吉原 泰介

株式会社ミガロ.

RAD事業部 技術支援課

[Delphi/400] iOS/Androidネイティブアプリケーション入門 ―マルチデバイス開発手法から社内配布

略歴
1978年3月26日生
2001年　龍谷大学 法学部卒
2005年7月 株式会社ミガロ. 入社
2005年7月 システム事業部配属
2007年4月 RAD事業部配属

現在の仕事内容
Delphi/400やJC/400の製品試験および月100件に及ぶ問い合わせサポートやセミナー講師などを担当している。

1.はじめに

この数年でスマートデバイスの導入が個人だけではなく、企業でも急速に進んできている。実際に2014年に総務省が行ったアンケート調査では、国内企業のスマートデバイス導入率は28.4%と推計されており、3割近い企業でスマートデバイスが使用されていることになる（総務省・経済産業省「平成24年経済センサス－活動調査」）。

2011年の同調査での企業導入率が10%未満だったことを考えると、飛躍的に導入数が伸びていることがわかる。

こうしたスマートデバイスを導入した企業の多くが「業務効率化」を目的としている。これはメールやWebの利用だけではなく、社内システムの活用が強く期待されていることである。

Delphi/400のテクニカルサポートにもスマートデバイスのお問い合わせをいただくことが多くなってきており、これからの社内システムにはスマートデバイスアプリケーションが必要とされる機会が増えてくると実感している。

では、実際にスマートデバイスを導入している企業では、どういった機種が使用されているのかというと、その大半がiOS（iPhone、iPad）やAndroidとなっている。それ以外のWindows Phoneなどの機種は、まだ企業導入されていないのが現状である。

つまり、これから企業のスマートデバイスでアプリケーションを活用するには、iOS、Android機種へ、いかに対応できるかが重要となってくる。

こうした大きな環境変化の背景もあり、従来のクライアント/サーバー（以下、C/S）、Web型開発に加えて、iOS、Androidのネイティブアプリケーション開発にも対応したDelphi/400 Version XE5の新機能を紹介したいと考えた。

本稿では、Delphi/400を使ったスマートデバイス向けのネイティブアプリケーション開発について、開発環境や開発手順、配布・運用のポイントを説明していきたい。

2.スマートデバイスアプリケーションの種類

この章では、スマートデバイス上で動作するアプリケーションの種類や特徴について説明する。

スマートデバイスアプリケーションは、大きく2つの種類に分類できる。アプリケーションの種類としては、Webアプリケーションとネイティブアプリケーションがある。Delphi/400ではどちらのアプリケーションも開発できるが、それぞれのアプリケーションの特徴を把握してみる。

Webアプリケーション

Webアプリケーションは、PCでの使用と同様にWebブラウザを使って使用するアプリケーションである。Webアプリケーションは Webサーバー上で動作しており、スマートデバイス端末ではWebブラウザを使って利用することができる。【図1】

図1　Webアプリケーション構成

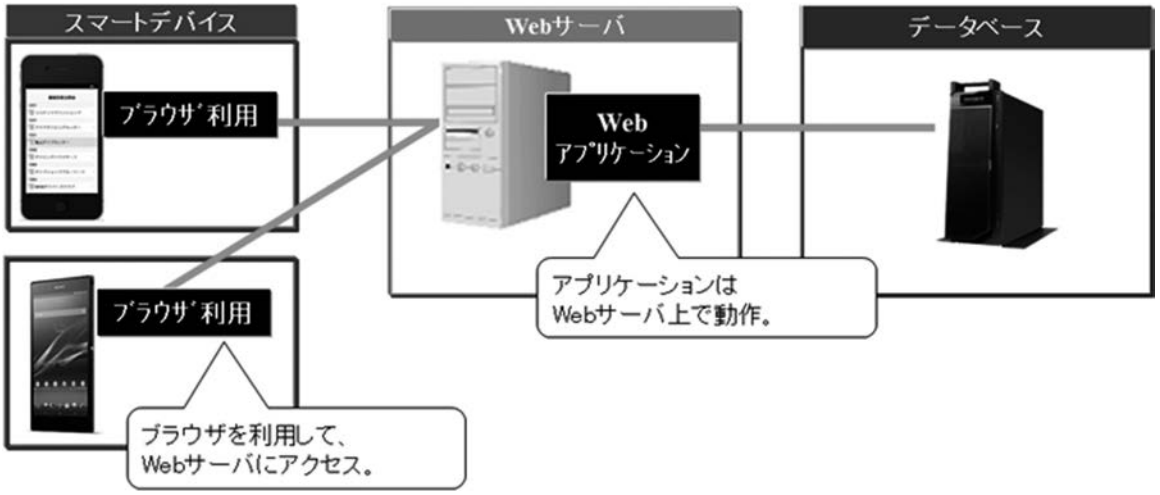


図2　ネイティブアプリケーション構成

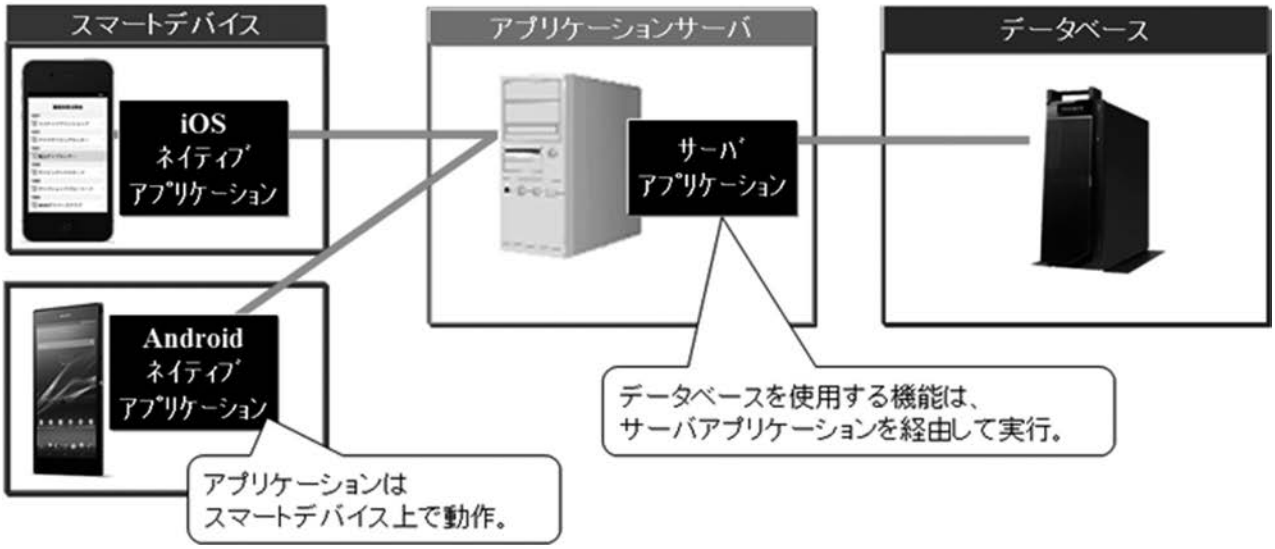


図3　ネイティブ・Webアプリケーションの特徴

	ネイティブ	Web
開発言語	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	△	◎

そのため、スマートデバイス端末にはアプリケーションがインストールされることはない。

この特徴のメリットは、「アプリケーションの配布が不要」という点と、「ブラウザに対応していれば機種の制限がない」という点が挙げられる。逆にデメリットとしては、「スマートデバイスのネイティブ機能が十分に活用できない」「ネットワークに接続していない環境では使用できない」という点がある。

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にインストールして使用するアプリケーションである。もちろん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する。【図2】

この特徴のメリットとしては「スマートデバイスのネイティブ機能を100%活用できる」「ネットワークに接続していない環境でも使用できる」という点である。またスマートデバイス端末上で直接実行するため、アプリケーションの動作レスポンスは、一般的にWebアプリケーションより優れている。

デメリットとしては、「iOS、Androidごとに別言語の習得・開発が必要となる」という点が挙げられる。しかしDelphi/400では、iOS、Androidのネイティブアプリケーションを Delphi言語のみで開発することができる。そのため、ネイティブアプリケーションの一般的なデメリットも、Delphi/400では逆に長所となっている。【図3】

これは「マルチデバイス開発」と呼ばれる画期的な開発手法によるものである。「マルチデバイス開発」については、後述の開発環境で詳しく説明する。

それぞれアプリケーションの種類によって、得手・不得手の部分があるが、Delphi/400ではどちらのアプリケーションも開発でき、用途によって選択することができる。

例えば、機能面が優先であればネイティブアプリケーション、運用管理の軽減が優先であればWebアプリケーションといった選択も可能である。

3.ネイティブアプリケーションの開発環境

この章では、Delphi/400のネイティブアプリケーション開発環境を詳しく説明していく。

iOS/Androidのネイティブアプリケーションは Delphi/400で開発できるが、対象とするスマートデバイスによって必要となる開発環境が違ってくる。それぞれの開発で必要となる環境や設定ポイントを簡単にまとめてみた。対象とする開発環境部分をご一読いただきたい。

