

【セッションNo. 2】

Delphi/400 最新技術情報

Delphi/400 XE5

—こんなに簡単！IBM iスマートデバイスネイティブ開発—

株式会社ミガロ.

RAD事業部 技術支援課

吉原 泰介

【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ケーションの種類
3. ネイティブアプリケーションの開発
 - 3-1. ネイティブアプリケーションの開発環境
 - 3-2. 簡単なネイティブアプリケーションの開発
 - 3-3. IBM iに接続するネイティブアプリケーションの開発
 - 3-4. ネイティブアプリケーションの配布
4. まとめ

1.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

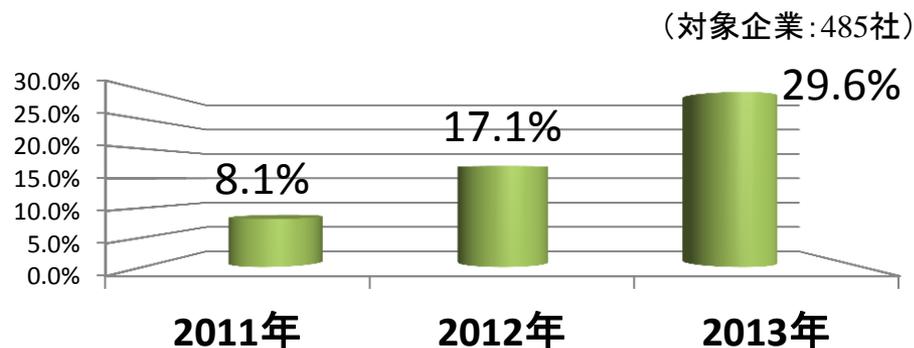
• スマートデバイス企業導入率と導入OSの傾向

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みます。

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。

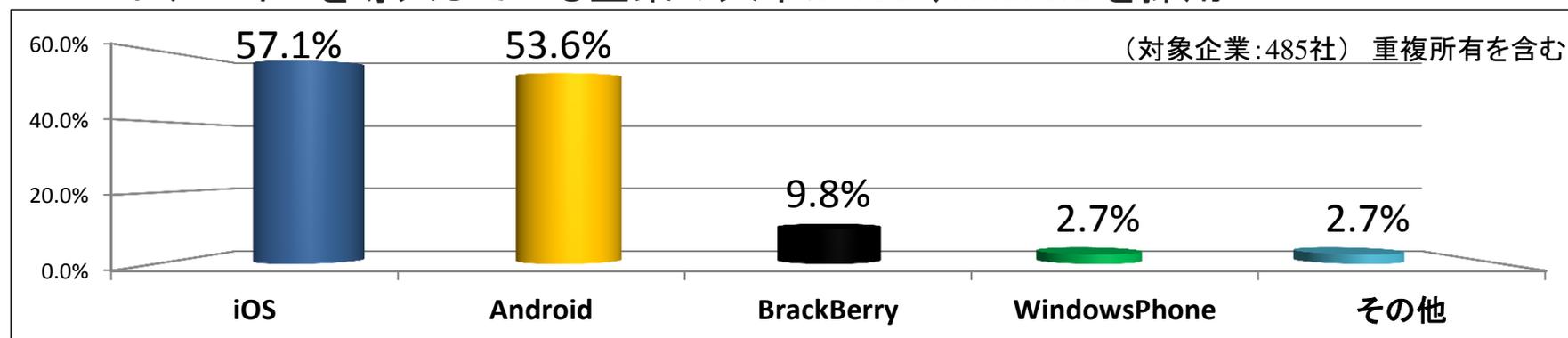
スマートデバイス企業導入率の遷移

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



参考: キーマンズネット業務用スマートフォン導入状況2013

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ケーションの業務利用**
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ケーションも業務利用が増加

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



PCアプリケーションの特徴

- 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ケーションの特徴

- 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

2.スマートデバイスアプリケーションの種類

2.スマートデバイスアプリケーションの種類

- 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ケーションは、大きく2種類のアプリケーションに分かれます。

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ケーションの種類

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 GooglePlay といったストアや、社内向けに公開したWebサーバからインストールして利用することができます。



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

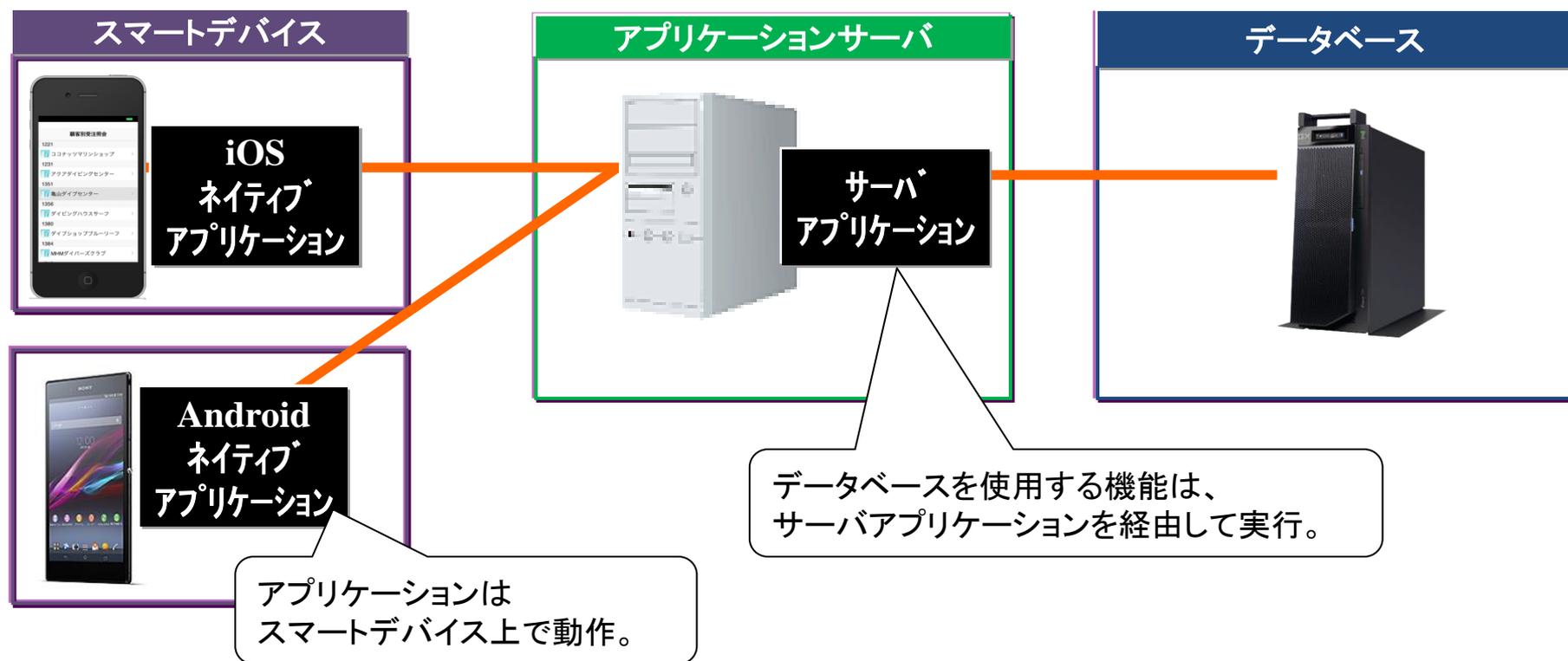
- ・iOS: Objective-C
- ・Android: Java等



2.スマートデバイスアプリケーションの種類

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ケーションの種類

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。

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

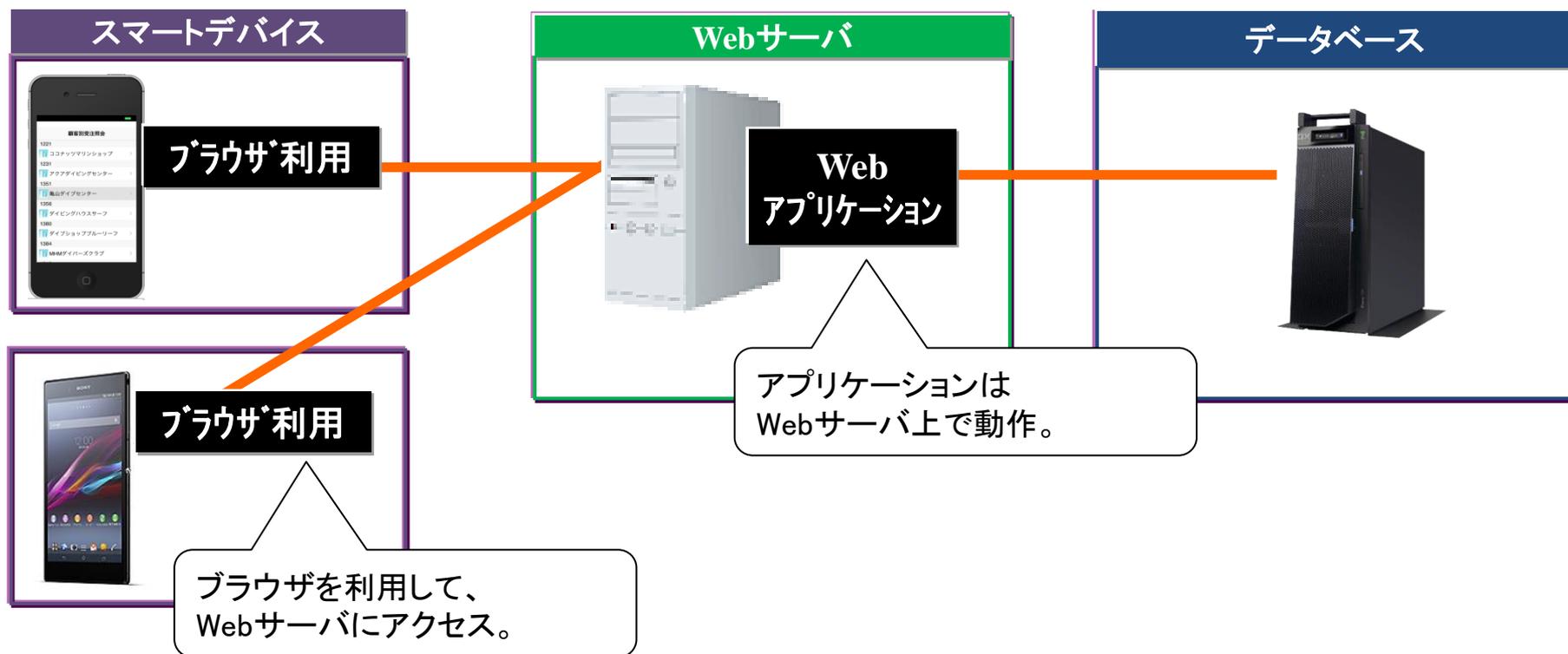
- ・HTML5、JavaScript、PHP、java等



2.スマートデバイスアプリケーションの種類

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ケーションの種類

• ネイティブ / Webアプリケーションの特徴

	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	△	◎

アプリケーションによって
多くの開発言語習得が必要

言語によって開発環境が
煩雑になるため、
生産性が低くなる

2.スマートデバイスアプリケーションの種類

- ネイティブ / Webアプリケーションの特徴 (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ケーションの種類

- Delphi/400 XE5 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 両方のスマートデバイスに対応できます。

