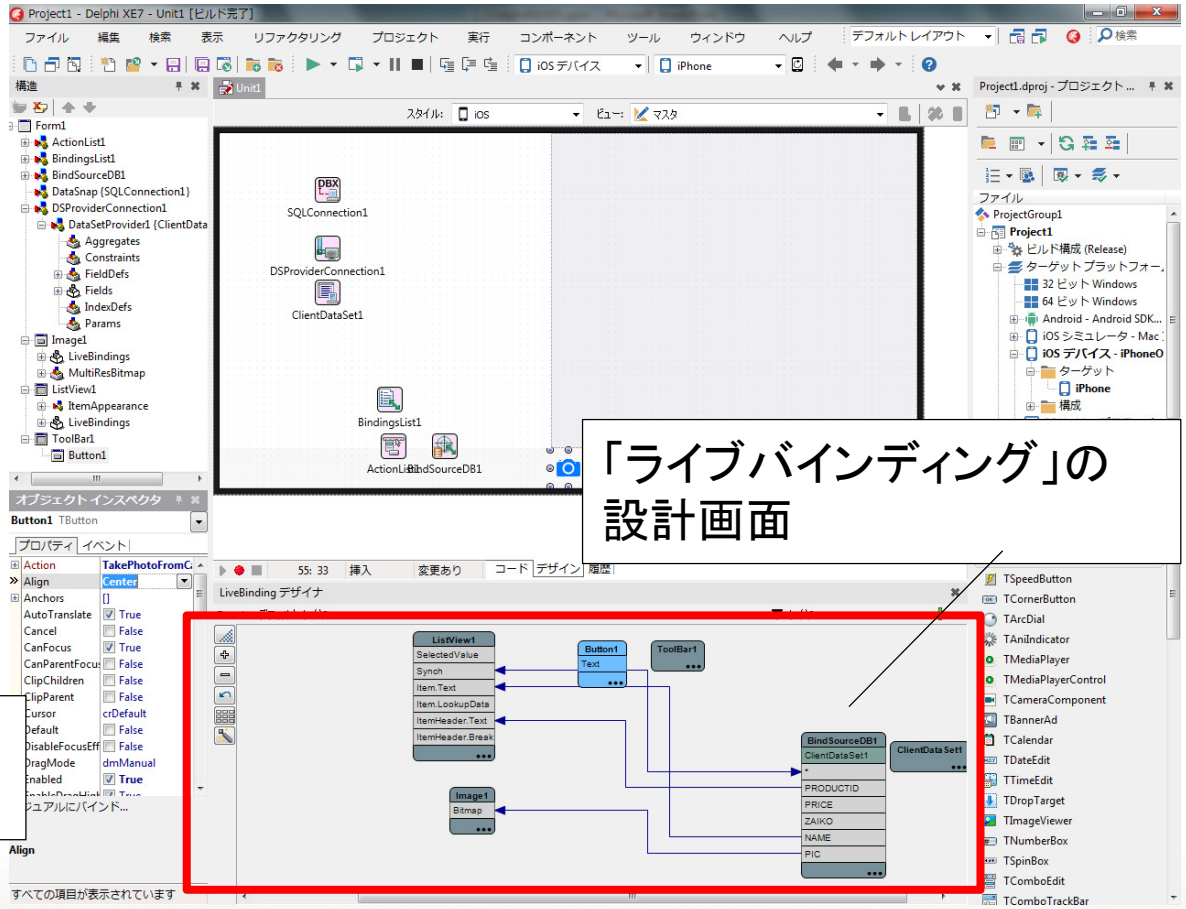
3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順5