3-1. iOS向け開発環境

Delphi/400でiOS向けのアプリケーションを開発する場合、Windowsだけではコンパイルやアプリケーションの配布が行えないため、OSX（Mac）が必要となる。【図4】

もちろんMac上に仮想環境(Windows)を構築すれば、1台のマシンで開発環境を用意することも可能である。

必要となる環境

- Windows端末（Delphi/400 Version XE5）
- Mac端末（OSX 10.7 ～ 10.9）
- iOS Developer Program（Xcode、配布）
- iOS実機（iPhone、iPadなどiOS 6.0 ～ 7.1）

Mac環境の構築

iOSの開発環境では、Delphi/400をインストールしているWindows端末とは別にMac端末が必要になる。Mac端末はOSX 10.7Lion以降をサポートしている。

Xcodeのインストール

Mac端末にはiOS6.0以降に対応したXcodeのインストールが必要になる。XcodeはApp StoreまたはAppleのDeveloperページからダウンロードしてインストールすることができる。【図5】

iOS Developer Program

iOSアプリケーションはApple社の規約により、iOSへ配布するためには「iOS Developer Program」に加入する必要がある。「iOS Developer Program」は1年間の有償プログラムとなっており、用途別に2種類用意されている。「iOS Developer Program」と「iOS Developer Enterprise Program」である。

① iOS Developer Program
https://developer.apple.com/jp/programs/ios/

「iOS Developer Program」は、主にApp Store向けのアプリケーションを配信するためのプログラムになる。Ad Hocと呼ばれる機能で社内向けにアプリケーションも配布することはできるが、台数は上限が100台に設定されており、端末の事前登録も必要になる。

② iOS Developer Enterprise Program
https://developer.apple.com/jp/programs/ios/enterprise/

「iOS Developer Enterprise Program」は社内専用向けアプリケーションを配布するためのプログラムになる。「iOS Developer Program」と違い、社内向けに配布できる台数に制限がなく、端末の事前登録も必要ない。ただしApp Store向けにアプリケーションを配信することはできない。

この2種類のプログラムは用途別に用意されているが、社内用アプリケーションを開発・運用する場合には、「iOS Developer Enterprise Program」のプログラムが目的に合っている。注意点としては、プログラムの購入から手続きの完了までApple社の処理に数日かかるため、開発前に事前に設定しておく必要がある（本稿執筆時2014年8月時点の情報。今後Apple社によって変更されることもあるため、最新情報はApple社サイトなどでご確認いただきたい）。

実機の登録

iOSアプリケーションの開発は、Mac上のiOSシミュレータでも実行できるが、実機のiPhoneやiPadとは違う部分も多い。そのため、実際の開発では実機でのテストが必須といえる。

実機をテストで使用するiOSは、Mac上でキーチェーンアクセスを行って登録しておく必要がある。キーチェーンアクセスはMac上の「アプリケーショ

図4　iOSアプリケーション開発環境

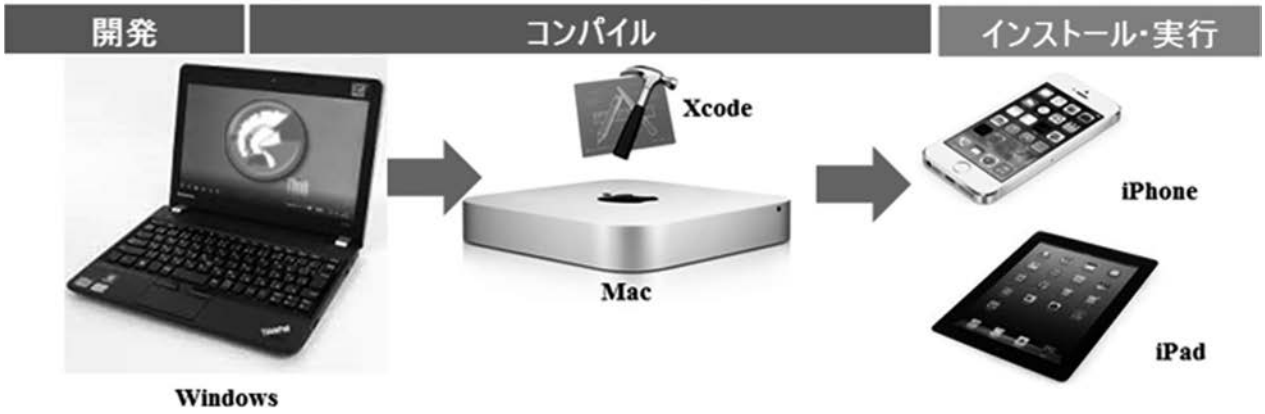


図5　Xcodeの入手

図6　Mac用　PA Server

ン | ユーティリティ」メニューから作業できるので、Apple 社のマニュアルを参考に登録作業を行う。

PAServer のインストール

Delphi/400 の Windows 開発 PC から、コンパイルしたアプリケーションを転送したり、デバッグを行うための PAServer（Platform Assistant Server）を Mac にインストールする。Delphi/400 Version XE5 では、開発 PC の下記フォルダに Mac 向けのインストーラが用意されている。

フォルダパス：
C:\Program Files\Embarcadero\RAD Studio\12.0\PAServer

このフォルダにある「RADPAServer XE5.pkg」を Mac 端末にコピーして、Mac 上でダブルクリックするとインストールが行える。【図 6】

インストールが完了すると、メニューの「アプリケーション」に「RAD PAServer XE5」として登録されているので、PAServer をダブルクリックして起動する。【図 7】

PAServer が起動するとコンソール画面で「Enter キーを押す」と表示されるので、[Enter] キーを押してサービスの開始が完了する【図 8】（iOS7.1 使用の場合は、Hotfix_6（29795）を適用する必要がある）。

接続プロファイルの作成

Delphi/400 開発 PC から、Mac に接続する設定を作成する。Delphi 開発画面の［ツール¦オプション］からオプション画面を開き、「接続プロファイルマネージャ」を選択する。【図 9】

追加ボタンを押すと【図 10】のようなダイアログが表示されるのでプロファイル名を設定して「次へ」を押す。プロファイル名は任意で設定できるので、分かりやすい名前（Mac など）にしておくと使用時に便利である。

次に表示される設定画面で Mac 端末の接続情報を設定し、接続テストが成功すれば完了である。接続テストには「接続テスト」ボタンが用意されている。【図 11】

SDK の取得

Delphi/400 上で対象のデバイス OS に合わせた開発を行うために、SDK の取り込みが必要になる。接続プロファイル同様に Delphi 開発画面の［ツール | オプション］からオプション画面を開き、「SDK マネージャ」を選択する。【図 12】

追加ボタンを押すと【図 13】のダイアログ画面が表示される。プラットフォームに「iOS デバイス」を選択して、接続するプロファイルには作成済のプロファイルを選択して設定する。最後に接続先から対象の SDK バージョンが自動表示されるので、選択して「OK」ボタンを押下する。これだけで、自動的に SDK がダウンロードされて組み込みが完了となる。

3-2. Android向け開発環境

Delphi/400 で Android 向けのアプリケーションを開発する場合、Windows 内に全ての開発環境を構築することができる。【図 14】

必要となる環境

- Windows 端末（Delphi/400 Version XE5）
- Android 実機
  （Android 2.3.3 以降の ARM7 + NEON 対応デバイス）

開発環境の構築

Android の開発環境では、iOS と異なり Windows 端末に全て環境を構築できる。ただし、開発の対象となる Android 実機の PC 接続用ドライバは事前にインストールが必要となる。Android の機種によってインストール方法が異なるため、機種の製造元が提供する方法を確認してインストールを行う。

SDK の取得

Delphi/400 上で対象のデバイス OS に合わせた開発を行うために、Android でも SDK の取り込みが必要になる。Delphi/400 では、Android SDK 専用の管理ツールとして「AndroidTools」が用意されている。

スタートメニューから［Embarcadero RADStudio XE5|Android SDKs］より「Android Tools」でツールを起動することができる。【図 15】

起動すると AndroidSDKManager の画面が表示されるので、開発対象の AndroidOS バージョンにチェックをして、「Install」ボタンを押下する。これだけで必要な SDK を取り込むことができる。Android は iOS に比べて OS のバージョンも非常に多いが、全ての SDK を取り込むとかなりのディスク容量を使用するので、必要なバージョンだけを取り込んだほうがよい。【図 16】

3-3. マルチデバイス開発

ここまでで iOS、Android それぞれの開発環境のポイントを説明したが、実際の開発では、iOS も Android も同じようにネイティブアプリケーションのプログラムを開発することができる。Delphi/400 のネイティブアプリケーションの開発画面は、従来の Windows フォーム設計部にスマートデバイス画面を表示して開発することができ、このスマートデバイス画面に対して、コンポーネント配置、コーディングといった従来の C/S 型、Web 型と同じ手法で開発を行う。【図 17】

スマートデバイス画面は、右上のプロジェクトマネージャにおいて、開発対象となるデバイスが選択できるようになっており、対象のデバイス（iOS、Android、Mac）を指定するだけで、1 つのプログラムからそれぞれのネイティブアプリケーションを生成できる。