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ケーションの種類

- ネイティブアプリケーション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



バーコード読取



GPS位置情報取得



2.スマートデバイスアプリケーションの種類

- ネイティブアプリケーション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ケーションの種類

- ネイティブアプリケーション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ケーション



2.スマートデバイスアプリケーションの種類

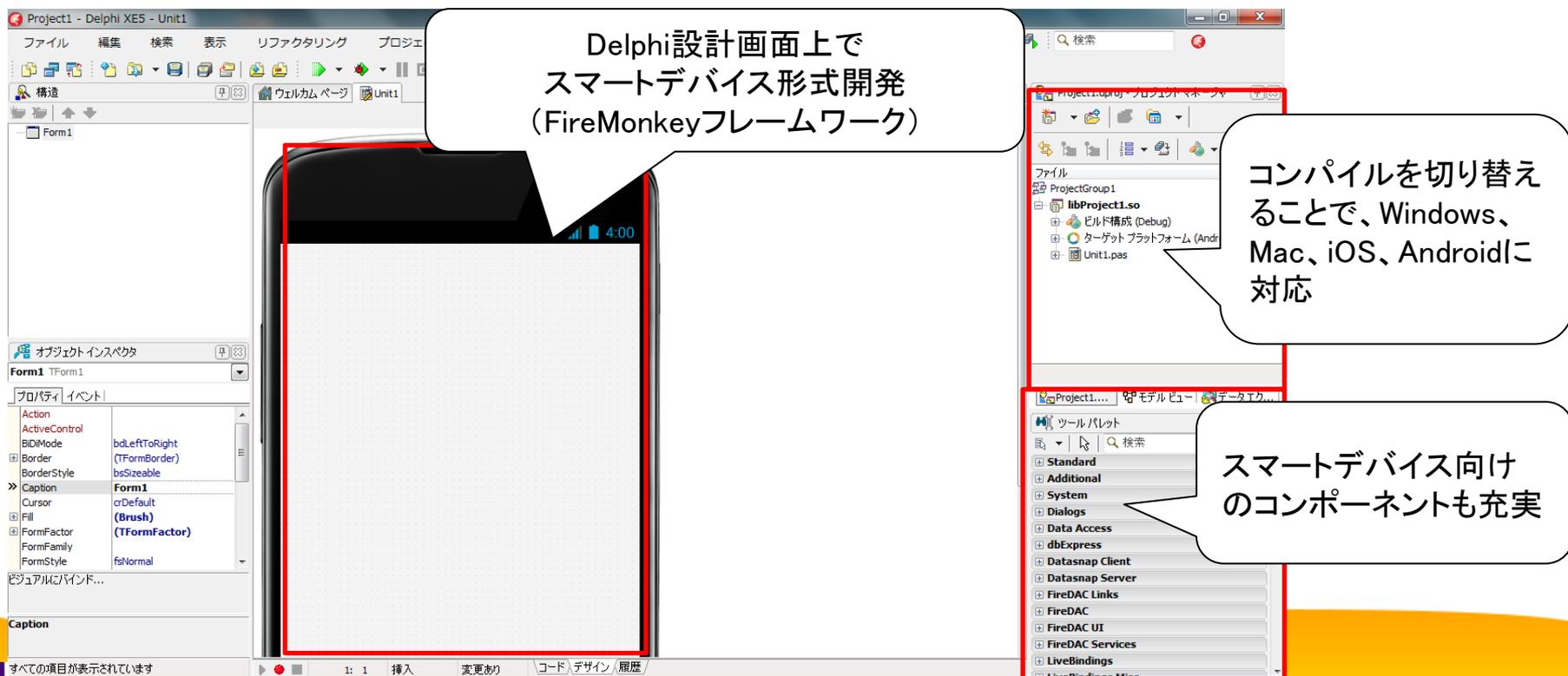
- ネイティブアプリケーションのデバイス機能連携例3
GPSを使って地図連携を活用したネイティブアプリケーション



3.ネイティブアプリケーションの開発

3-1. ネイティブアプリケーションの開発環境

- Delphi/400 XE5 のネイティブアプリケーション開発機能
Delphi/400 XE5 では、Windowsアプリケーションの開発機能に加え、スマートデバイスのネイティブアプリケーション開発機能が追加されました。FireMonkeyフレームワークを利用することで、Delphiスキルを使ってiOS、Androidアプリケーションが開発できます。



3-1. ネイティブアプリケーションの開発環境

- iOSネイティブアプリケーション開発に必要な環境
 - Windows端末 (Delphi/400 XE5)
 - Mac端末 (OSX 10.7~10.9)
 - iOS Developer Program (Xcode, 配布)
 - iOS実機 (iPhone、iPad等 iOS6.0~7.1)



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

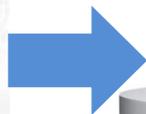
開発

コンパイル

インストール・実行



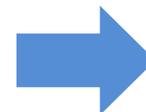
Windows



Xcode



Mac



iPhone



iPad

3-1. ネイティブアプリケーションの開発環境

- Androidネイティブアプリケーション開発に必要な環境
 - Windows端末 (Delphi/400 XE5)
 - Android実機
(Android 2.3.3以降のARM7 + NEON対応デバイス)

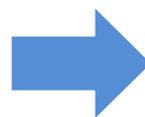
開発

コンパイル

インストール・実行



Windows



Android
Phone/Tablet

3-2. 簡単なネイティブアプリケーションの開発

- 簡単なネイティブアプリケーションの開発

この部分だけを開発します

スマートデバイス



iOS
ネイティブ
アプリケーション



Android
ネイティブ
アプリケーション

アプリケーションサーバ



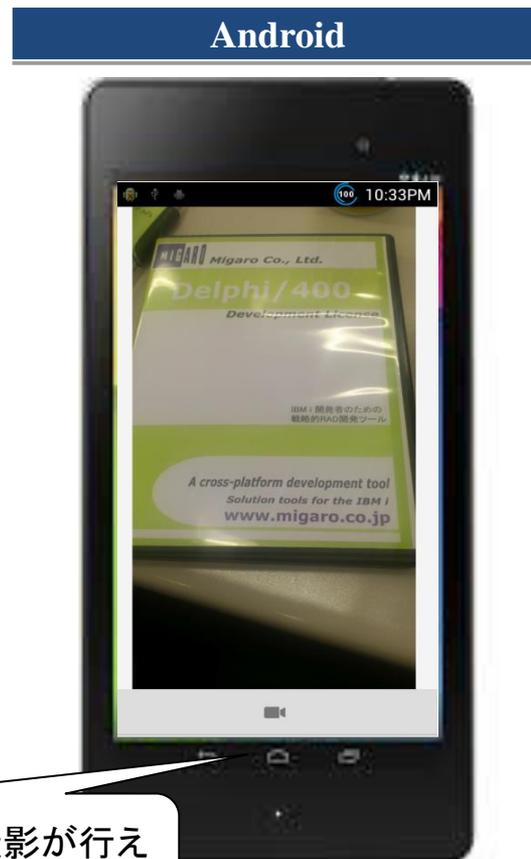
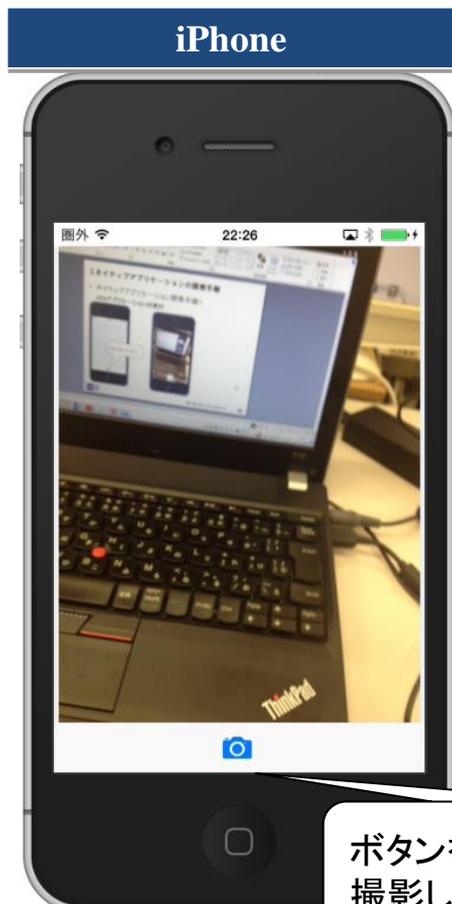
サーバ
アプリケーション

DB



3-2. 簡単なネイティブアプリケーションの開発

- 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



ボタンをタッチするとカメラで撮影が行え
撮影した写真を画面に取り込みます。

3-2. 簡単なネイティブアプリケーションの開発

• 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開発手法と同じです。

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



```
Unit3
begin
  procedure TForm3.Button1Click(Sender: TObject);
  begin
    Button1.Text := 'ボタン名称';
  end;
end.
```

Delphi言語で
プログラミング



3-2. 簡単なネイティブアプリケーションの開発

• ネイティブアプリケーション開発手順1

新規作成よりモバイルアプリケーションを選択

The image shows a screenshot of the Delphi XE5 IDE. The 'File' menu is open, and the 'New' option is selected. The 'New' submenu is visible, showing several application templates. The 'FireMonkey Mobile Application - Delphi (I)' option is highlighted with a red box. A green arrow points from this option to the 'FireMonkey Mobile Application' dialog box. The dialog box displays several application templates, with the 'Empty Application' option highlighted by a red box. A callout box points to the dialog box with the text '数種類のテンプレートが用意されています。' (Several templates are prepared).

Delphi XE5

ファイル(E) 編集(E) 検索(S) 表示(V) リファクタリング(Q) プロジェクト(P) 実行(R) コンポーネント(C)

新規作成(N)

開く(O)...

プロジェクトを開く(J)... Ctrl+F11

バージョン管理リポジトリから開く(Z)...

開き直す(R)

VCL フォーム アプリケーション - Delphi(V)

VCL Metropolis UI アプリケーション - Delphi(L)

FireMonkey デスクトップ アプリケーション - Delphi(F)

FireMonkey モバイル アプリケーション - Delphi(I)

FireMonkey Metropolis UI アプリケーション - Delphi(R)