データ表示をライブバインディング機能で実装



右クリックから
「ビジュアルにバインド」を起動

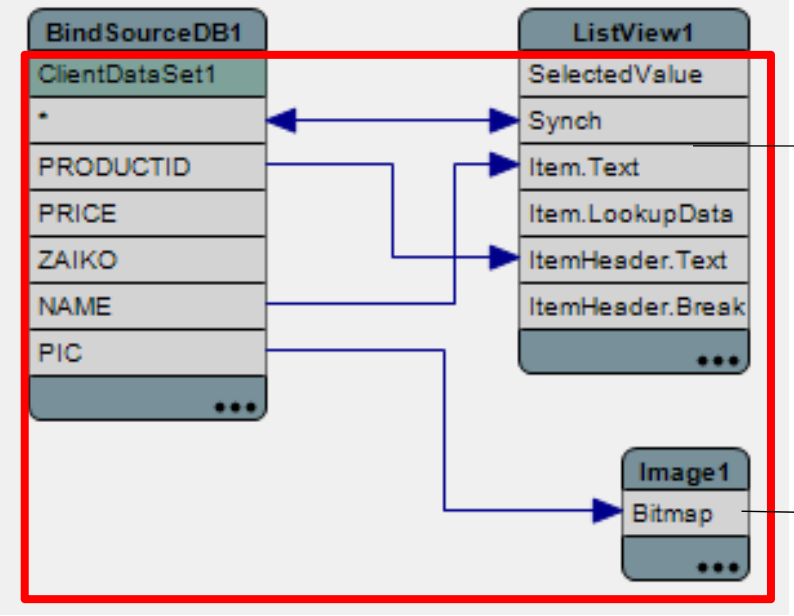


「ライブバインディング」の
設計画面

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順6

TClientDataSetの項目をTListViewとTImageに
ドラッグ&ドロップでリンク



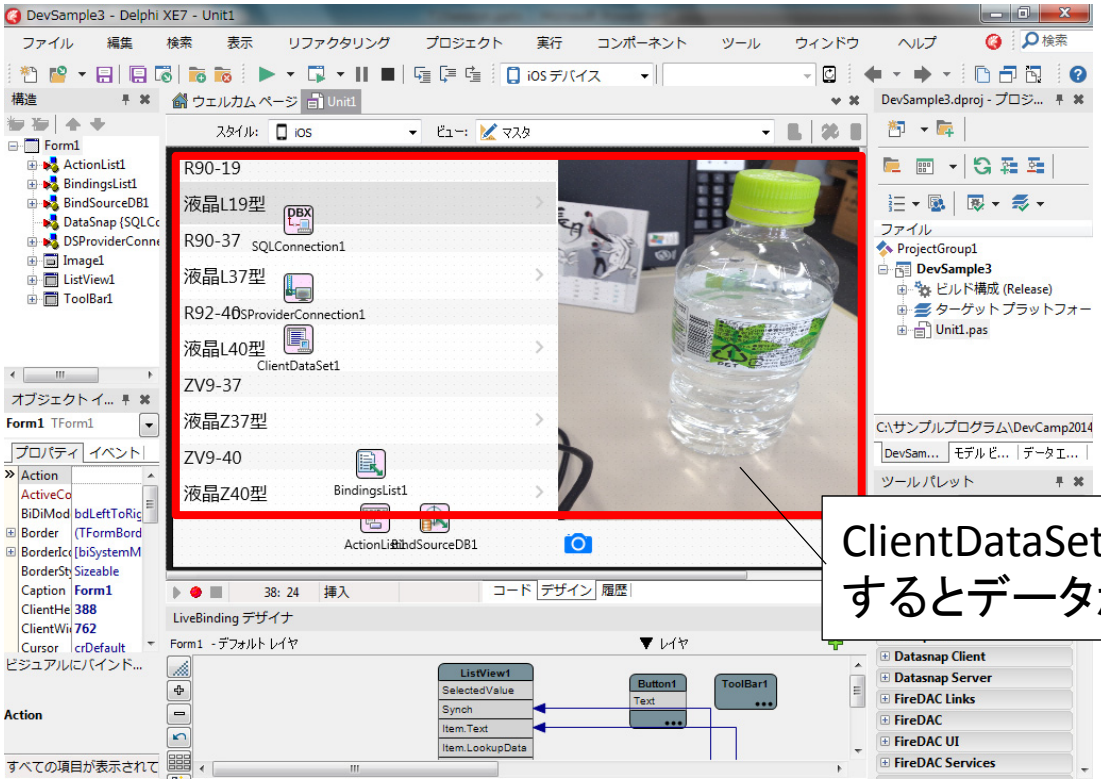
<ListView1>･･･ヘッダーにID、テキストに名称
*をSynchにリンクして同期
PRODUCTIDをItemHeaderTextにリンク
NAMEをItemTextにリンク