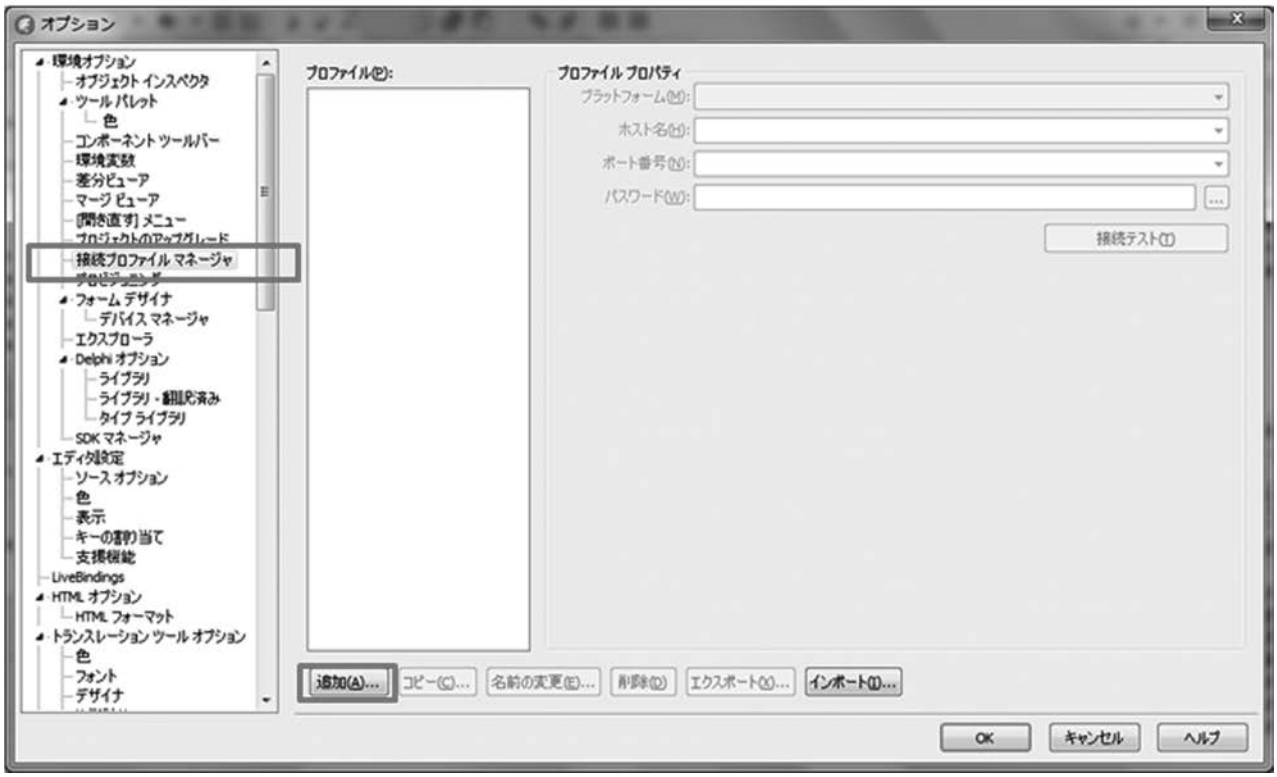
これは FireMonkey と呼ばれる Delphi の新しいフレームワークを使用しており、コンパイル先に指定したデバイス向けのアプリケーションに自動で対応できる。デバイスの違いは FireMonkey フレームワークが吸収してくれるため、開発者は Delphi 言語で開発するだけで、Windows のみならず、スマートデバイスにも対応できる。【図 18】

この開発手法は「マルチデバイス開発」と呼ばれ、Windows や iOS、Android など複数デバイスのアプリケーション開発が必要となる場合に、Delphi/400 ならではの高い生産性を発揮することができる。

図7　PAServerの起動

図8　PAServerのコンソール

図9　接続プロファイルマネージャ



4.ネイティブアプリケーションの開発手順

この章では、Delphi/400 のネイティブアプリケーション開発の流れを、簡単なアプリケーション開発例を題材に詳しく説明していく。

4-1. 基本的な開発手順

ネイティブアプリケーションの開発は先の章でも述べた通り、従来の C/S 型、Web 型のアプリケーションと同様の手順で開発できるのでご安心いただきたい。

基本的な開発の流れは次のようになる。【図 19】

①画面でコンポーネントを配置する
②イベントにプログラミングを行う
③コンパイルして実行（実行時にデバイスにインストールされる）

今回はスマートデバイスのカメラ機能を組み込んだ簡単なアプリケーションを例として、開発の手順を説明していく。【図 20】

ネイティブアプリケーションを新規に作成する場合には、［ファイル｜新規作成］より「FireMonkey モバイルアプリケーション -Delphi」を選択する。

選択時にダイアログでテンプレート選択画面が表示される。【図 21】今回は基本となる「空のアプリケーション」を選択する。他にもリスト形式画面などのテンプレートが用意されているので、新規作成時には便利である。

新規作成されたスマートデバイス画面では、右上のコンボボックスで、デバイスに合わせた設計画面イメージを選択することができる。【図 22】

選択肢としては iPhone や iPad、Android の主要機種（Nexus、Galaxy）が用意されている。また Android の場合、市場の機種が多いため解像度ベースでの画面イメージも選択できる。これは画面表示サイズなどの設計イメージを容易にするための機能なので、開発したプログラムには直接影響しない（Android の見た目で設計して iOS にコンパイルすることも可能だが、表示デザインなどが調整しにくい）。

ここからはアプリケーションの開発内容を説明する。

①画面でコンポーネントを配置する

今回はフォームに ToolBar、Button、Image、ActionList のコンポーネントを【図 23】のように配置する。

Button は StyleLookup プロパティで、選択しているデバイスのイメージに合わせた表示スタイルが設定できるようになっているので、カメラのアイコンになるスタイルを設定しておく。他のコンポーネントも同様に StyleLookup プロパティで表示スタイルの設定が可能である。別のデバイスを切り替えた場合には、そのデバイスに用意された同様の表示スタイルが自動で適用される。

②イベントにプログラミングを行う

アプリケーションの処理は、コンポーネントのイベントにコーディングすることができる。今回は Button にカメラ機能を使うイベントを設定してコーディングを行う。

Button の Action プロパティで「標準アクションの新規追加」から「メディアライブラリ「TTakePhotoFromCameraAction」を設定する。【図 24】

これだけで、スマートデバイスのカメラ撮影機能を Button で利用することができる。

イベントタブには OnDidFinishTaking というイベントがあるので、このイベントをダブルクリックしてコーディング処理部分を作成する。OnDidFinishTaking イベントはカメラ撮影が終わったあとに実行されるイベント処理である。【図 25】

コーディングする内容は【ソース 1】のように 1 行だけ記述する。このデバイスで撮影された画像を画面の Image コンポーネントにセットするというプログラムコードである。

③コンパイルして実行

ここまでの作業でネイティブアプリケーションのプログラム自体は完成している。最後にコンパイルを行ってアプリケーションの動作を確認する。

プロジェクトマネージャ画面にコンパイル先のデバイスが選択できるようになっている。【図 26】

選択できるのは Android、iOS シミュレータ、iOS デバイスである。それぞれ「ターゲット」という部分に接続しているデバイスの端末名が表示されるので、端末名をダブルクリックして選択する。

今回は iOS のデバイスを選択してコンパイル実行（メニューの実行、または F9）でアプリケーションを生成して実行してみる。実行するとコンパイル完了あとに、iOS の実機上でアプリケーションがインストールされ、作成したカメラアプリケーションが起動する。アプリケーションのボタンを押すとカメラ機能が起動し、撮影を行うことができる。撮影した画像はアプリケーションの画面にセットされる。

これだけでカメラ機能を連携した iOS ネイティブアプリケーションが完成したことになる。【図 27】

それでは Android ネイティブアプリケーションでは、どのように開発するかというと、実は今作成したプログラムのコンパイル先を変更するだけでよいのである。

プロジェクトマネージャの「ターゲット」に表示される Android 端末を選択して、コンパイルしてみる。すると、プログラムは 1 行も変えていないので、同じアプリケーションが Android 上にインストールされて実行される。【図 28】

これが先に説明した Delphi/400 のマルチデバイス開発である。

1 つのプログラムからコンパイル先の指定だけで、複数のデバイスに対応できる。この開発手順を試していただくと、簡単にスマートデバイス向けのアプリケーションが開発できることを実感していただける。

4-2. IBM i 活用手順

先の例では、ネイティブアプリケーションの基本的な開発手順を説明してきた。ここからはネイティブアプリケーションから、Delphi/400 の機能を使って IBM i へ接続する方法を説明する。

通常、PC から社内のデータベースに接続する場合は、PC にデータベース接続用のドライバをインストールしている。

しかし、スマートデバイスでは、社内のデータベースに直接接続することはできない。これは IBM i に限らず、Oracle や SQLServer など、どのデー

図10　接続プロファイルの追加

図11　接続プロファイルの設定

タベースでも同じである。その理由は、iOS や Android などのデバイス上には、データベースに接続するためのドライバがインストールできないからである。そのため、スマートデバイスのネイティブアプリケーションから IBM i に接続する場合には、【図 29】のようにアプリケーションサーバーを経由した 3 階層方式の接続となる。