FireMonkey モバイル アプリケーション

FireMonkey モバイル アプリケーション

FireMonkey モバイル アプリケーションの種類を選択します。

空のアプリケーション

3D アプリケーション

スマートフォン用マスター詳細

タブ

タブレット用マスター詳細

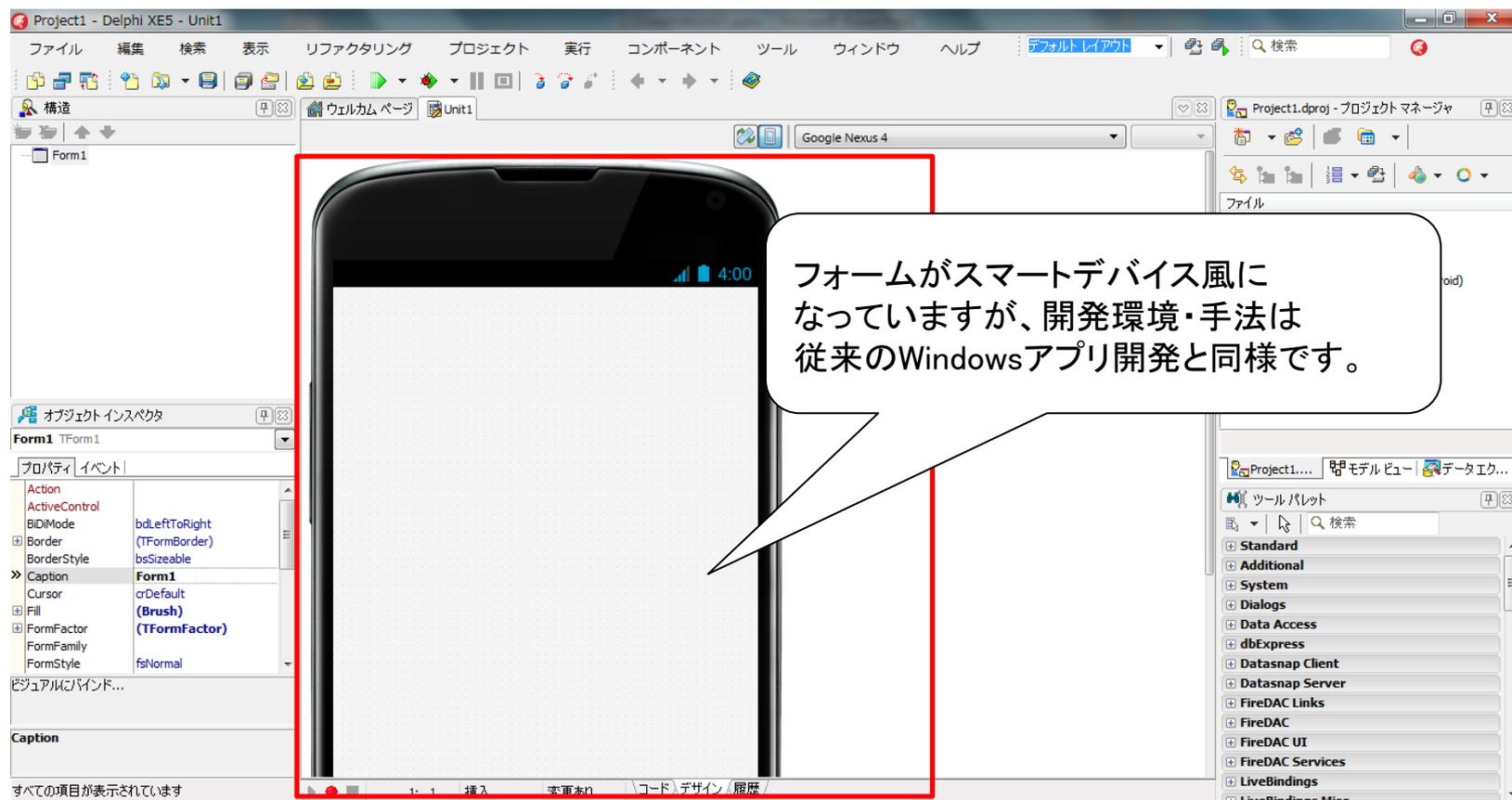
レスポンス付きタブ

数種類のテンプレートが用意されています。

OK キャンセル ヘルプ

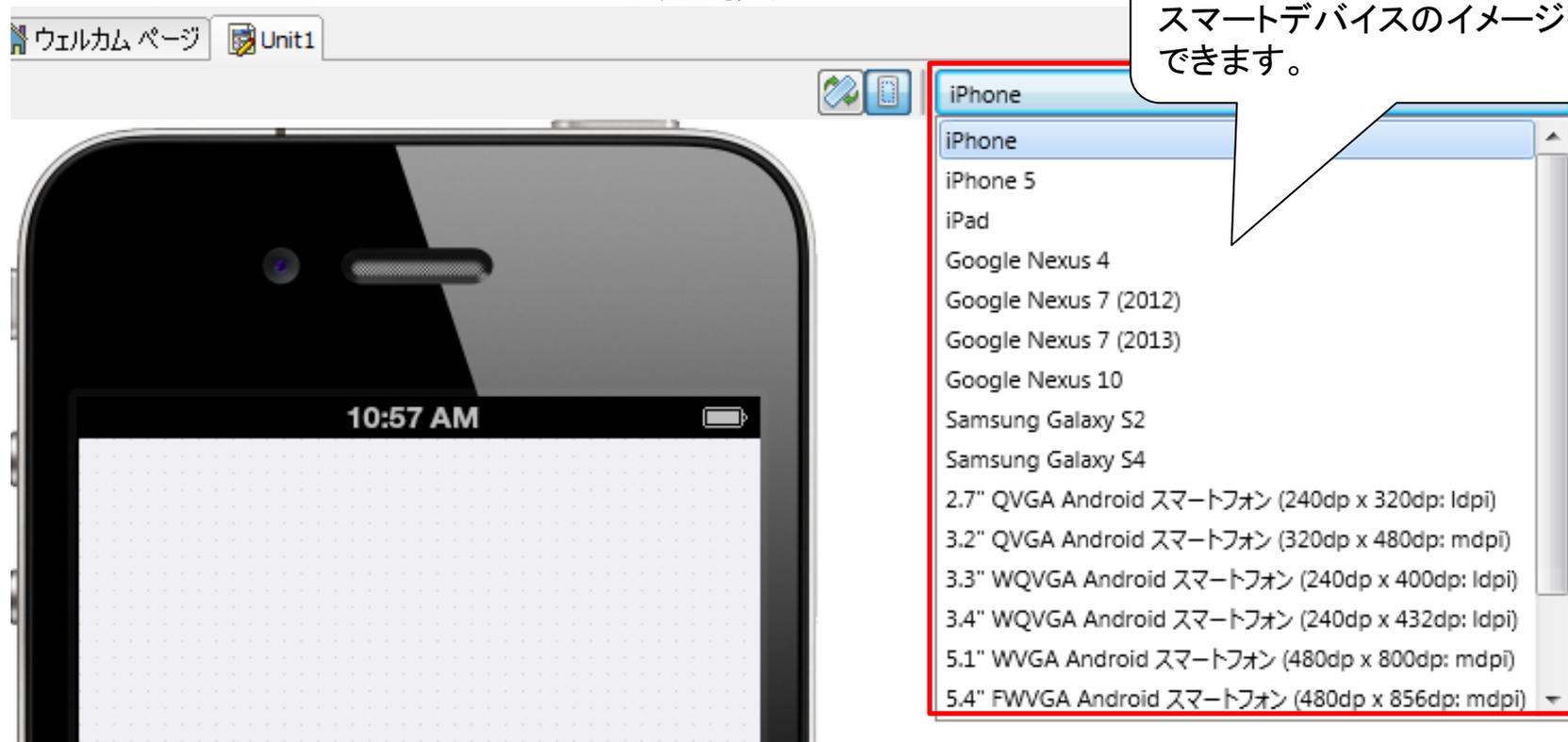
3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順2
ネイティブアプリケーション開発画面



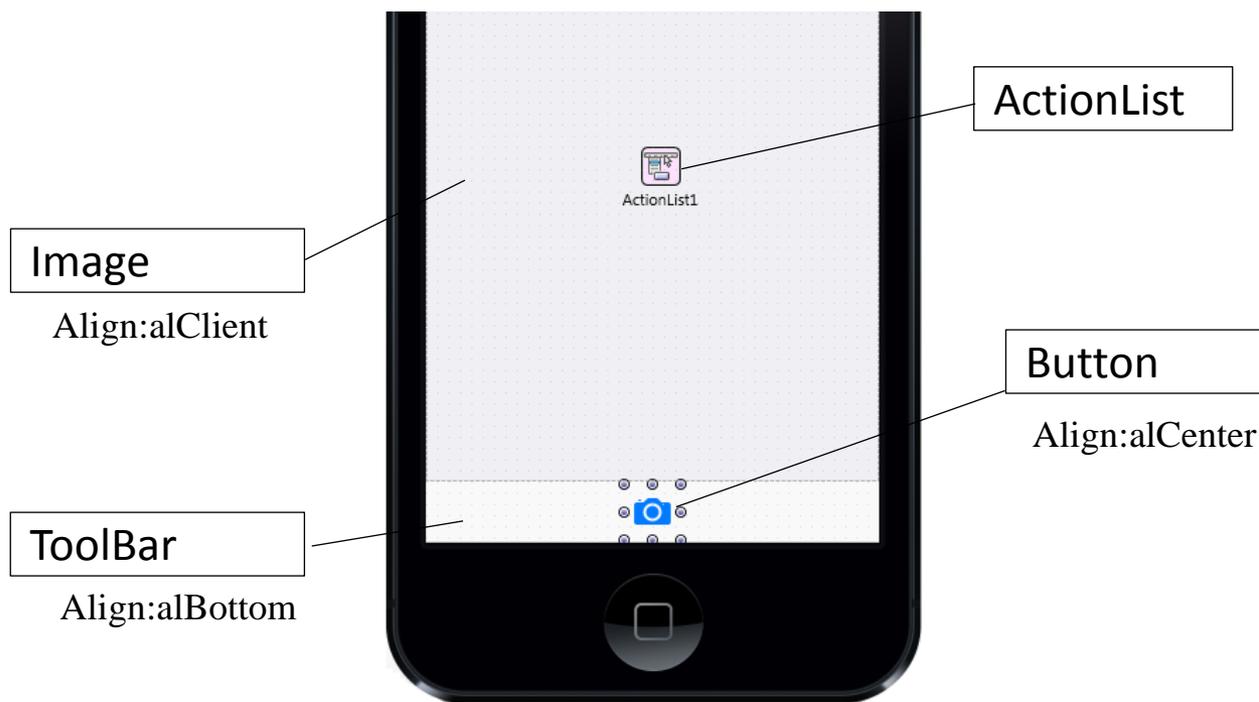
3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順3
フォームテンプレートを選択

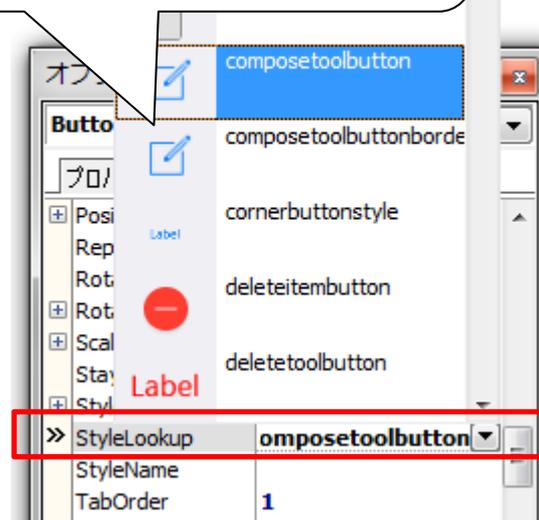


3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順4
フォームに次のコンポーネントを配置
ToolBar、Button、Image、ActionList