<Image1>･･･画像フィールド
PICをBitmapにリンク

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順7

データ表示の確認



3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順8

OnDidFinishTakingイベントをDB更新に若干変更

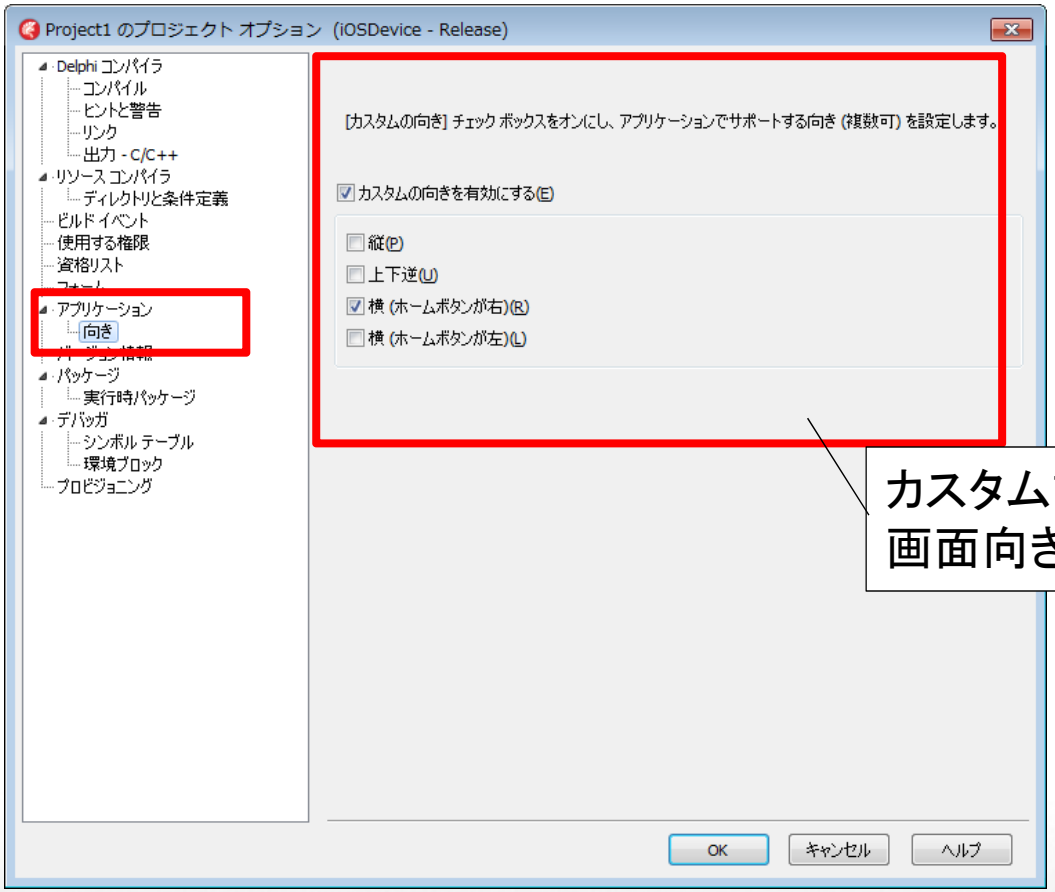
OnDidFinishTaking処理（カメラ撮影終了処理）

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
begin
  ClientDataSet1.Edit;