アプリケーションサーバーには、C/S アプリケーション同様に IBM i に接続するサーバーアプリケーションが必要になる。Delphi/400 では、このサーバーアプリケーションにも「DataSnap」と呼ばれる専用開発機能が用意されており、容易に開発が可能である。

サーバーアプリケーションには、SQLConnection や SQLQuery などの DB コンポーネントを設定したり、処理関数をプログラミングすることで機能を実装する。

「DataSnap」を使ったサーバーアプリケーションの開発方法は、本稿では割愛させていただくが、『Migaro. Technical Report No.5』に「DataSnap を使用した 3 階層アプリケーション構築技法」というレポートで詳しくまとめているので、こちらを参考にしていただきたい。

それでは、この「DataSnap」のサーバーアプリケーションに接続するネイティブアプリケーションの開発手順を説明する。

今回は、IBM i 上の得意先マスタ（CUSTOMER ファイル）の一覧を表示するアプリケーションを例として、開発手順を確認していく。【図 30】

①画面でコンポーネントを配置する

まずフォームに ToolBar、Label、Switch、ListView を【図 31】のように配置する。Label には“得意先一覧”とタイトル名を設定しておく。

スマートデバイス上に配置する Label では、Label の表示文字が部品以上に長い場合、“得意先…”というように画面上で自動省略されてしまう。そのため、Label の AutoSize プロパティを True に設定しておくことをお勧めする。この設定をしておくと、表示文字の長さに合わせて部品のサイズ側が自動調整してくれる。【図 31】

次に IBM i の接続コンポーネントを配置する。フォームに SQLConnection、DSProviderConnection、ClientDataSet を【図 32】のように配置する。

それぞれのコンポーネント設定内容を説明する。SQLConnection コンポーネントは ConnectionName プロパティに“DataSnapConnection”を設定する。【図 33】

そして Params プロパティに、アプリケーションサーバーの IP アドレスとポート番号を設定しておく。これにより「DataSnap」のサーバーアプリケーションに接続することができる。また、スマートデバイス上ではダイアログが出ないので、LoginPrompt プロパティは False に設定しておく必要がある。

次に、DSProviderConnection コンポーネントを設定する。【図 34】

SQLConnection プロパティには、先ほど設定した SQLConnection を指定する。ServerClassName プロパティには、サーバーアプリケーションで作成しているクラス名を設定するが、デフォルト名では“TServerMethods1”を指定する。

最後に ClientDataSet コンポーネントの設定を行う。【図 35】

RemoteServer プロパティに、先ほど設定した DSProviderConnection を指定する。この設定を行うと、ProviderName プロパティにサーバーアプリケーションの Provider が自動で表示されるので選択する。最後に CommandText プロパティに IBM i にアクセスしたい SQL 内容をセットする。

今回は得意先マスタ（CUSTOMER ファイル）にアクセスするため、次のような SQL を記述する。

“SELECT * FROM CUSTOMER”

ここまでの設定で IBM i へ接続して、得意先マスタのデータにアクセスすることができる。ClientDataSet をダブルクリックして、リストを右クリックから「すべてのフィールドの追加」を選択する。これで IBM i から得意先マスタの項目を取り込むことができる。

次に、アクセスしたデータをアプリケーションの画面上に表示する部分を作成する。C/S 型、Web 型アプリケーションの場合、DBGrid などのコンポーネントが便利だが、残念ながら FireMonkey のフレームワークには同じコンポーネントが存在しない。そのため、今回は LiveBinding というビジュアルリンク機能を使用する。LiveBinding は取得したデータを画面上のコンポーネントに自動でリンクしてくれる便利な機能である。これは簡易作成機能なので、もちろんプログラミングでデータをセットしても問題ない。

画面に配置した ListView にデータを表示するには、フォームを右クリックから「ビジュアルにバインド」を選択する。【図 36】

開発画面下部にビジュアルバインディングの設計画面が起動されるので、この画面で項目のリンク設定を行う。

リンクの方法は簡単である。データ項目と表示したいコンポーネントの項目をドラッグ＆ドロップするだけで、感覚的にリンクを設定できる。【図 37】

設定ができたら ClientDataSet の Active プロパティを True に設定すると、表示結果を確認できる。【図 38】

このように、実際に開発設計画面上に IBM i のデータがリンク表示されるので、プログラムをコンパイルしなくとも、画面を細かく調整することができる。

②イベントにプログラミングを行う

アプリケーションで得意先マスタのデータを表示 / 非表示ができるように Switch コンポーネントの OnSwitch イベントにコーディングを行う。プログラムを【ソース 2】のように 1 行だけ記述する。

これでスィッチの ON/OFF によって、データの表示 / 非表示を操作できる。データの表示制御は LiveBinding が自動で行ってくれるので細かいプログラミング制御は必要ない。

③コンパイルして実行

ここまでの作業でプログラムは完成である。プロジェクトマネージャで対象のデバイスを選択して、コンパイルを実行する。マルチデバイス開発なので、iOS でも Android でも可能である。【図 39】

このように IBM i のデータを活用するネイティブアプリケーションも、ほとんどプロパティの設定だけで簡単に開発

図12　SDKの追加

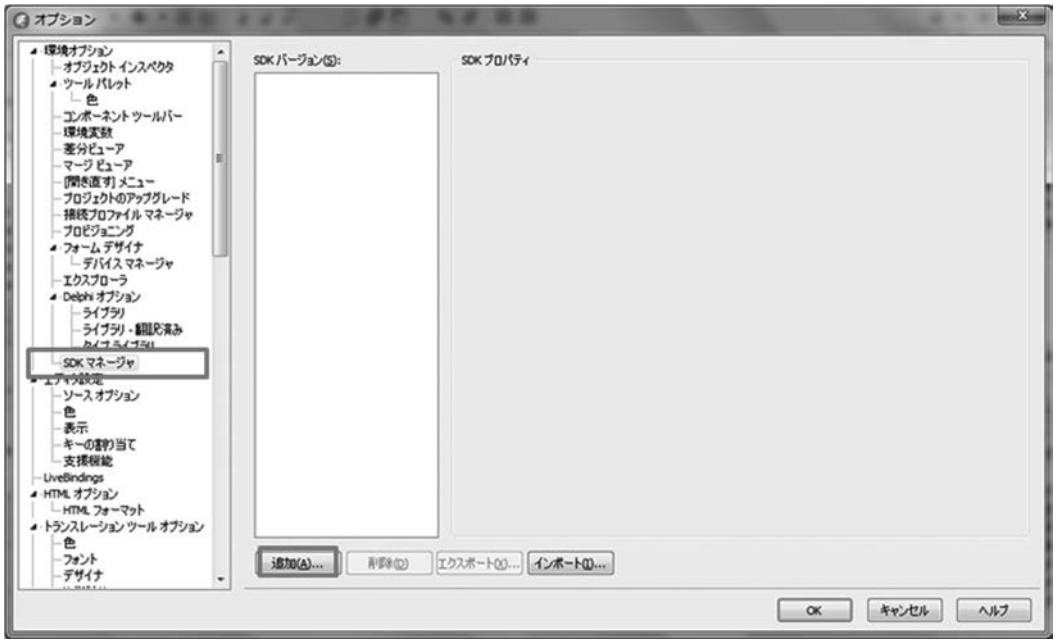


図13　SDKの選択

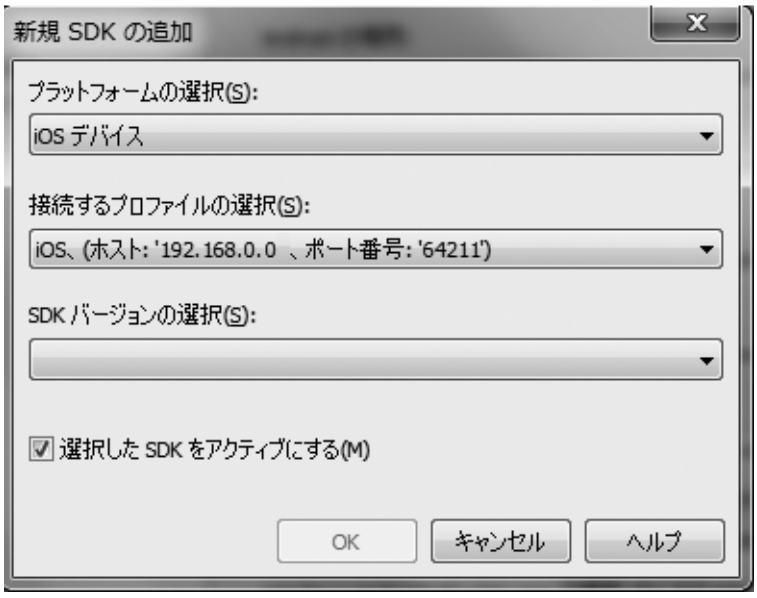


図14　Androidアプリケーション開発環境

できる。例えば、この得意先マスタの一覧のデータをタッチすることで、得意先の受注一覧を表示するといった機能カスタマイズも、同じような手順で作成できる。【図 40】

こうしたアプリケーションであれば、コンポーネントの設定と数行のコーディングで開発できてしまう。

5.ネイティブアプリケーション開発のポイント

前章までは Delphi/400 のネイティブアプリケーションの開発手順について説明してきた。この章では、ネイティブアプリケーション開発時にヒントになりそうなポイントをいくつか補足したい。