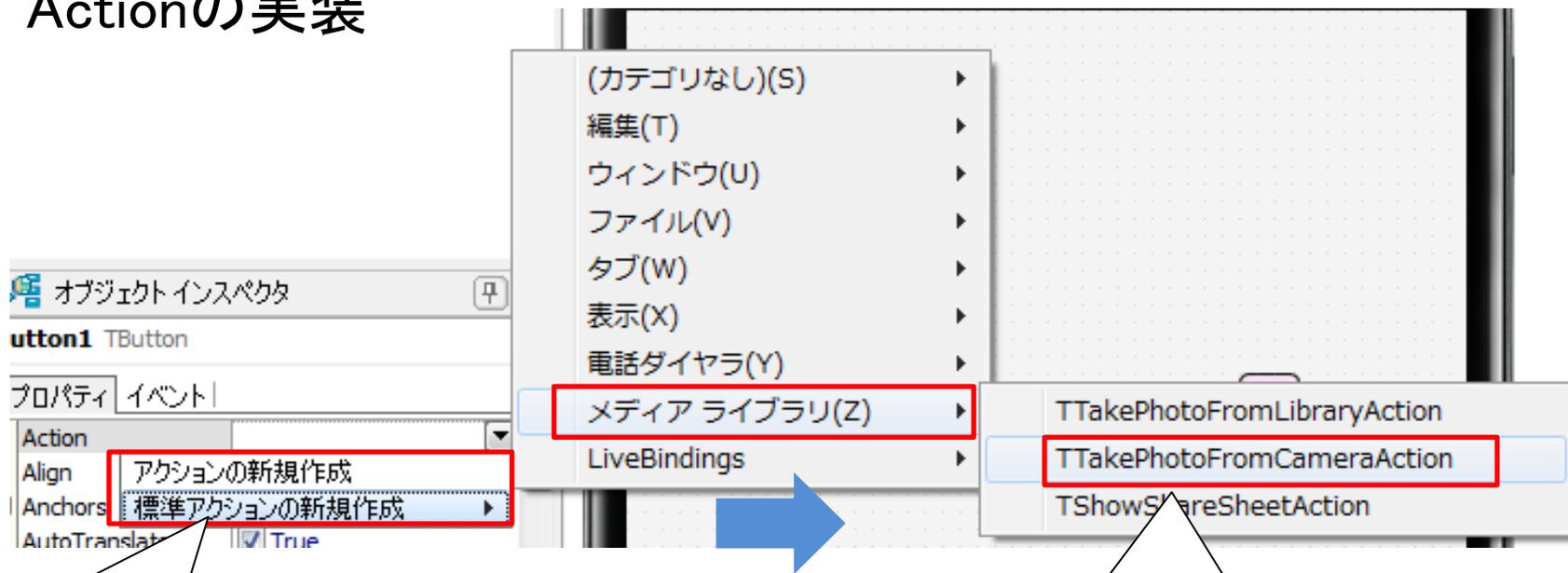
ボタンの見た目は
StyleLookupプロパティに
豊富に用意されています。



3-2. 簡単なネイティブアプリケーションの開発

• ネイティブアプリケーション開発手順5

Actionの実装



UIButton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ケーションの開発

- ネイティブアプリケーション開発手順5

Actionのイベントにプログラムを実装



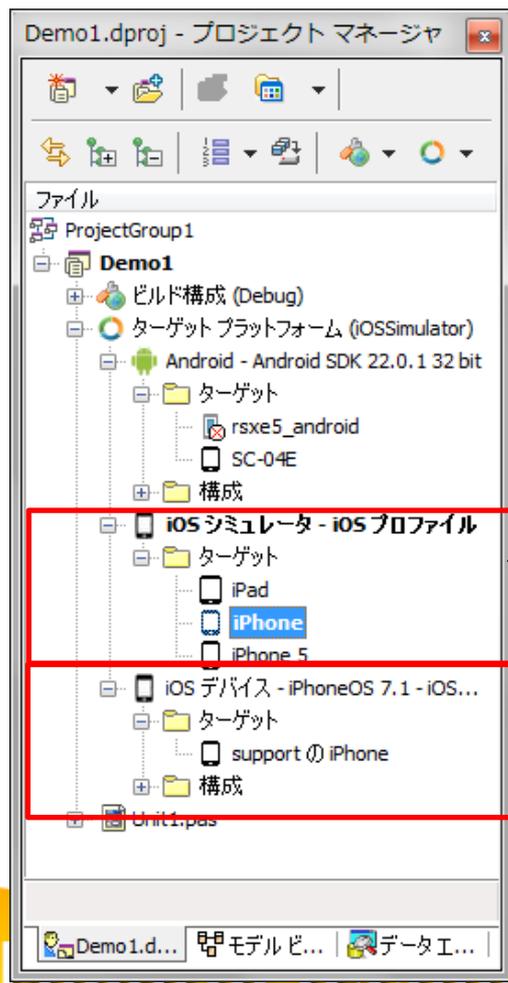
OnDidFinishTakingイベントを作成

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ケーションの開発

- ネイティブアプリケーション開発手順6
iOS実機向けにコンパイル



iOSシミュレータ向けのコンパイルも可能

iOS実機向けコンパイル

3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順7
iOSアプリケーションの実行

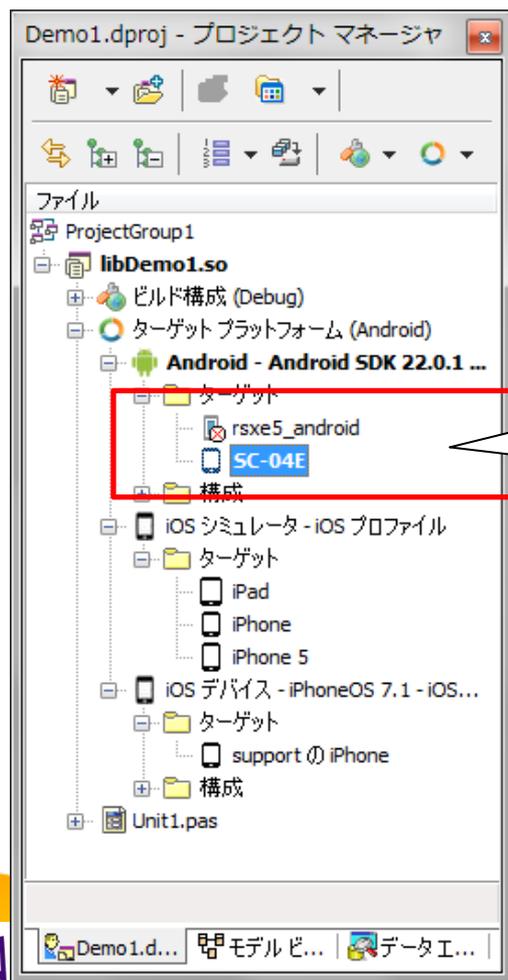


スマートデバイスの機能を使えば、カメラ撮影を連携したアプリケーションもPCやWebのアプリケーションと比べて、簡単に実現ができます。

3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順8

Android実機向けにコンパイル



同じプログラムを
Android実機向けに
コンパイル

3-2. 簡単なネイティブアプリケーションの開発

- ネイティブアプリケーション開発手順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ケーションの開発

- 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。



3-2. 簡単なネイティブアプリケーションの開発

- 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。

The image shows two screenshots from development tools. The top screenshot is from Xcode, showing the 'Release' configuration for an iOS device. A table lists files to be included in the build. The file 'alarm.mp3' is highlighted, and a callout points to the 'Remote Path' column, indicating it should be placed in the '%StartUp%Documents%' directory. The bottom screenshot is from Android Studio, showing the 'Release' configuration for an Android device. A table lists files to be included. The file 'alarm.mp3' is highlighted, and a callout points to the 'Remote Path' column, indicating it should be placed in the 'assets%internal%' directory.

ローカルパス	ローカル名	種類	プラットフォーム	リモートパス
\$(BDS)%bin%Artwork%...	FM_ApplicationIcon_57x5...	iPhone_AppIco...	[iOSDevice]	.\$
\$(BDS)%bin%Artwork%...	FM_LaunchImage_320x4...	iPhone_Launch...	[iOSDevice]	.\$
\$(BDS)%bin%Artwork%...	FM_LaunchImageLandscap...	iPad_Launch2048	[iOSDevice]	.\$
alarm.mp3		File	[Android,iOSDe...]	.\$StartUp%Documents%

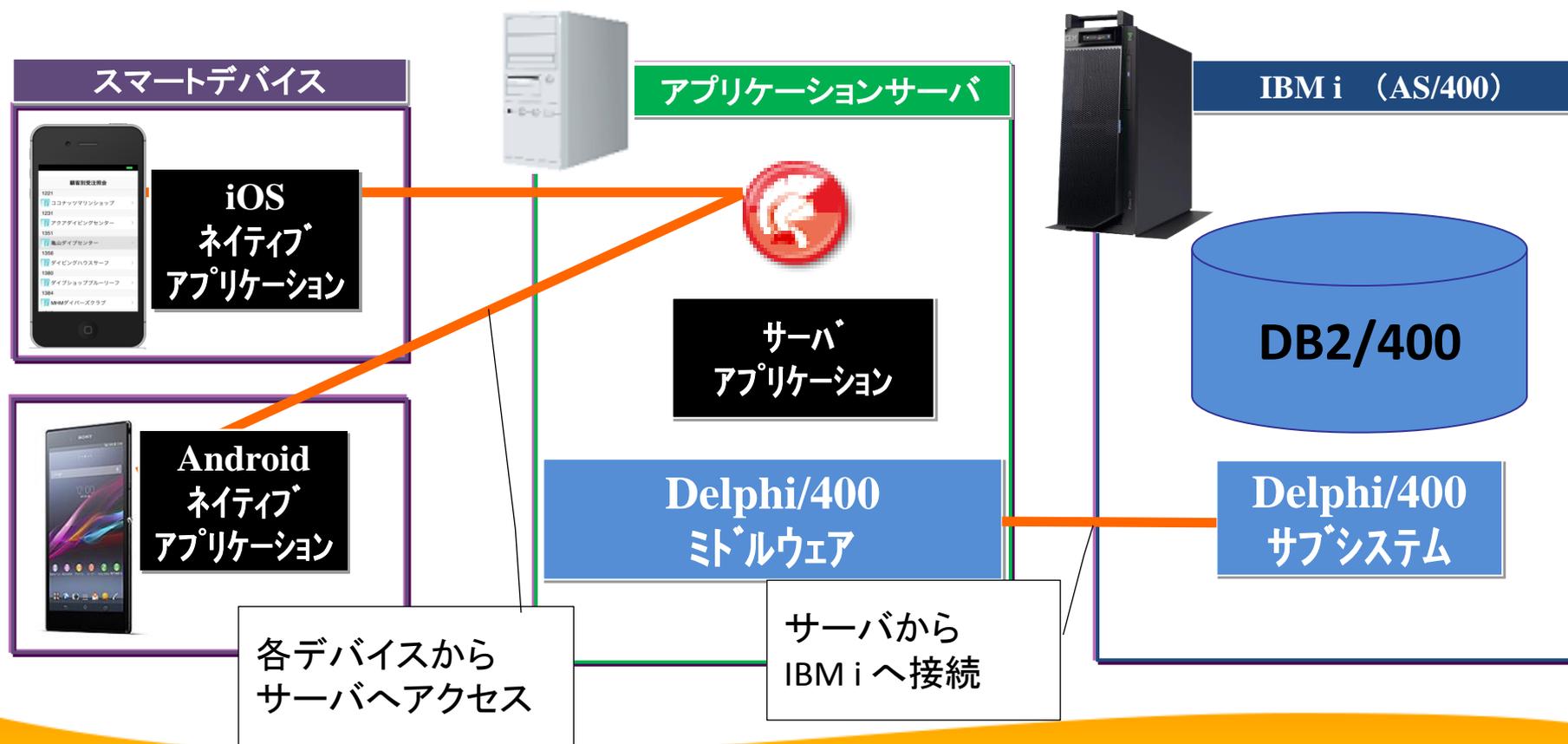
.\$StartUp%Documents%
に配置します

ローカルパス	ローカル名	種類	プラットフォーム	リモートパス
\$(BDS)%bin%Artwork%...	FM_LauncherIcon_36x36...	Android_Launc...	[Android]	res%drawable
Android%Release%	AndroidManifest.xml	ProjectAndroid...	[Android]	.\$
Android%Release%	libTimer.so	ProjectOutput	[Android]	library%lib%armeabi%
\$(BDS)%bin%Artwork%...	FM_LauncherIcon_48x48...	Android_Launc...	[Android]	res%drawable-mdpi%
\$(BDS)%bin%Artwork%...	FM_LauncherIcon_72x72...	Android_Launc...	[Android]	res%drawable-hdpi%
alarm.mp3		File	[Android,iOSDe...]	assets%internal%

assets%internal%
に配置します

3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーションの仕組み
ネイティブアプリケーションからIBM i 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. IBM i に接続するネイティブアプリケーションの開発

- 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/400で開発するサーバアプリケーション

Delphi/400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、SQLConnectionやSQLQuery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。

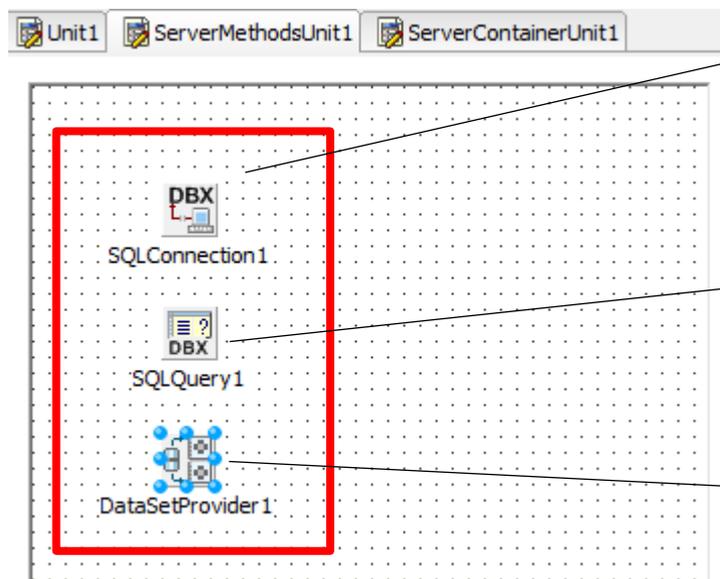
サーバアプリケーション開発手順の詳細はP61以降の補足資料記載

3-3. IBM i に接続するネイティブアプリケーションの開発

- DataSnapサーバアプリケーションの設定概要

【スマートデバイスから IBM i へ SQLを実行できるサーバ機能を実装】

SQLConnection、SQLQuery、DataSetProviderを配置して構成します。



【SQLConnectionコンポーネント】
dbExpressドライバを“CO400”に設定

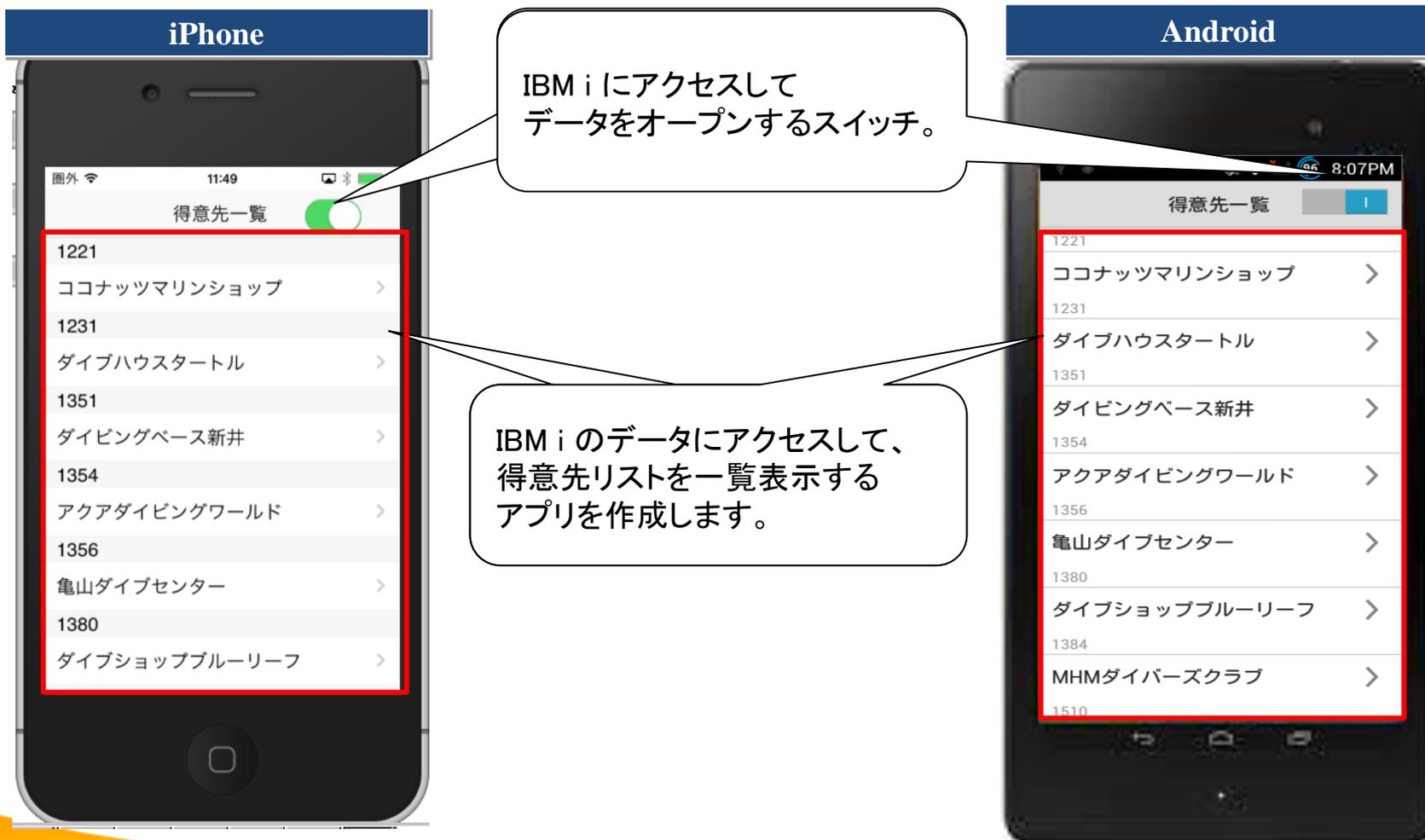
【SQLQueryコンポーネント】
接続を” SQLConnection1”に設定