  ClientDataSet1.FieldByName('PIC').Assign(Image);
  ClientDataSet1.Post;
  ClientDataSet1.ApplyUpdates(-1);  //元DBへ反映
  // Image1.Bitmap.Assign(Image);
end;
```

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順9

[プロジェクト|オプション]のアプリケーションで向きを固定

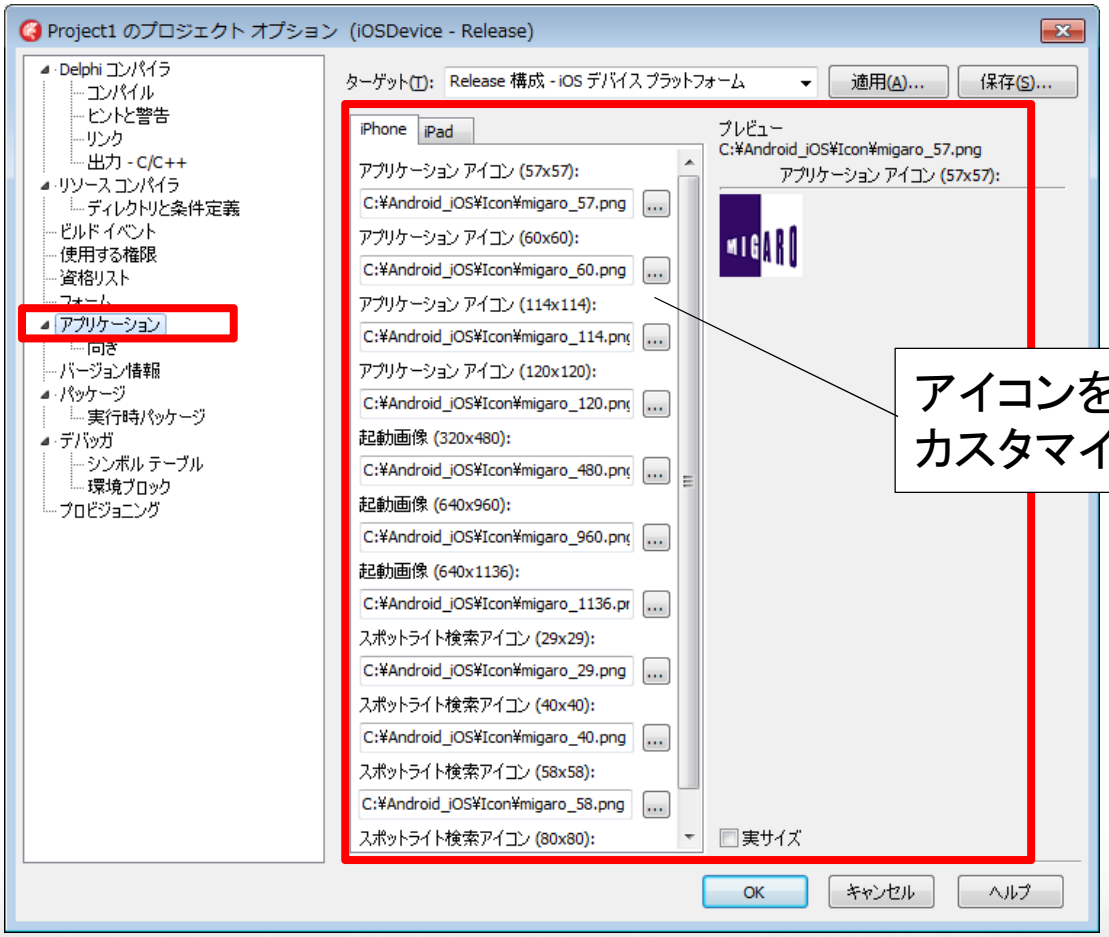


カスタム設定で
画面向きを固定できる

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順10