5-1. iOSとAndroidの違い

Delphi/400 では、iOS でも Android でも 1 つのプログラムで開発できるが、iOS と Android ではデバイスの違いがあるため、設計上でいくつか考慮しておく点がある。

1 つはハードウェアキーの違いである。Android には「ホームボタン」や「戻るボタン」「メニューボタン」が物理的に存在するが、iOS には「ホームボタン」しか用意されていない。【図 41】

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ってしまう。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くる。

また、デバイスの構造が違うので、アプリケーション内でファイルを扱う場合にも考慮が必要である。

例えば、アプリケーションで音声や動画を流したりする場合には、オーディオファイルなどをアプリケーションと一緒に配布する必要がある。

しかし、当然ながらデバイス上の構造が違うので、ファイルを保存するためのパスも違ってくる。従って、ファイルのパスなどは、iOS/Android ごとに設定をしておく必要がある。

ファイルの配置は［プロジェクト｜配置］からデバイスごとに設定できるので、iOS であれば“.¥StartUp¥Documents¥”、Android であれば“assets¥internal¥”に設定する。【図 42】（アプリケーションの外に配置する場合はパスも異なる）

こうしたアプリケーション固有の配置ファイルパスをプログラムで取得する場合には、コーディングも違ってくる。配置した Alerm.mp3 というオーディオファイルを TMediaPlayer コンポーネントに設定する場合であれば、【ソース 3】のように記述することができる。

デバイスごとに異なるプログラムコードは、iOS であれば｛$IFDEF IOS｝、Android であれば｛$IFDEF ANDROID｝というタグを記述しておけば、特定のデバイス実行時のみ有効なコーディングも可能である。

もちろん配布したファイルを読み込むだけでなく、C/S 型アプリケーションのように設定ファイルをデバイス上に作成・保持することもできる。またネットワークに接続されていない場合に、ローカル環境で動作するように CSV などのファイルデータをデバイス内で保存し、ネットワークにつながってから IBM i にデータを登録するといったことも実現できる。

5-2. アプリケーションのカスタマイズ設定

プログラミングとは別に、ネイティブアプリケーションで設定しておけるカスタマイズ設定も補足しておく。

アイコンのカスタマイズ

例えば、スマートデバイスにインストールしたアプリケーションのアイコンも設定が可能である。［プロジェクト｜オプション］から「アプリケーション」を選択するとデバイスごとにアイコンを設定できるようになっている。【図 43】

ここで png などの画像ファイルで作成した任意のアイコンを設定すると、インストール時に自社用のアイコンで登録することも可能である。アイコンはデバイスによって解像度がさまざまなので、対象のデバイスに合わせたサイズのアイコンを用意する必要がある。

デバイス向きのカスタマイズ

また、同じ［プロジェクト｜オプション］から「アプリケーション」の設定画面で「向き」というタブを選択すると、デバイス固有の向き設定を固定化することもできる。【図 44】

標準では縦横の画面変更時に表示調整が行われるが、例えば縦専用で設計した画面の場合、この設定を行えば横表示にならないようにアプリケーション画面を縦固定にすることができる。

セキュリティ権限のカスタマイズ

他にも Android アプリケーションの機能で、特別な権限が必要な場合は、［プロジェクト｜オプション］から「使用する権限」の設定画面で、権限を付与したり、あるいはセキュリティ上で制限したりすることも可能である。設定はチェックの ON/OFF だけで、かなり細かい設定まで行うことができる。【図 45】

5-3. ネイティブ機能の連携例

先に説明した開発手順では、ネイティブ機能連携の一例としてカメラ機能の連携開発を説明したが、他にもさまざまなネイティブ機能を連携することができるので、一例を紹介しておきたい。

例えば、カメラ機能を応用して、バーコードを読み取ったり【図 46】、GPS の位置情報を利用して GoogleMap を利用することもできる。【図 47】

また、加速度センサーなどを使用すれば、デバイスの傾きなどを利用した画面操作を行うことも可能である。さらに、音声データを録音・再生したり、先にも紹介したアイコンに通知を表示することもできる。【図 48、図 49】

この章で紹介した設定機能やネイティブ連携機能は、Web アプリケーションでは実現できない内容も多く、ネイティブアプリケーションならではの機能性、拡張性の高さといえる。

6.ネイティブアプリケーションの配布・運用

この章では開発したアプリケーションをユーザーのスマートデバイスに、どのように配布して運用するかについて、説明する。

6-1.社内公開と一般公開

ネイティブアプリケーションのスマートデバイスの配布には大きく、社内公開と一般公開の配布方法がある。それぞれの特徴は次の通りである。【図 50、図 51】

図15　Android Toolsの起動

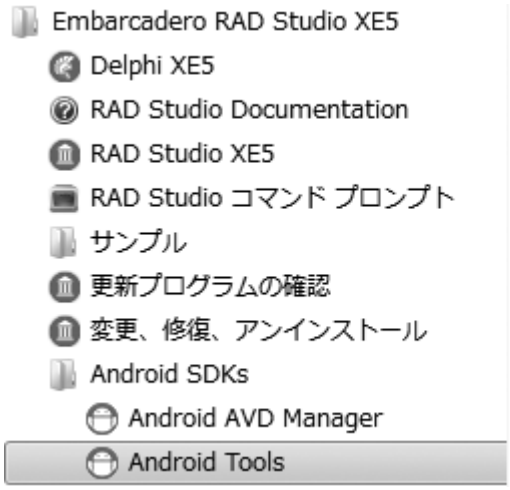


図16　Android SDK Manager

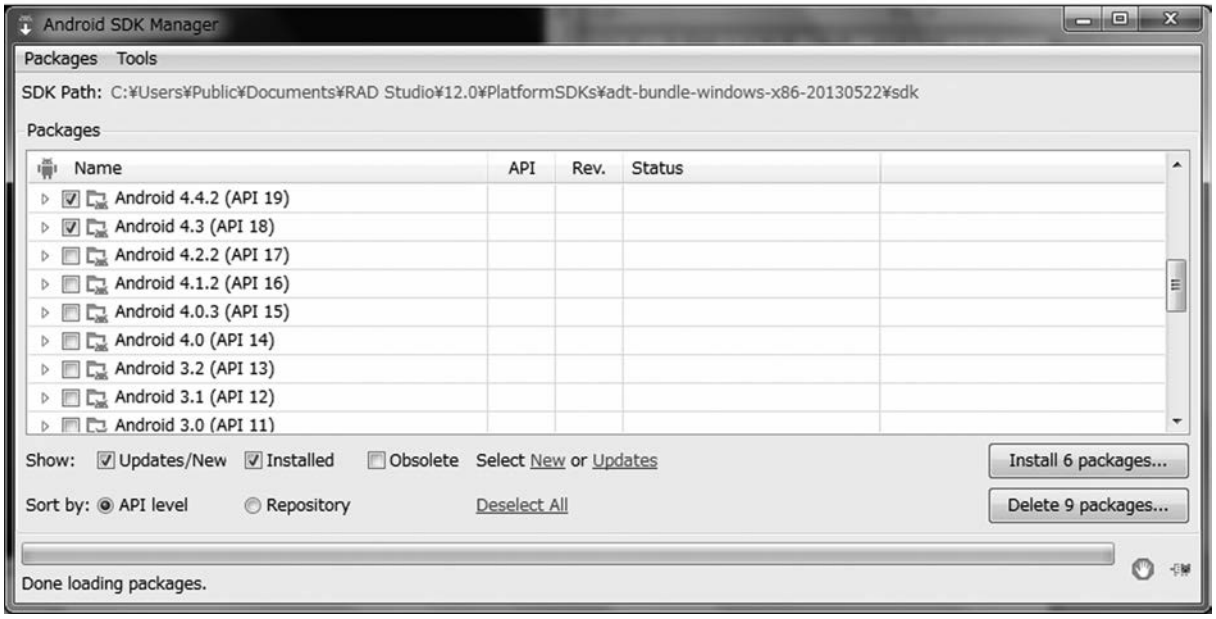


図17　スマートデバイスアプリケーション開発画面

社内公開

社内公開の場合、開発したネイティブアプリケーションを社内用のWebサーバーに配置して、各スマートデバイスからインストールする。社内専用のアプリケーションであれば、自由に開発、運用することができるため、ストアなどの審査も必要としない。アプリケーションの配布タイミングも制約はないので、自由にリリースすることができる。

ただしiOSの場合には、開発環境の章でも説明した通り、iOS Developer Programに加入しなければ配信することができず、また配信台数もプログラムの種類によって制限されるので考慮が必要である。

一般公開

一般公開の場合、開発したネイティブアプリケーションを専用ストアから配信して、各スマートデバイスからインストールする。専用ストアはiOSであればApp Store、Androidであれば GooglePlayStoreを利用することになる。