【 DataSetProviderコンポーネント】
Option”poAllowCommandText”を
“True”に設定
この設定でクライアントアプリから
SQLが自由に発行できます

今回は、このプログラムをアプリケーションサーバ上で起動しておきます。

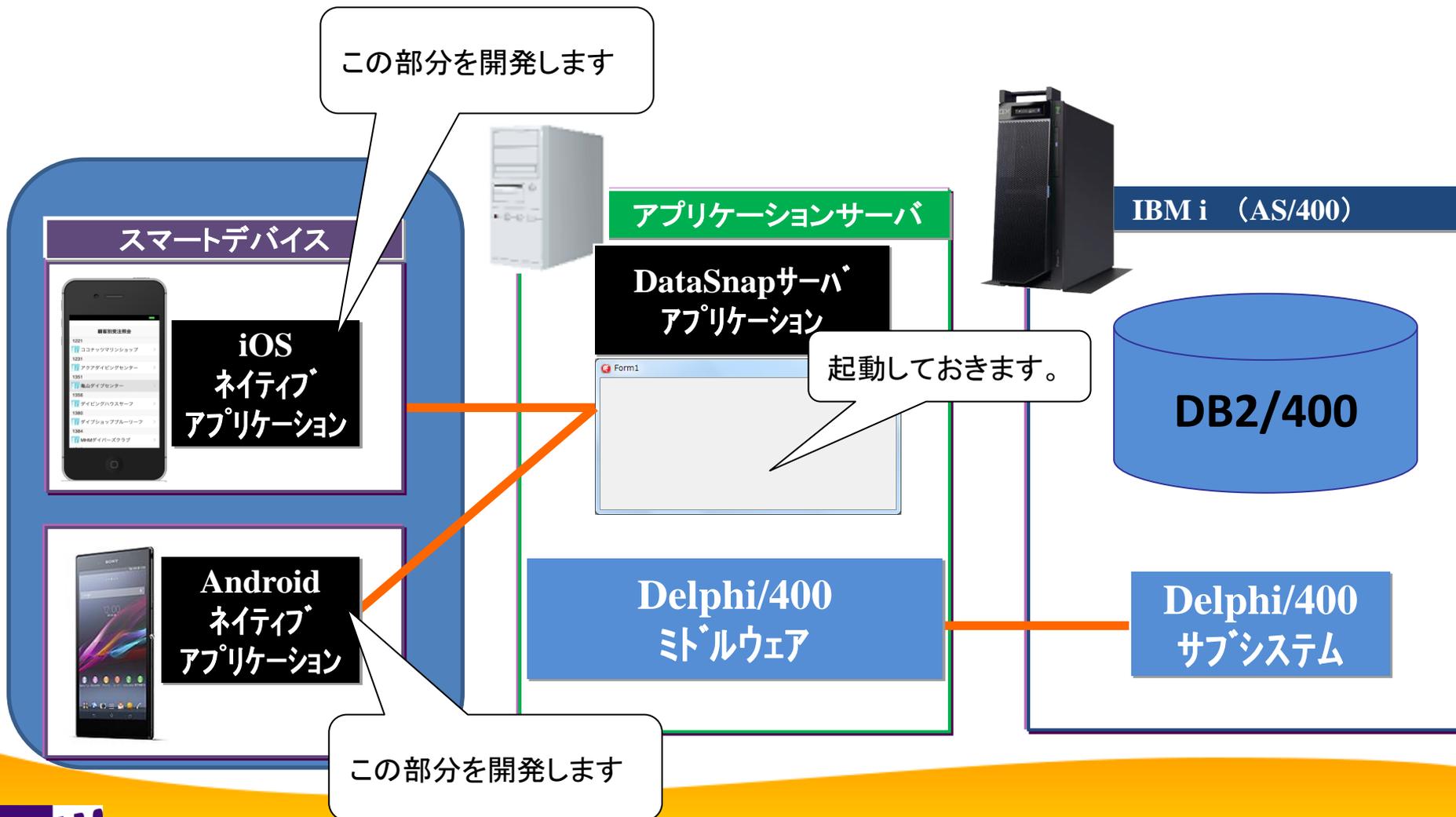
3-3. IBM i に接続するネイティブアプリケーションの開発

• デモで開発するネイティブアプリケーション



3-3. IBM i に接続するネイティブアプリケーションの開発

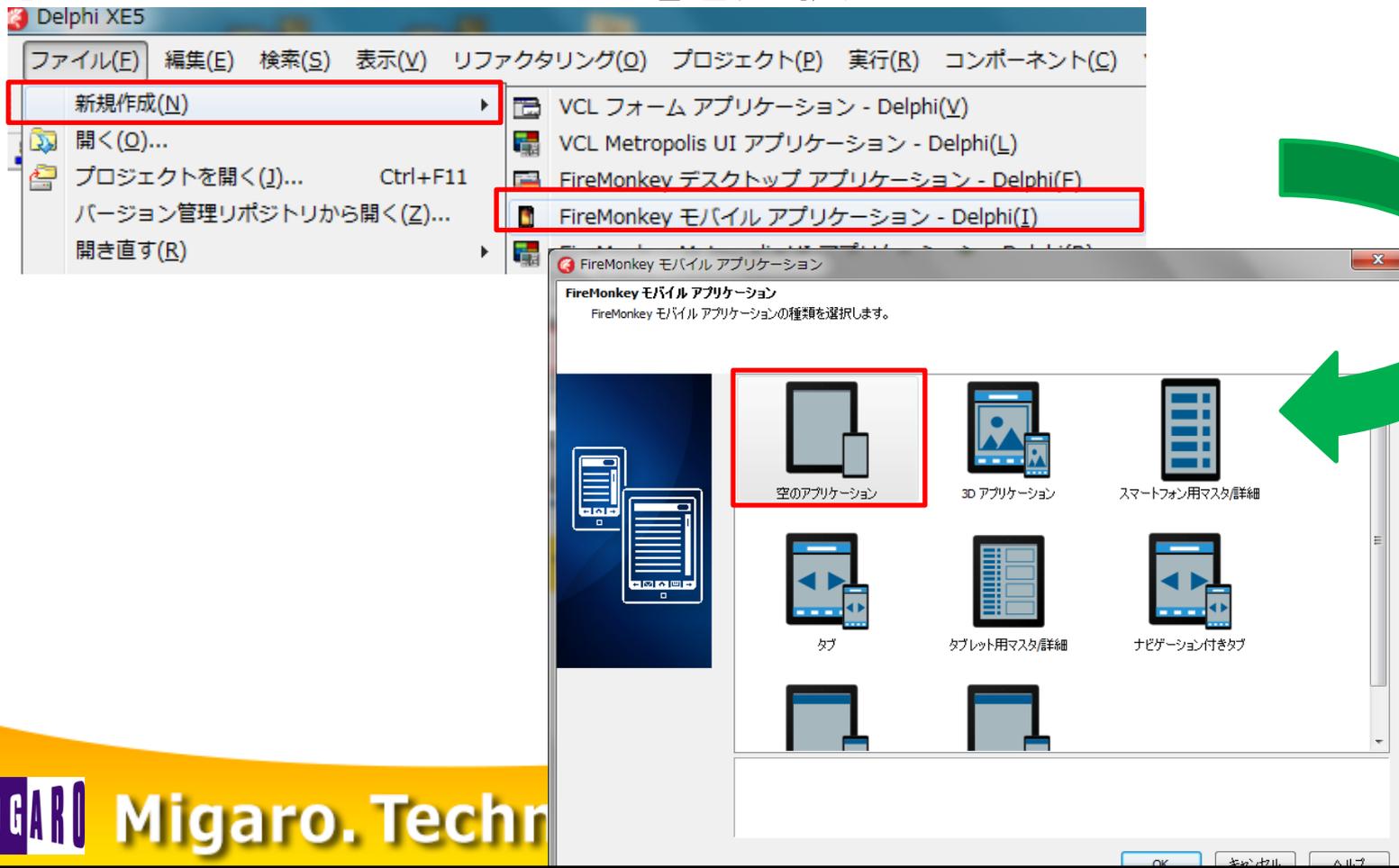
- IBM i に接続するネイティブアプリケーション



3-3. IBM i に接続するネイティブアプリケーションの開発

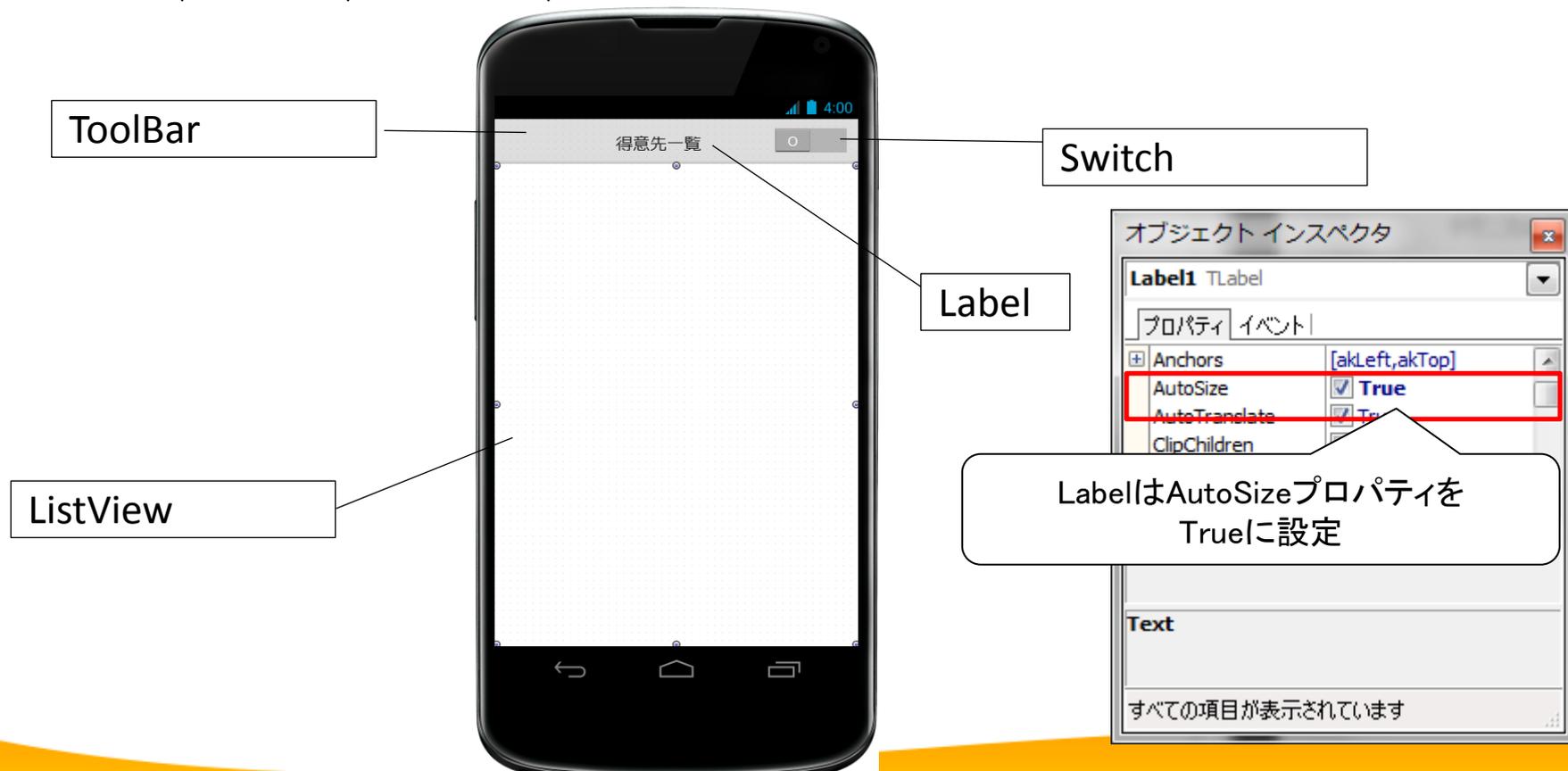
• IBM i に接続するネイティブアプリケーション開発手順1

メニューの[ファイル|新規作成]から『モバイルアプリケーション』を選択



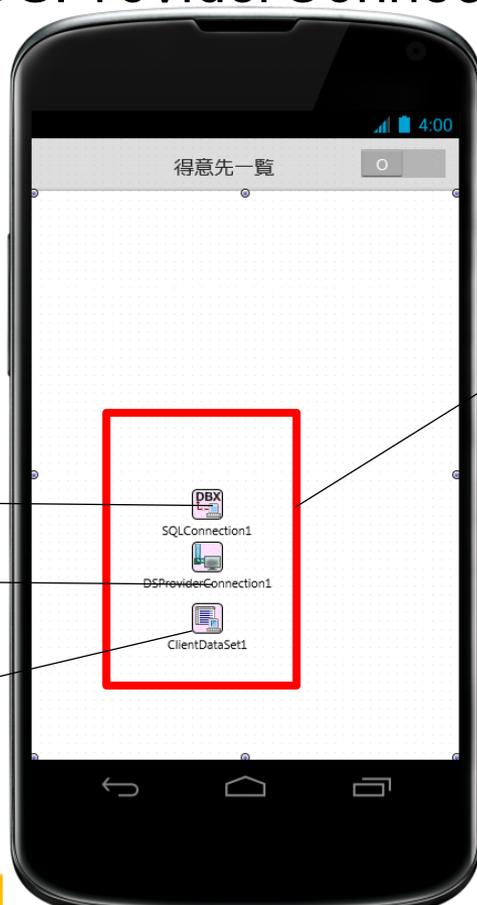
3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順2
フォームに次のコンポーネントを配置
ToolBar、Label、Switch、ListView