[プロジェクト|オプション]のアプリケーションでアイコンを設定

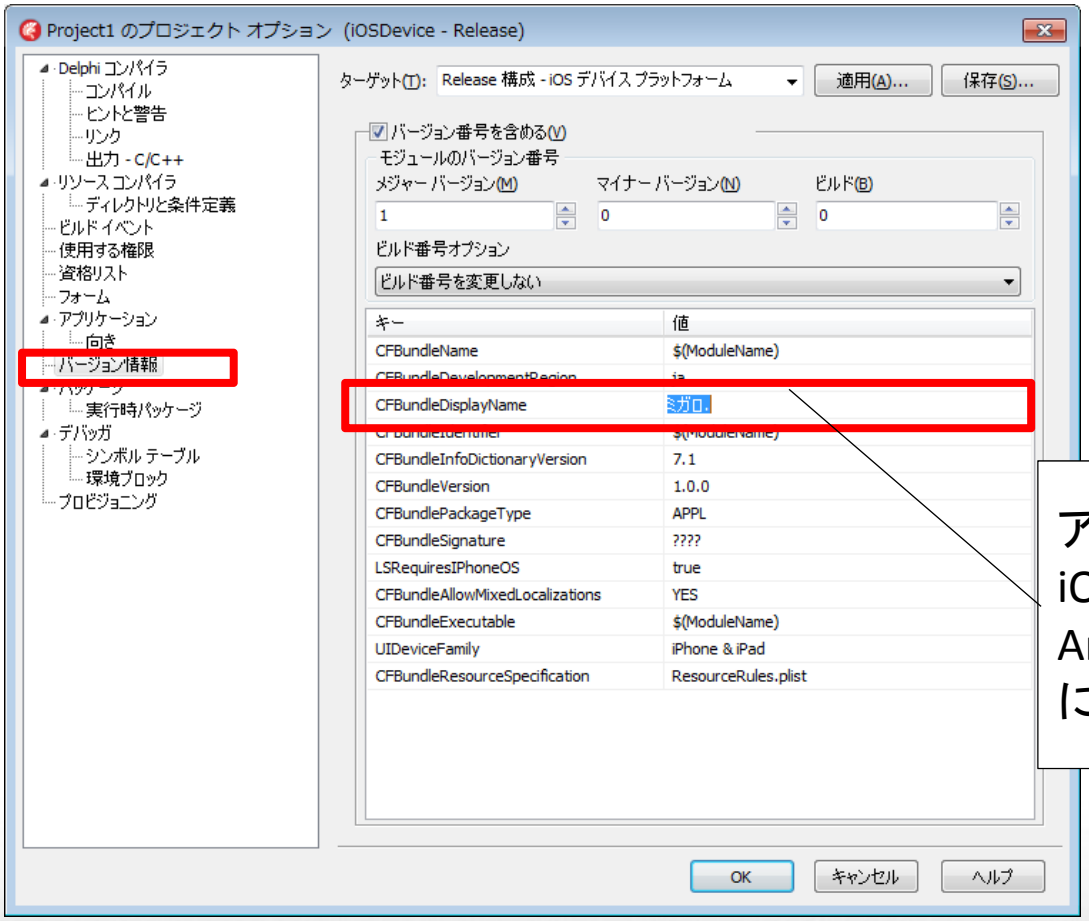


アイコンを用意しておけば
カスタマイズ可能

3-3. DBに接続するネイティブアプリの開発

- 業務DBに接続するネイティブアプリ開発手順11

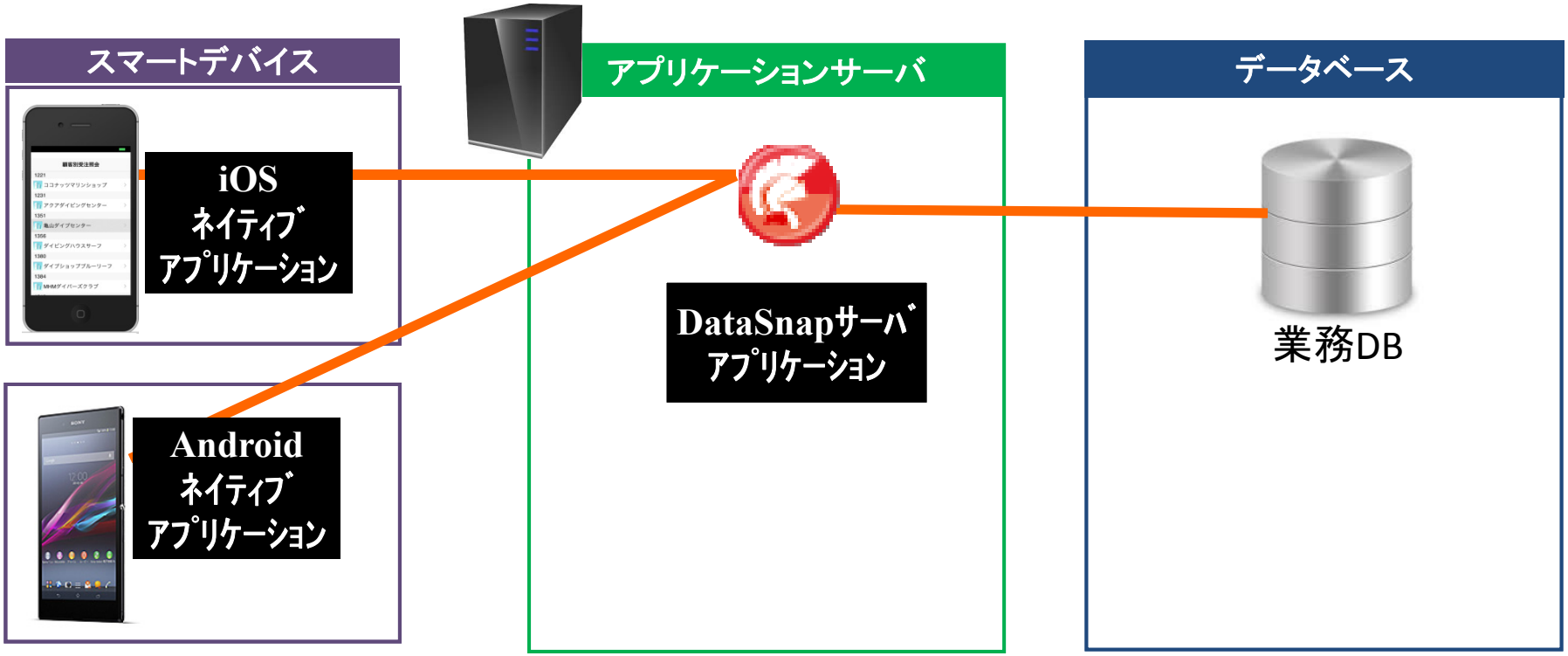
[プロジェクト|オプション]のバージョン情報設定



アプリのアイコン表示名は
iOS:CFBundleDisplayName
Android:Label
に設定

3-3. DBに接続するネイティブアプリの開発

- 3階層によってスマートデバイスからDBへ接続完成