ストアを経由して配信するため、インストールも容易となるが、誰でもインストールすることができてしまうため、ストア配信時には審査がある。そのため、社内利用限定のアプリケーションは配信することは難しく、またリリース後に変更があっても、すぐにリリースはできない。

6-2. ネイティブアプリケーションの配布ファイル

社内公開でWebサーバーから配信する場合、アプリケーションファイルを配置してHTMLからリンクすることになる。

アプリケーションファイルはコンパイルの際に生成することができる。iOSであれば、ipa、plistファイル、Androidであればapkファイルが配布の対象となるので、これらをWebサーバー上に配置する必要がある。

HTMLのリンクの記述内容は難しくはないが、例を記載しておくので参考にしていただきたい。【iOS：ソース4, Android：ソース5】

またiOSの場合、Webサーバー上に配置するplistファイルはipaファイルへのURLリンクパスを含めて作成する必要がある。メモ帳などで編集できるので、以下のように作成する。【ソース6】

6-3. Webサーバーの設定・考慮点

Webサーバーは IISや Apacheなど使用することができ、htmlは先に説明した内容を公開すればアプリケーションを配布できる。

ここでもiOSでは、1つ注意点がある。

iOS7.1からはApple社での仕様が大きく変わり、SSLでの配信でなければ、インストールを行うことができなくなっている。つまり、通常のhttpでの配信を行えないので、httpsに対応したSSL配信ができるWebサーバー環境が必要になってくるので注意していただきたい（iOS7.0まではhttpで配信が行える）。

また、「6-2」で説明した配布アプリケーションファイルの拡張子は、Webサーバーに MIMEタイプとして設定を登録しておく必要がある。IISの場合は、IISマネージャを使用して、サーバーの「プロパティ」ページで次の MIMEタイプを追加する。Apacheの場合は、mime.typesに追加する。各拡張子の設定内容は次の通りである。

ipa ファイル
　application/octet-stream
plist ファイル
　text/xml
apk ファイル
　application/vnd.android.package-archive

ここまで設定が一度完了すれば、アプリケーションを開発するごとにファイルを配置して、リンクの追加だけで配布・運用することができる。

7.まとめ

本稿ではDelphi/400の新機能であるiOS/Android向けの開発手順や運用ポイントを説明してきた。スマートデバイス開発と聞くと、敷居が高く感じられる開発者の方も多いだろうが、Delphi/400では、従来のC/S、Web型と同様の手法でスマートデバイスのネイティブアプリケーショも開発できる。

冒頭でふれた通り、これからスマートデバイス向けに社内アプリケーションが必要とされる機会が増えてくる。

しかし、忘れてはならない重要なポイントがある。それは、これまで使用しているWindowsの社内システムがスマートデバイスの社内システムに置き変わることは少ないということである。

実際にWindows開発者に対して実施されたアンケート調査結果では、モバイルアプリケーションはデスクトップアプリケーションの置き換えにはならないという意見が9割を超えていた。【図52】

これはスマートデバイスとPCは使い勝手や用途が異なるので、Windowsの社内システムは今後も必要とされるということを意味している。つまり今後は、両システムの併用が一般的になるかもしれない。

そうしたとき、今回紹介したDelphi/400のマルチデバイス開発は、これまでのWindows向けの開発スキルをスマートデバイスでも活かせる技術であり、社内開発を拡張していくための最適解の1つといえる。Delphi/400を使って、今後の社内システムでスマートデバイスに対応される場合には、本稿を開発の足がかりにしていただければ幸いである。