3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順3
フォームに次のコンポーネントを配置
SQLConnection、DSProviderConnection、ClientDataSet



SQLConnection

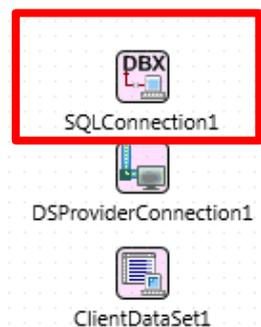
DSProviderConnection

ClientDataSet

今回はサーバアプリケーションを利用して、IBM i のデータを表示する機能を実装します

3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順4
SQLConnectionコンポーネントの設定



オブジェクト インспекタ

SQLConnection1 TSQLConnection

プロパティ イベント

Connected False

ConnectionName* DataSnapCONNECTION

Driver DataSnap

KeepConnection True

LiveBinding デザイン LiveBinding デザイン

LoadParamsOnConnect False

LoginPrompt False

Params (TStrings)

TableScope [*Table, tableView]

Tag 0

接続パラメータの保存
接続パラメータの再読み込み
DataSnap クライアントクラスの生成

表示されています

ConnectionNameプロパティを
“DATASNAPCONNECTION”に設定

値リストの編集

キー	値
DriverName	DataSnap
HostName	999.999.999.999
port	211

Datasnap
サーバ

使用するポート
デフォルト:211

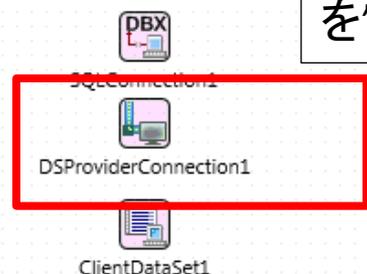
エディタ(O)... OK(O) キャンセル ヘルプ

LoginPromptプロパティ
を“False”に設定

Paramsプロパティ
を設定

3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順5
DSProviderConnectionコンポーネントの設定



ServerClassNameプロパティ
を” TServerMethods1”に入力設定



SQLConnectionプロパティ
を” SQLConnection1”に設定



3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順6
ClientDataSetコンポーネントの設定

② RemoteServerを設定していると
ProviderNameプロパティが選択できるの
で"DataSetProvider1"を設定
※サーバアプリケーションが起動している必要があります。

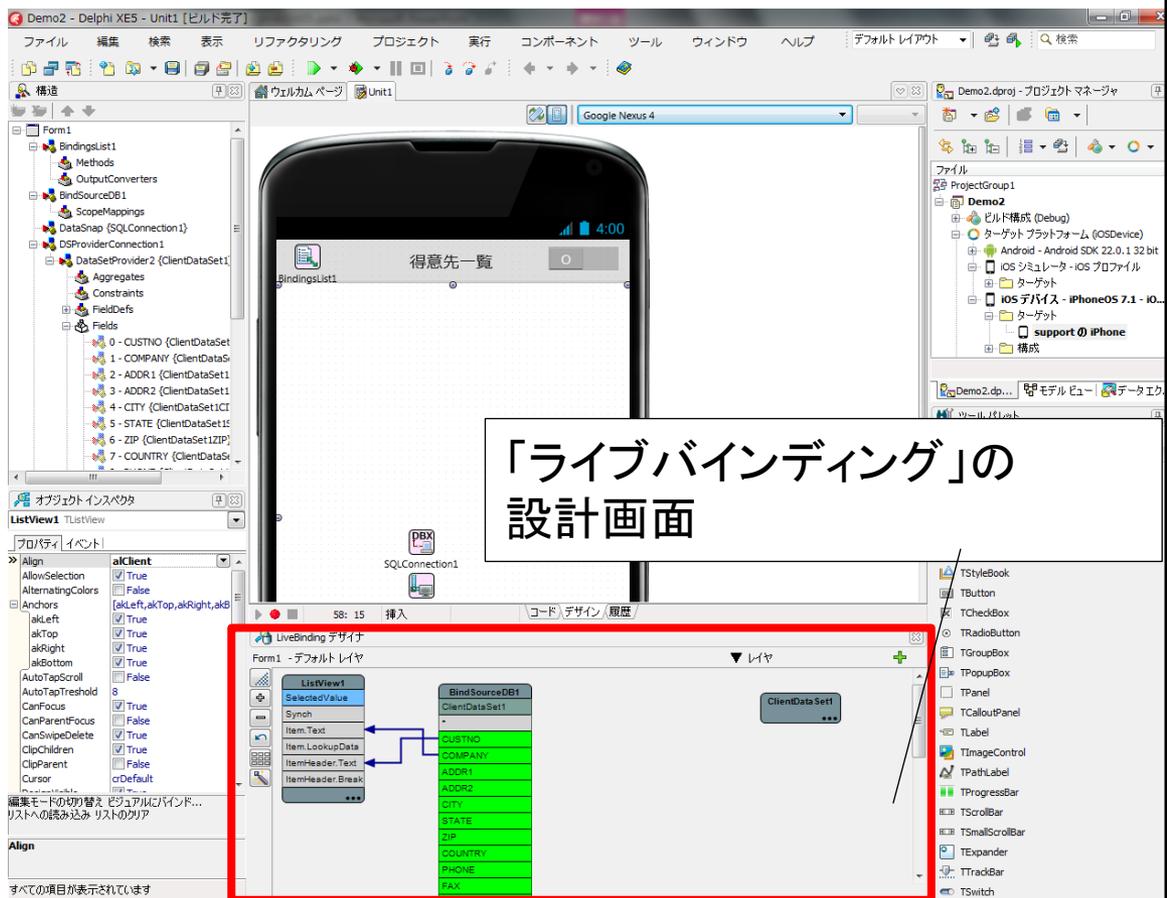
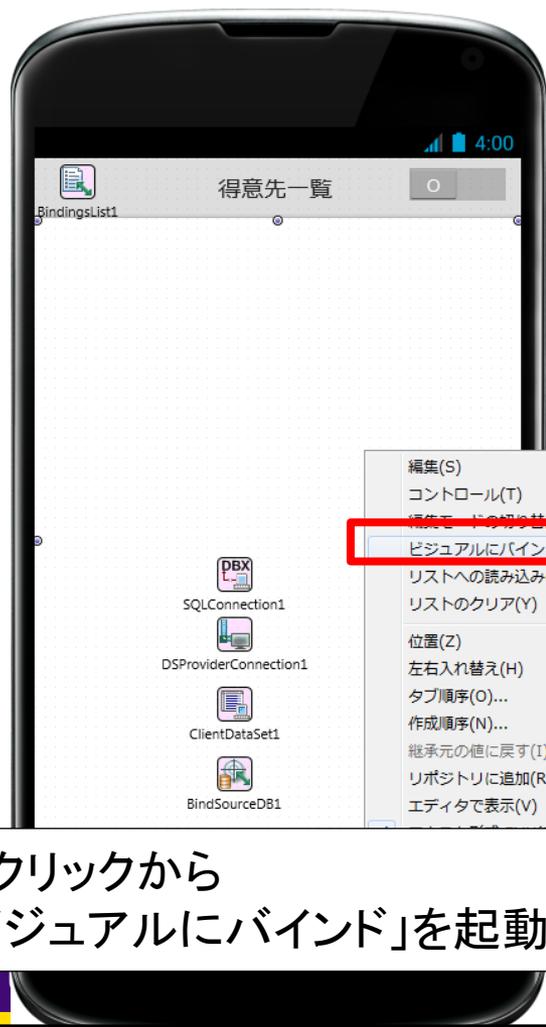
CommandTextプロパティ
に実行するSQLを設定
例) SELECT * FROM CUSTOMER

① RemoteServerプロパティに
"DSProviderConnection1"を設定

Property	Value
CommandText	SELECT * FROM CUSTOMER
RemoteServer	DSProviderConnection1
ProviderName	DataSetProvider1

3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順7
データ表示をライブバインディング機能で実装

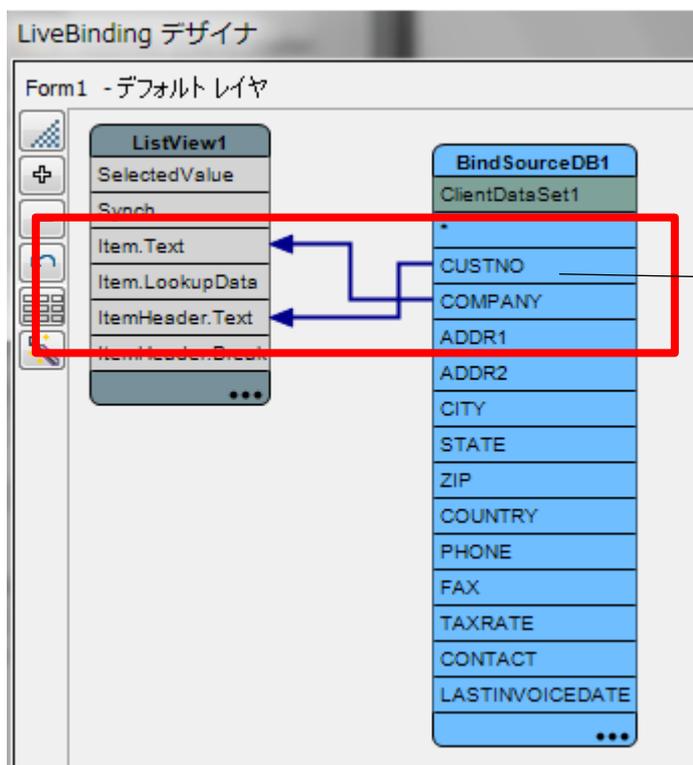


「ライブバインディング」の
設計画面

右クリックから
「ビジュアルにバインド」を起動

3-3. IBM i に接続するネイティブアプリケーションの開発

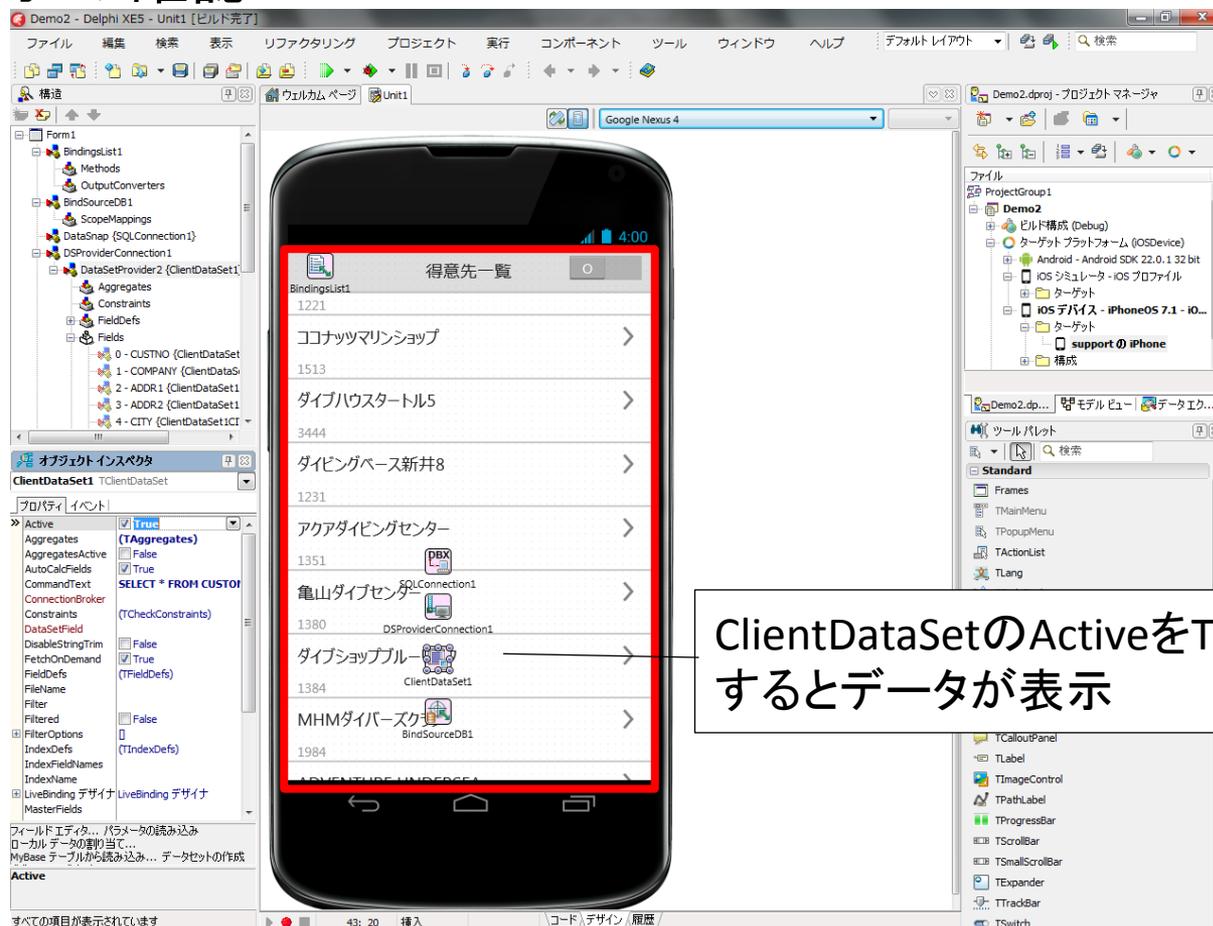
- IBM i に接続するネイティブアプリケーション開発手順8
ClientDataSetの項目をListViewにドラッグ&ドロップでリンク