本資料では設定内容を全て記載して頁数多いですが、
プロパティ設定をしていくだけなので簡単！
※常時接続が安定しない環境であれば、都度接続処理が◎

3-4.ネイティブアプリケーションの配布

- ネイティブアプリはスマートデバイスに
  配布･インストールする方法が2つあります。

社内公開での配布

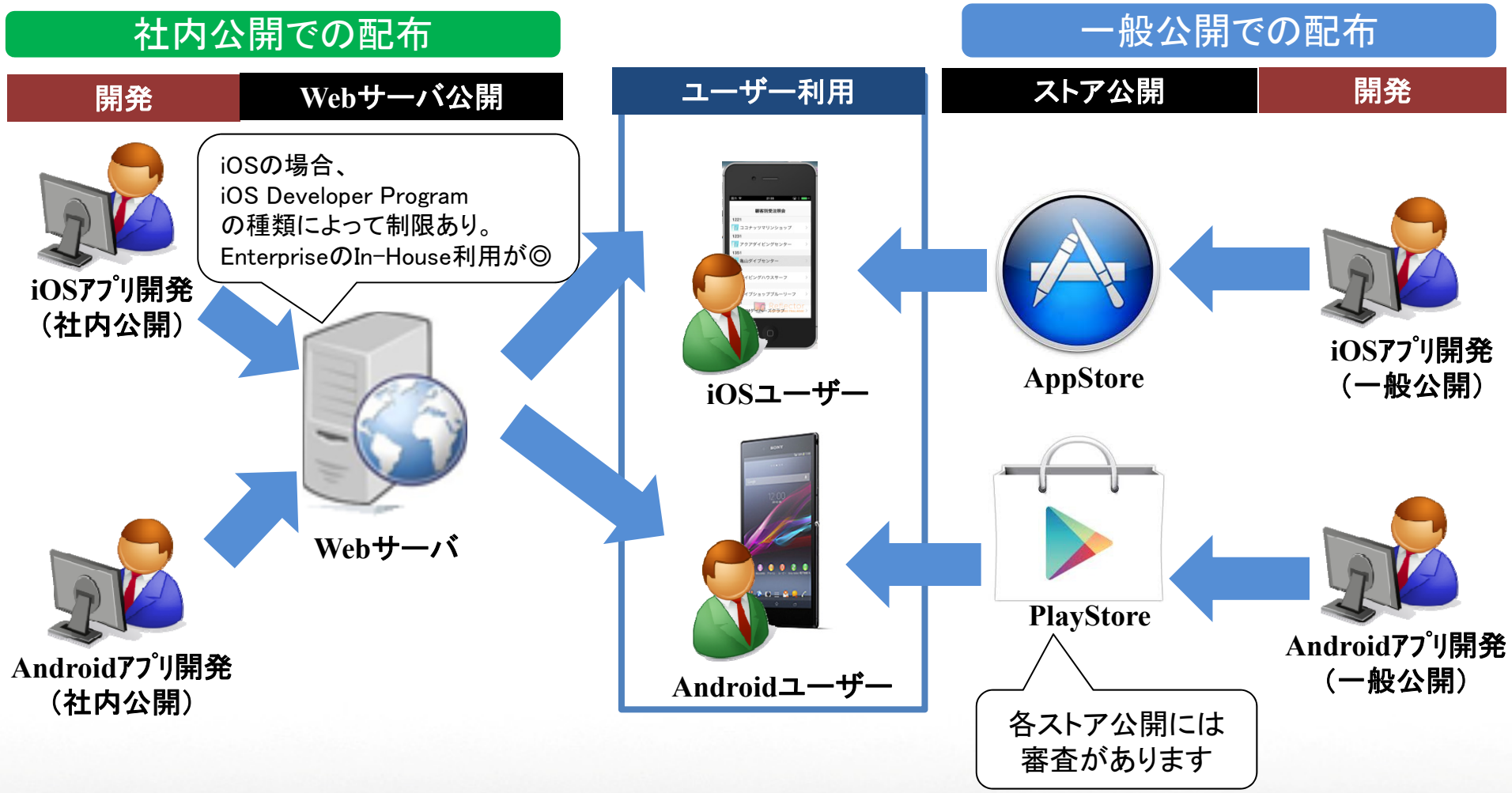
開発者はWebサーバ上にネイテブアプリのファイルを公開して
配布します。

一般公開での配布

開発者はiOSであればAppStore、AndroidであればPlayStore
にネイティブアプリを公開して配布します。

3-4.ネイティブアプリケーションの配布

- 社内公開と一般公開の配布方法の違い



3-4.ネイティブアプリケーションの配布

- 社内公開と一般公開のメリット/デメリット

	社内公開	一般公開
メリット	・社内だけで配布・利用できる。 ・審査がないため、社内専用のアプリケーションが開発できる。	・ストアで公開するため、どこからでもすぐにインストールして利用できる。
デメリット	・Webサーバ等を用意して、配布環境の構築・運用が必要。	・誰でも利用できてしまう。 ・公開には審査が必要。 （自社用アプリの公開は難しい）

3-4.ネイティブアプリケーションの配布

- 配布
  社内公開するアプリは、配布用にコンパイルしたファイルをWebサーバ等に配置して、リンクでダウンロードできるように準備します。

- 配布Webサーバ
  関連する拡張子はMIMEタイプを事前登録しておきます。