M

図18　FireMonkeyフレームワーク

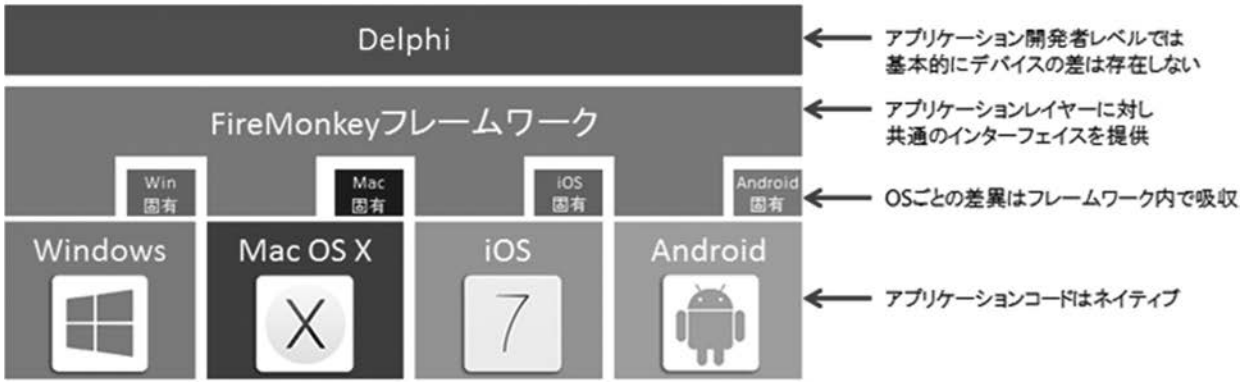


図19　開発の流れ

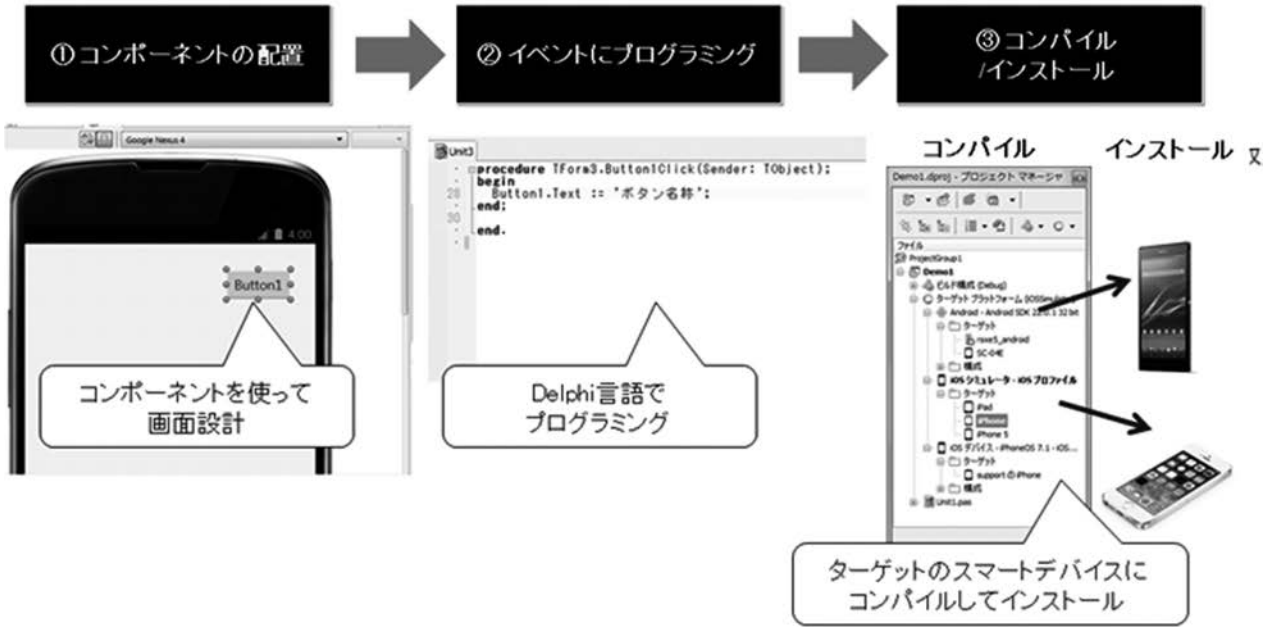


図20　カメラアプリ

図21　テンプレートの選択

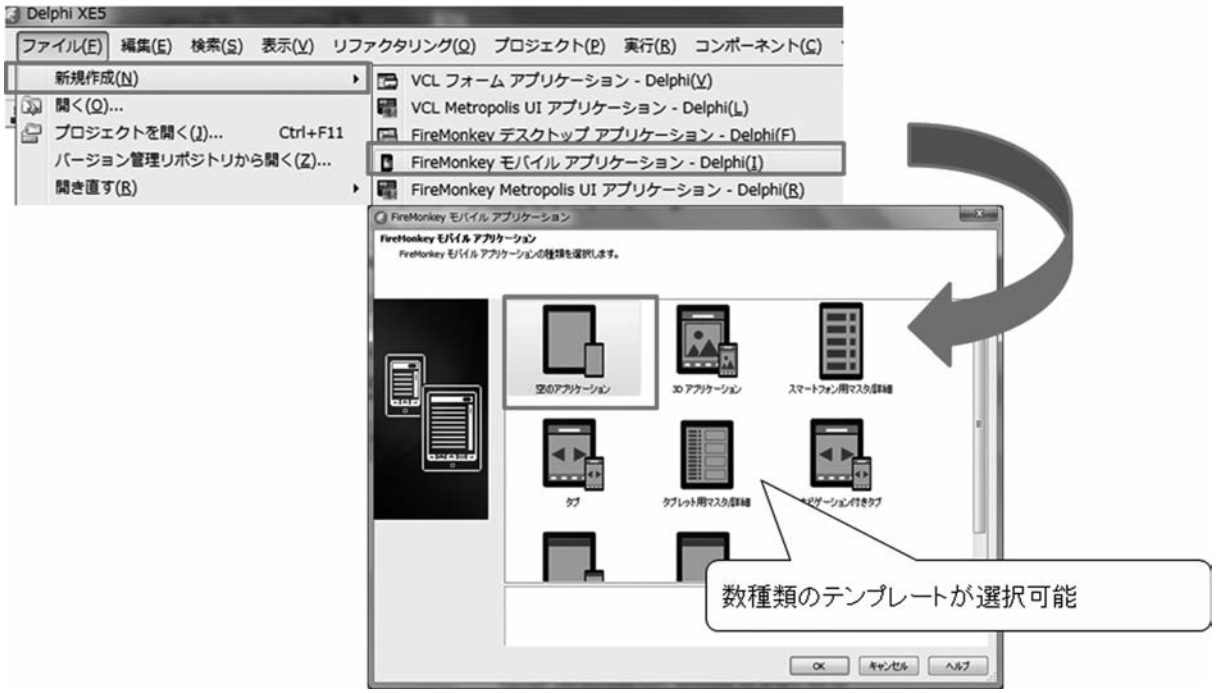


図22　設計デバイスイメージ

図23　コンポーネントの配置

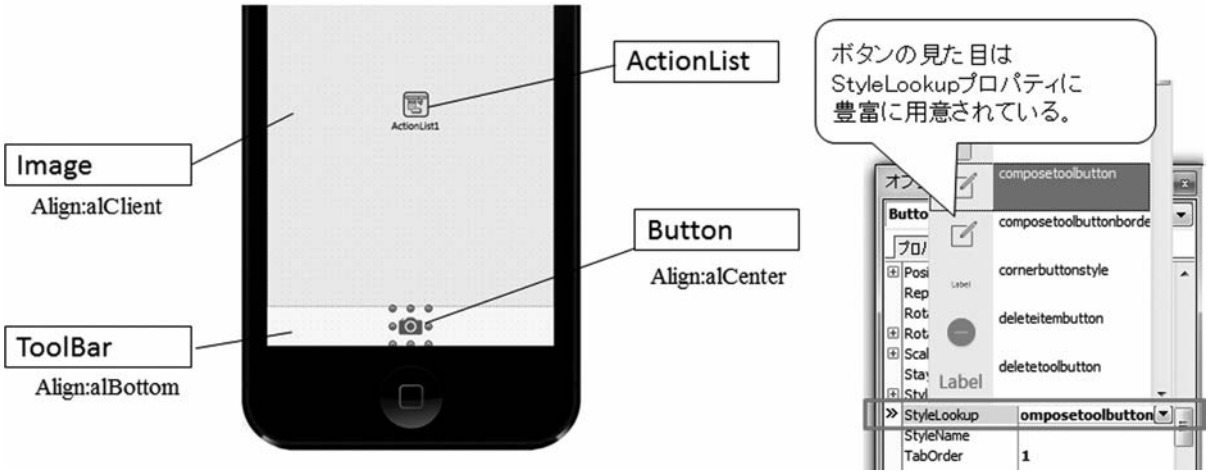


図24　Actionの設定

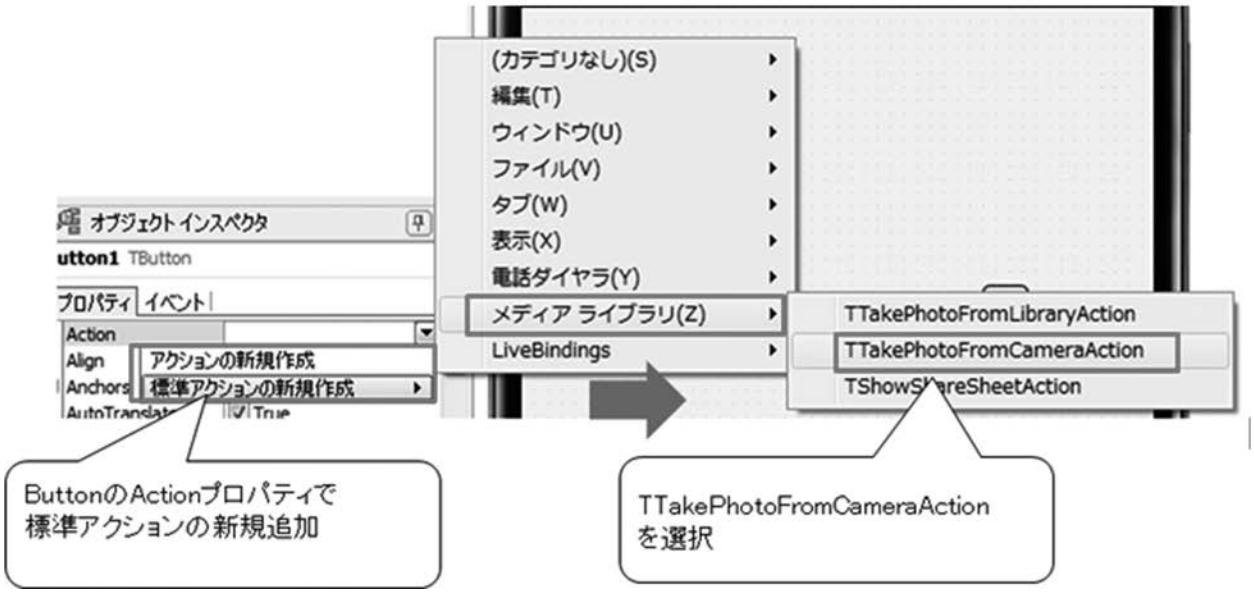


図25　カメライベントの設定

ソース1

OnDidFinishTaking処理（カメラ撮影終了処理）

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