CUSTNOをItemHeaderTextにリンク
COMPANYをItemTextにリンク

3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順9
データ表示の確認



3-3. IBM i に接続するネイティブアプリケーションの開発

- IBM i に接続するネイティブアプリケーション開発手順10
SwitchのonSwitchイベントにプログラムを実装

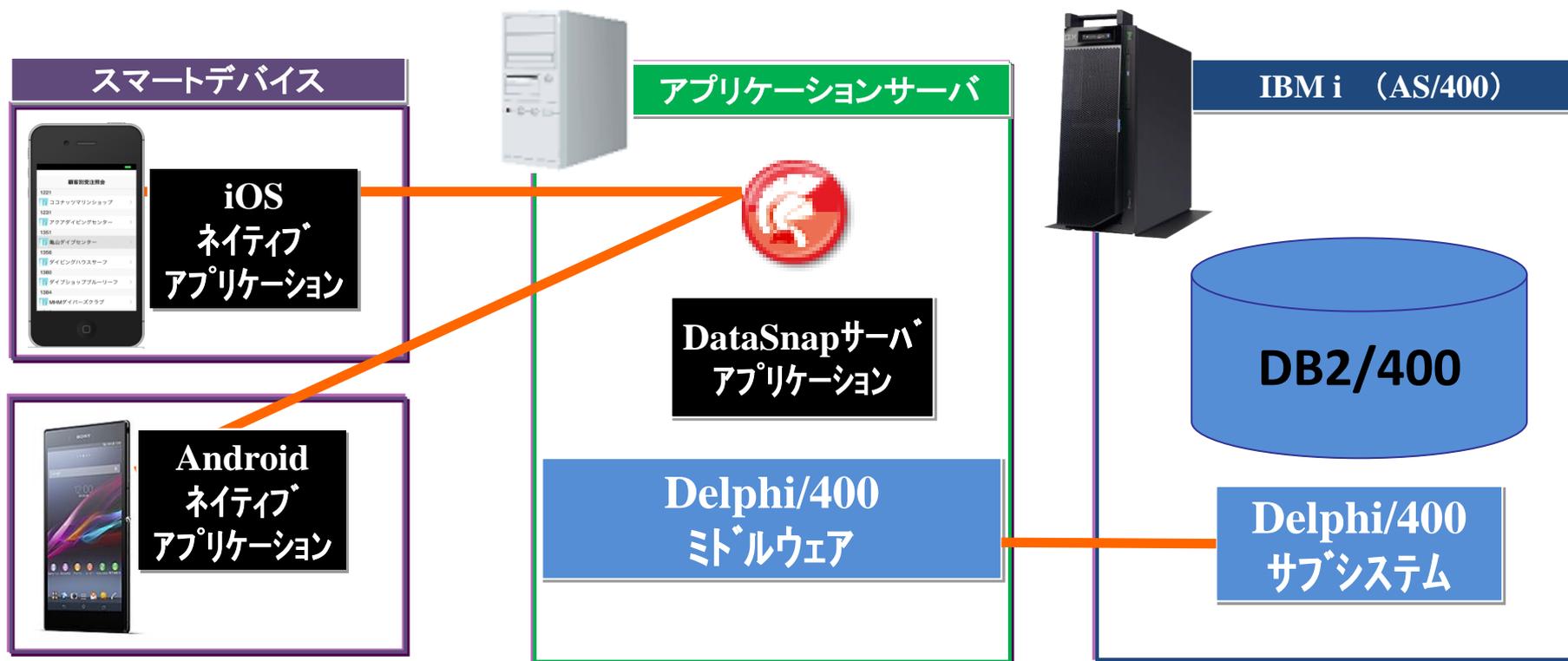


スイッチ切り替え処理

```
procedure TForm1.Switch1Switch(Sender: TObject);  
begin  
  ClientDataSet1.Active := Switch1.IsChecked;  
end;
```

3-3. IBM i に接続するネイティブアプリケーションの開発

- 3階層によってスマートデバイスからIBM i へ接続完成



本資料では設定内容を全て記載していますが、
開発時はプロパティ設定をしていくだけなので簡単！

3-3. IBM i に接続するネイティブアプリケーションの開発

- 補足: カスタマイズ例

選択先の受注明細画面を作成すれば、得意先別受注照会のようなカスタマイズも簡単に実現できます。



複数ページの切り替えはTabControlを使って設計可能



StringGridへライブバインディング

3-4. ネイティブアプリケーションの配布

- ネイティブアプリケーション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サーバ上にネイティブアプリケーション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AppleStore、AndroidであればPlayStoreにネイティブアプリケーション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

社内公開での配布

開発

Webサーバ公開



iOSアプリ開発
(社内公開)

iOSの場合、
iOS Developer Program
の種類によって数が制限



Webサーバ



Androidアプリ開発
(社内公開)

ユーザー利用



iOSユーザー



Androidユーザー

一般公開での配布

ストア公開

開発



AppleStore



iOSアプリ開発
(一般公開)



PlayStore



Androidアプリ開発
(一般公開)

各ストア公開には
審査があります

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メリット/デメリット

	社内公開	一般公開
メリット	<ul style="list-style-type: none">・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。	<ul style="list-style-type: none">・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	<ul style="list-style-type: none">・Webサーバ等を用意して、配布環境の構築・運用が必要。	<ul style="list-style-type: none">・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)

社内公開するアプリは、配布用の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します。

【配布用のコンパイルファイル】

iOS: ipa、ipListファイル

Android: apkファイル

4.まとめ

4.まとめ

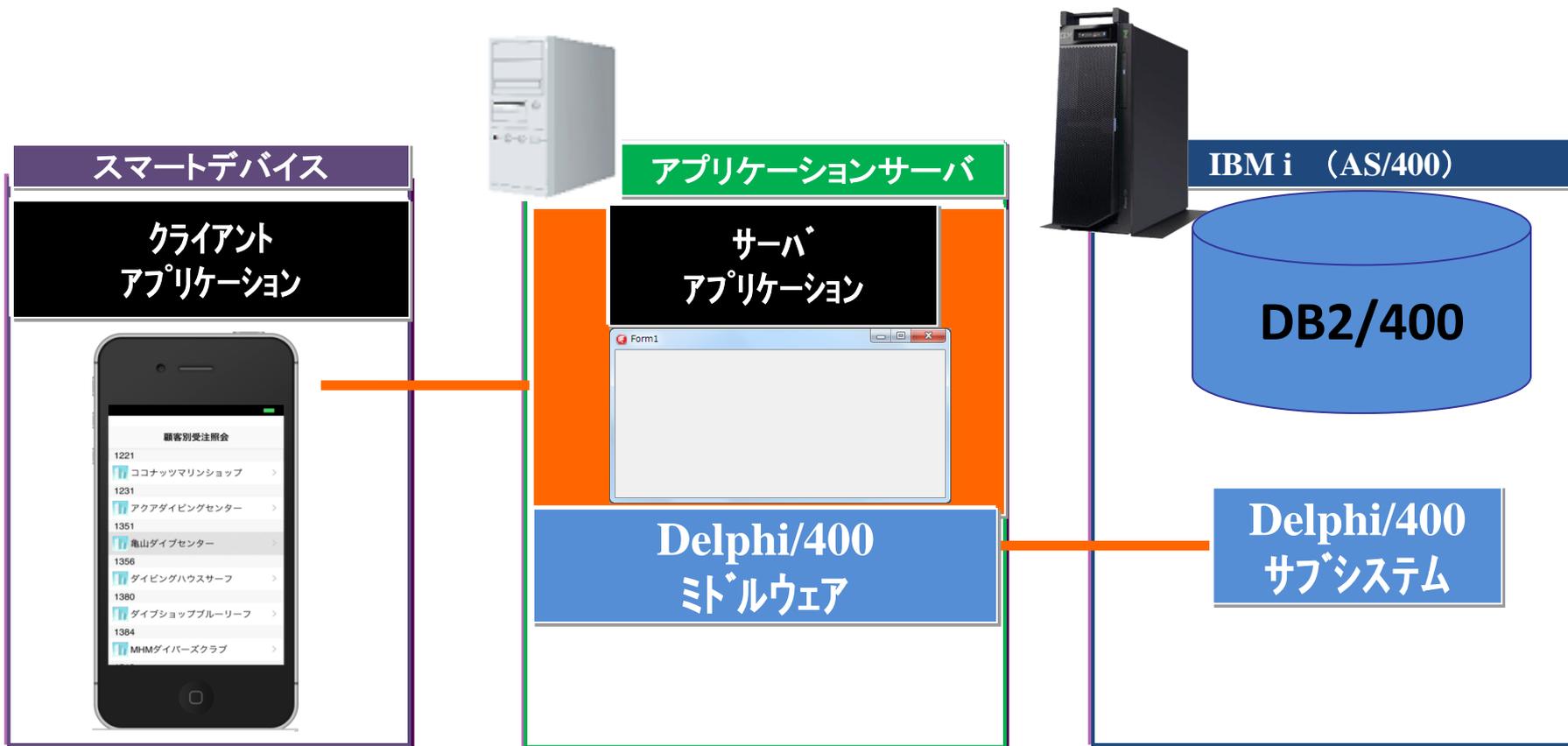
- 企業導入されるスマートデバイスはiOS、Androidが主流
- Delphi/400XE5ではiOSとAndroidのネイティブ開発が可能
- IBM i への接続はDataSnapサーバを使った3階層方式
- 配布は社内公開と一般公開で方法が異なる