```
＜iOS＞
  ipaファイル・・・・・・・application/octet-stream
  plistファイル・・・・・・text/xml

<android>
  apkファイル・・・・・・application/vnd.android.package-archive
```

iOS7.1からplistファイルの配布にhttps（SSL）が必須です。
Dropbox等で代用もできます。

73

3-4.ネイティブアプリケーションの配布

- 配布ファイル
  社内公開するアプリは、配布用にコンパイルしたファイルをWebサーバ上に配置して、リンクでダウンロードできるように準備。

【Android配布用のファイル】
apkファイル

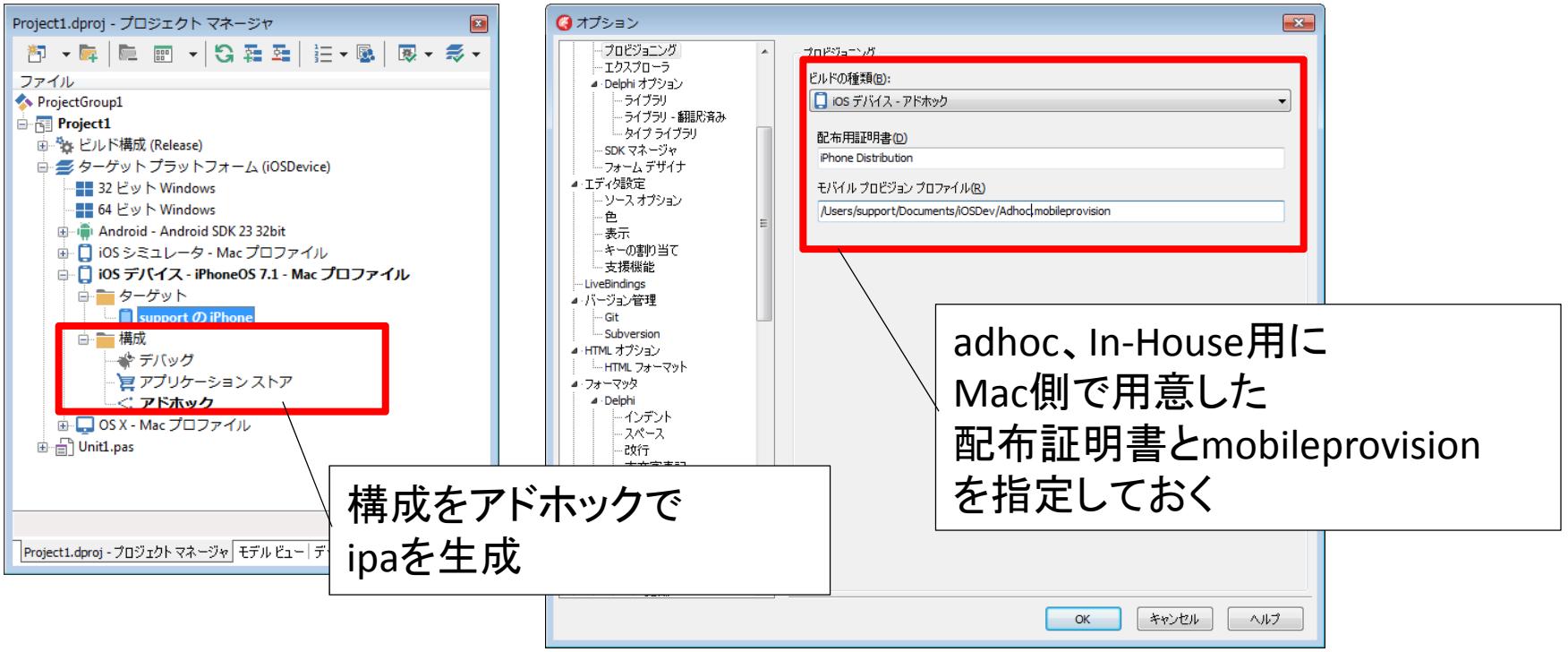
ストアアプリではないので、
[提供元不明のアプリ]の許可が必要

ダウンロード用HTML例(Android)

```
<h1>Androidダウンロードサイトサンプル</h1>
<form>
   <a href="./ Project1.apk" type="application/vnd.android.package-archive">
    アプリケーションダウンロード</a><br>
</form>
```

3-4.ネイティブアプリケーションの配布

【iOS配布用のファイル】
iOS:　ipa、plistファイル



配布証明書やmobileprovisionファイルは適切に設定できていないとエラーになるので、Mac環境でキーチェーンやXcode上の認証をしっかり確認が必要。

3-4.ネイティブアプリケーションの配布

ダウンロード用plist例

```
<key>assets</key>
<array>
        <dict>
                <key>kind</key>
                <string>software-package</string>
                <key>url</key>
                <string>https://Webサーバ /Project1.ipa</string>
        </dict>
</array>
```

ipaを配置したurlを指定

ダウンロード用HTML例(iOS)

```
<h1>iOSダウンロードサイトサンプル</h1>
 <form>
   <a href="itms-services://?action=download-manifest&url=https://Webサーバ /Project1.plist">
    アプリケーションダウンロード</a><br>
 </form>
```

3-4.ネイティブアプリケーションの配布

- インストール
  配布Webサーバにアクセスしてリンクからダウンロード&インストール。
  例）iOS7.1

77

まとめ

4.まとめ

- 企業でのスマートデバイス導入も増えてきて
  スマートデバイス用の業務アプリケーション需要も高い
- Delphiなⅰら主流のiOSとAndroidの開発が簡単
- 業務アプリケーションの社内DB接続はDataSnapサーバを使った3階層方式
- 配布は社内公開と一般公開で方法が異なるので
  しっかり環境想定・準備が必要

ご静聴ありがとうございました