ご静聴ありがとうございました

補足資料

補足資料

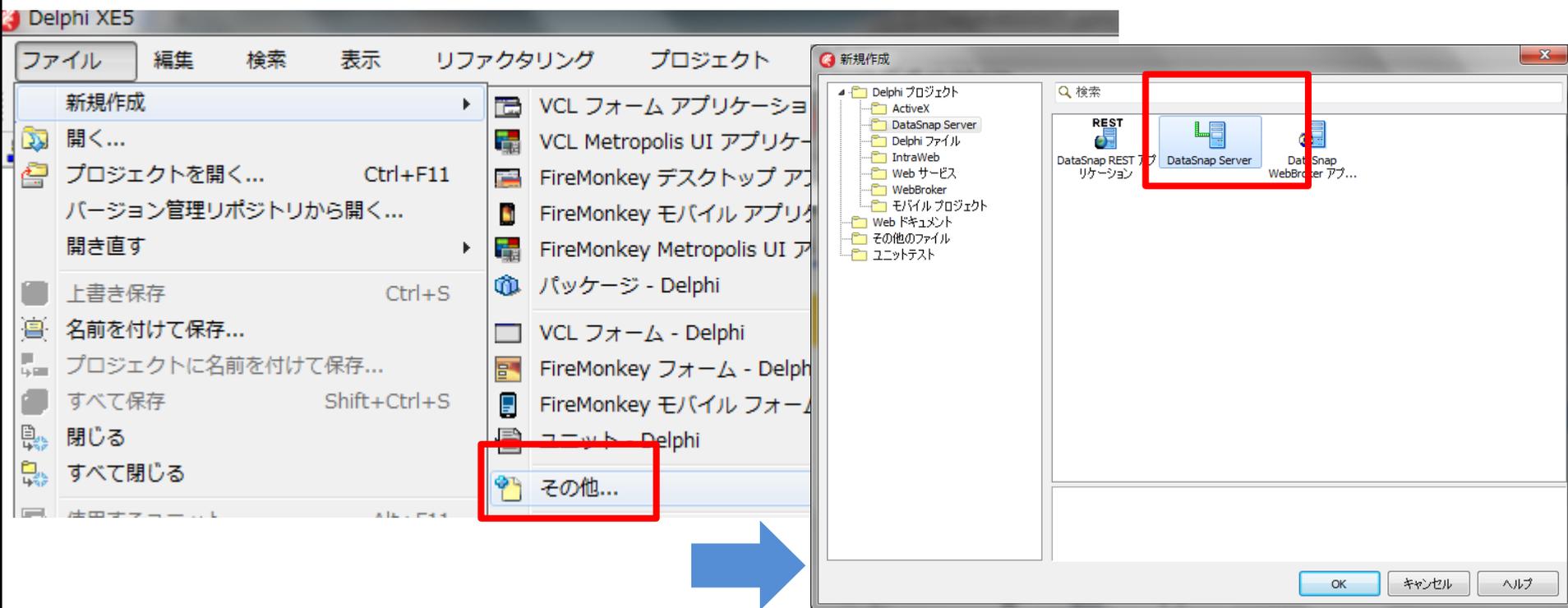
- サーバアプリケーションの開発手順



補足資料

• サーバアプリケーションの開発手順1

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



補足資料

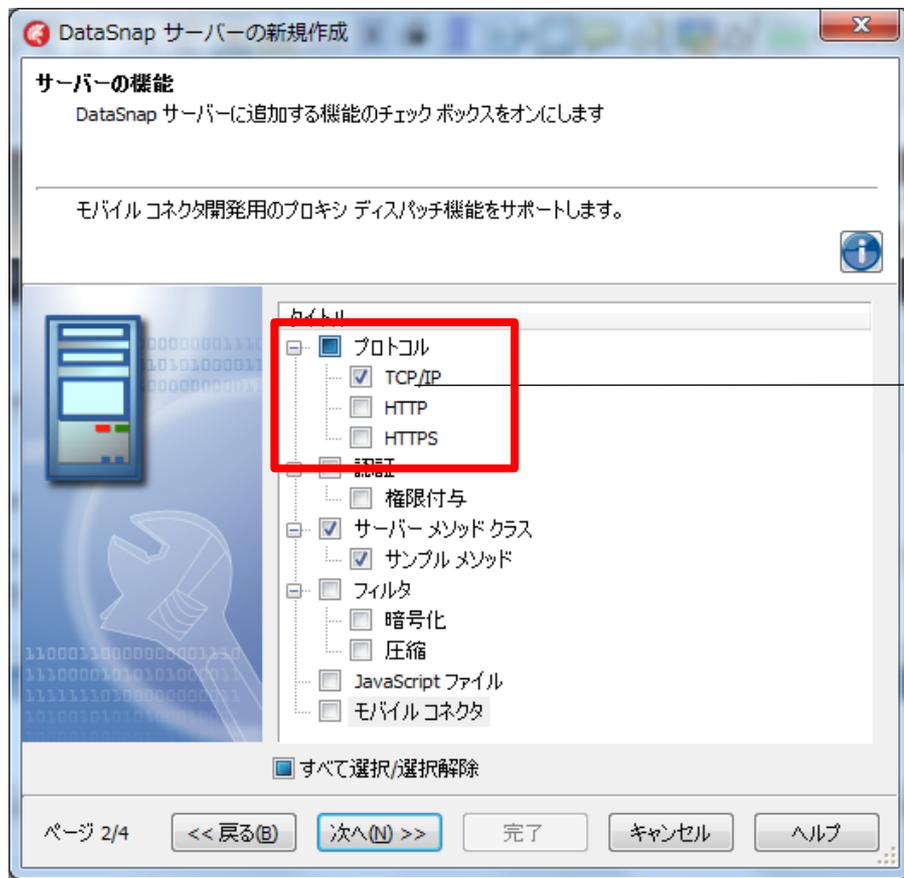
- サーバアプリケーションの開発手順2
DataSnapServerの形式を選択



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。

補足資料

- サーバアプリケーションの開発手順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。

補足資料

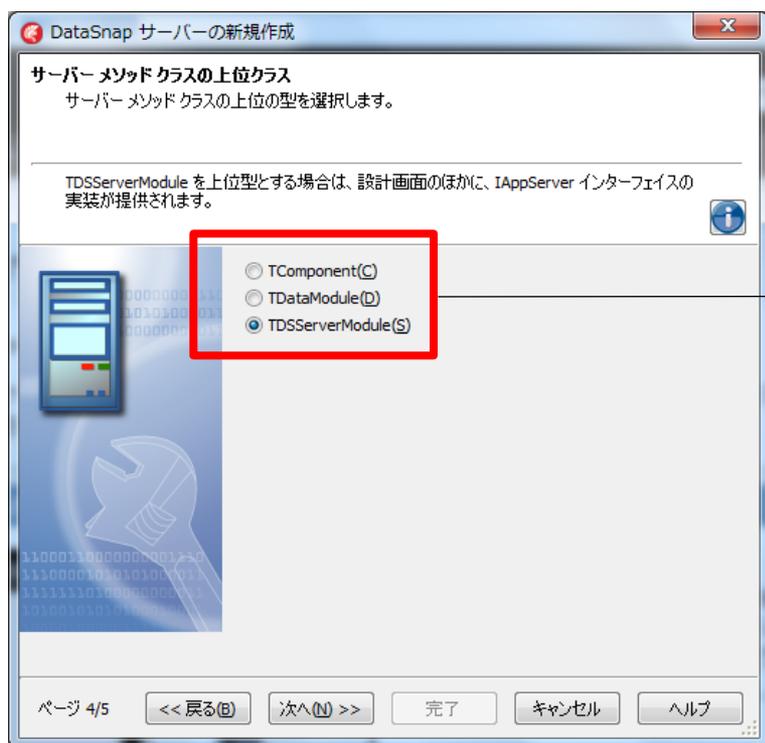
- サーバアプリケーションの開発手順4
使用するポート番号を指定



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)

補足資料

- サーバアプリケーションの開発手順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。

補足資料

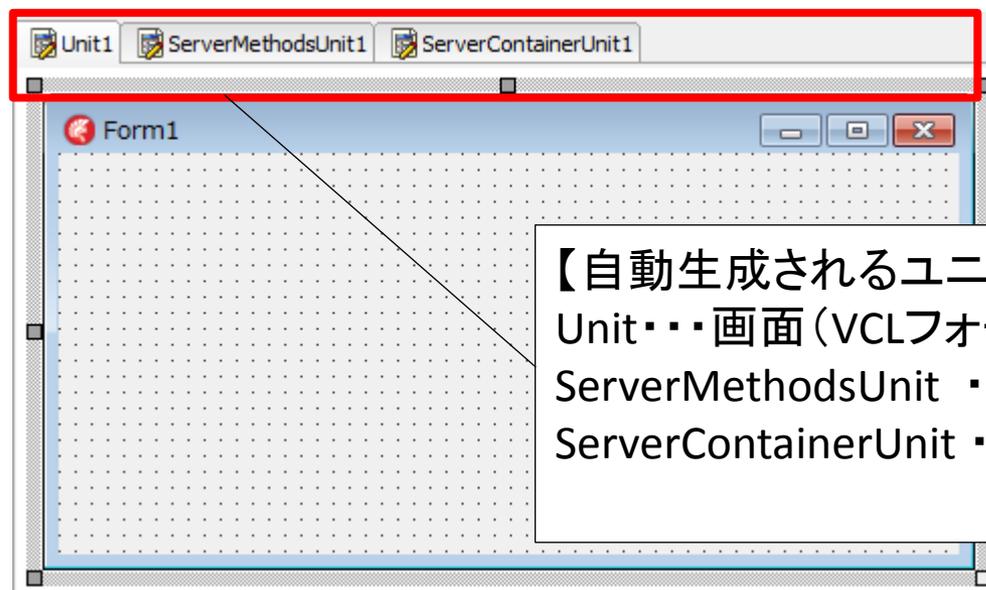
- サーバアプリケーションの開発手順6
ソースの保存先を選択



プロジェクトソースの保存場所を
指定します。

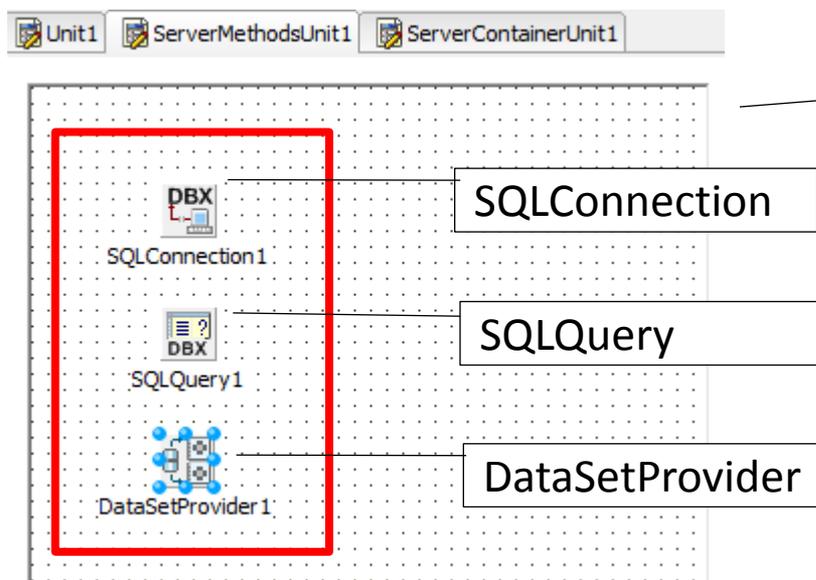
補足資料

- サーバアプリケーションの開発手順7
選択した構成によってモジュールが生成



補足資料

- サーバアプリケーションの開発手順8
ServerMethodsに次のコンポーネントを配置
SQLConnection、SQLQuery、DataSetProvider



今回はIBM i 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。

補足資料

- サーバアプリケーションの開発手順9
SQLConnectionコンポーネントの設定

オブジェクト インспекタ

SQLConnection1 TSQLConnection

プロパティ イベント

Connected False

» ConnectionName **CO400CONNECTION**

Driver **CO400**

GetDriverFunc **getSQLDriverCO400**

KeepConnection True

LibraryName **dbco430.dll**

LoadParamsOnConnect False

LoginPrompt True

Name SQLConnection1

Params **(TStrings)**

TableScope [tsTable,tsview]

Tag 0

VendorLib **co400loc.dll**

接続パラメータの保存 接続パラメータの再ロード

ConnectionName

すべての項目が表示されています

値リストの編集

キー	値
Multiple Transaction	False
RoleName	MGTEC14LIB
HostName	MIGARO15
User_Name	D400
Password	D400
Decimal Separator	N
IsolationLevel	ReadCommitted
LocaleCode	0000
DriverName	CO400
ErrorResourcefile	
SubSite	1
Database	MIGARO15

ライブラリ名

接続名

ユーザーパスワード

接続名

Paramsプロパティを設定

ConnectionNameプロパティを“CO400CONNECTION”に設定

補足資料

- サーバアプリケーションの開発手順10
SQLQueryコンポーネントの設定

The image displays a development environment window with a component tree on the left and an Object Inspector on the right. The component tree shows a hierarchy: Unit1, ServerMethodsUnit1, and ServerContainerUnit1. Under ServerContainerUnit1, there are three components: SQLConnection1, SQLQuery1 (highlighted with a red box), and DataSetProvider1. The Object Inspector shows the properties for SQLQuery1 (TSQLQuery). The SQLConnection property is set to SQLConnection1, which is also highlighted with a red box. A callout box with a white background and black border points to the SQLConnection property, containing the text: "SQLConnectionプロパティを\"SQLConnection1\"に設定".

補足資料

- サーバアプリケーションの開発手順11
DataSetProviderコンポーネントの設定

DataSetプロパティを"SQLQuery1"に設定

Optionプロパティの"poAllowCommandText"を"True"に設定

オブジェクト インспекタ

DataSetProvider1 TDataSetProvider

プロパティ	イベント
Constraints	<input checked="" type="checkbox"/> True
DataSet	SQLQuery1
Exported	<input checked="" type="checkbox"/> True
LiveBinding デザイン	LiveBinding デザイン
Name	DataSetProvider1
Options	[poAllowCommandTe
poFetchBlobsOn	<input type="checkbox"/> False
poFetchDetailsO	<input type="checkbox"/> False
poIncFieldProps	<input type="checkbox"/> False
poCascadeDelet	<input type="checkbox"/> False
poCascadeUpda	<input type="checkbox"/> False
poReadOnly	<input type="checkbox"/> False
poAllowMultiRec	<input type="checkbox"/> False
poDisableInserts	<input type="checkbox"/> False
poDisableEdits	<input type="checkbox"/> False
poDisableDelete	<input type="checkbox"/> False
poNoReset	<input type="checkbox"/> False
poAutoRefresh	<input type="checkbox"/> False
poPropagateCh	<input type="checkbox"/> False
>> poAllowComman	<input checked="" type="checkbox"/> True
poRetainServerC	<input type="checkbox"/> False
poAllowCommandText	

すべての項目が表示されています

補足資料

- サーバアプリケーションの開発手順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