図26　コンパイルターゲット

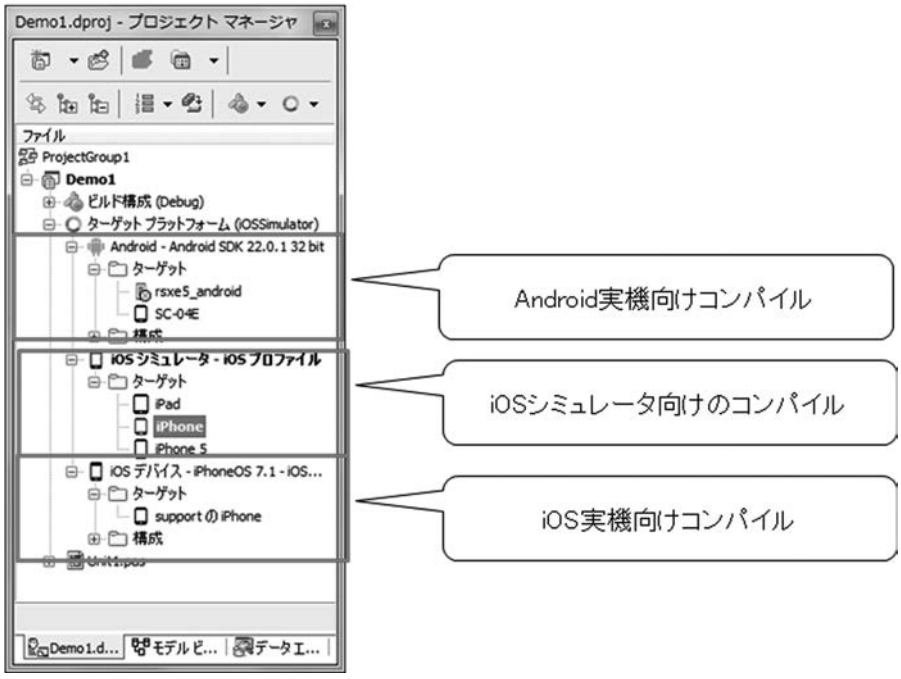


図27　iOSアプリケーションの実行

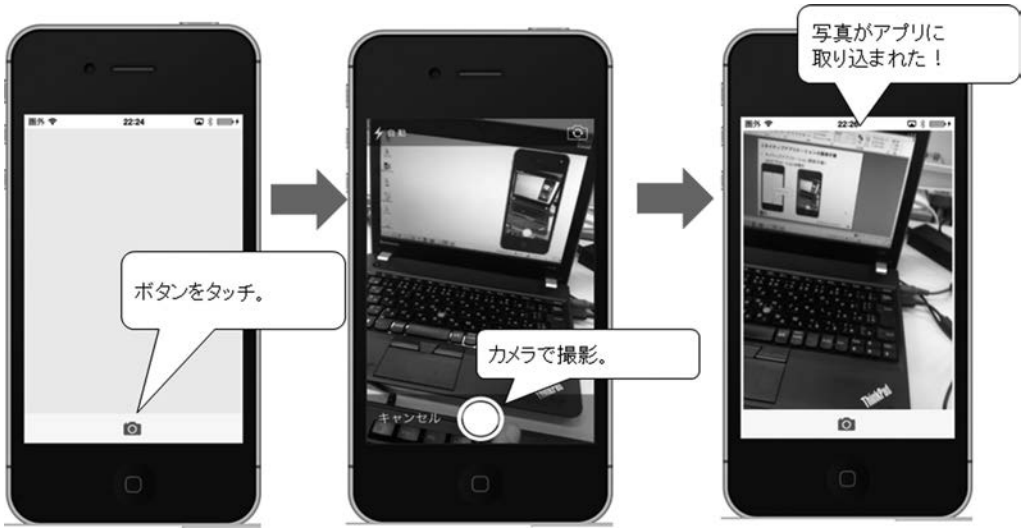


図28　Androidのアプリケーションの実行

図29　IBM iへの3層接続

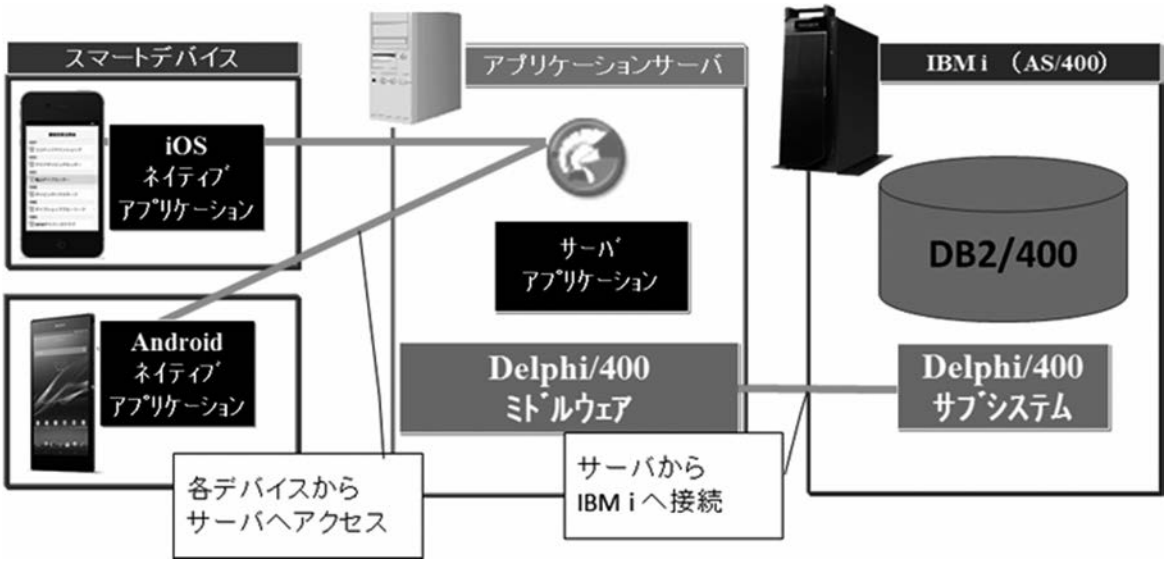


図30　得意先一覧照会アプリ

図31　コンポーネントの配置

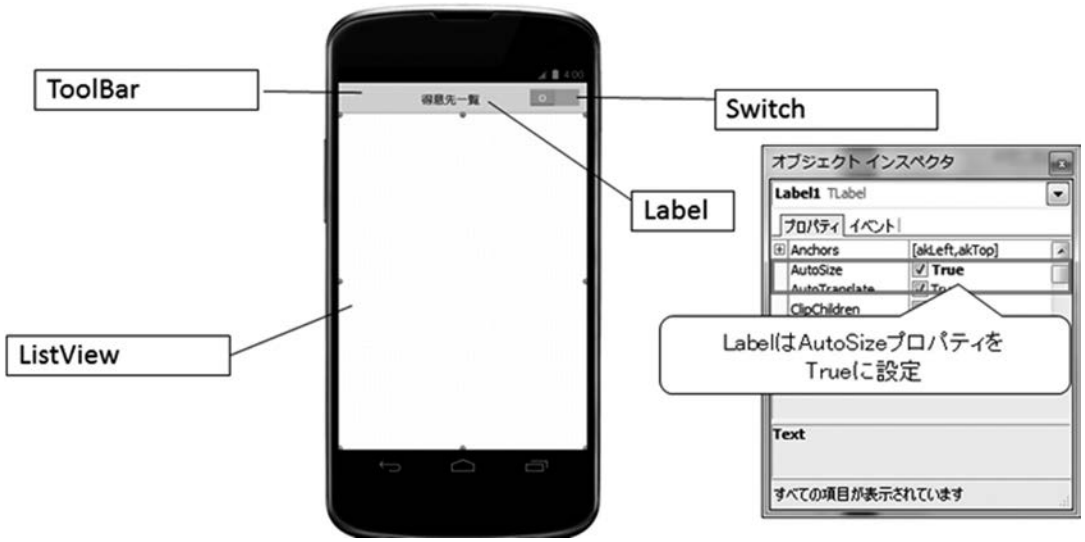


図32　DBコンポーネントの配置

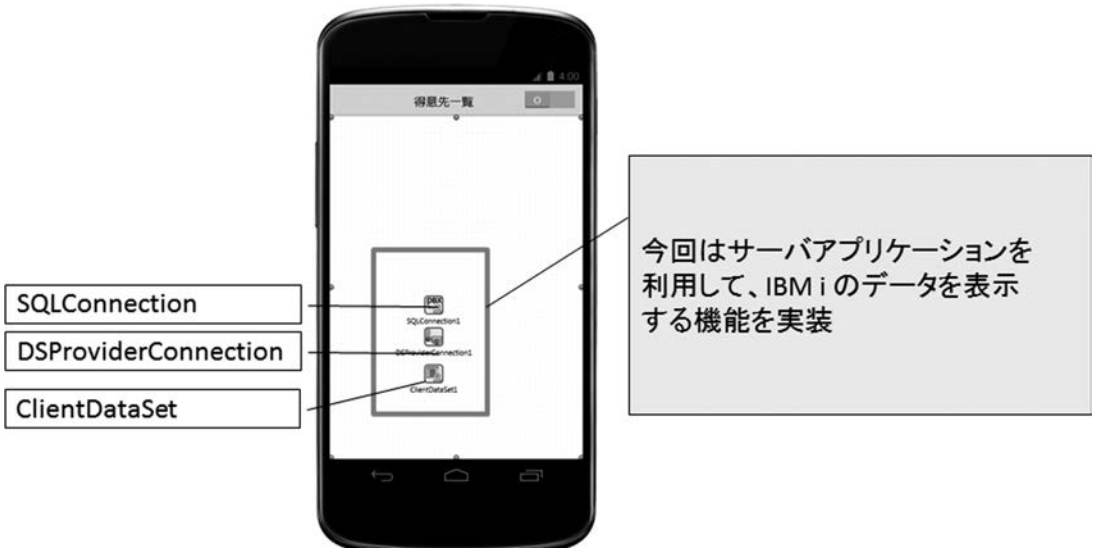


図33　SQLConnectionコンポーネント設定

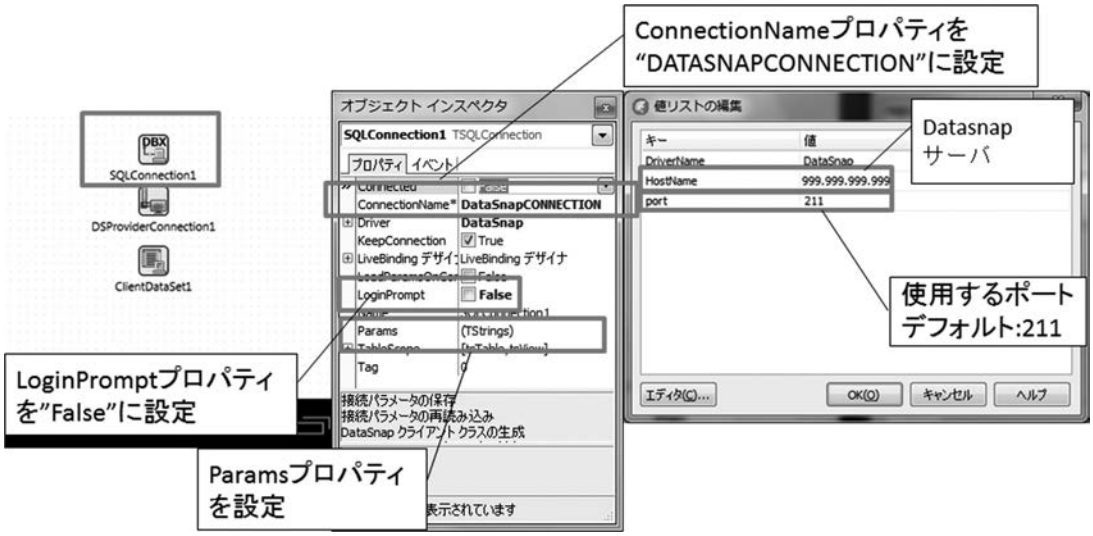


図34　DSProviderConnectionコンポーネント設定

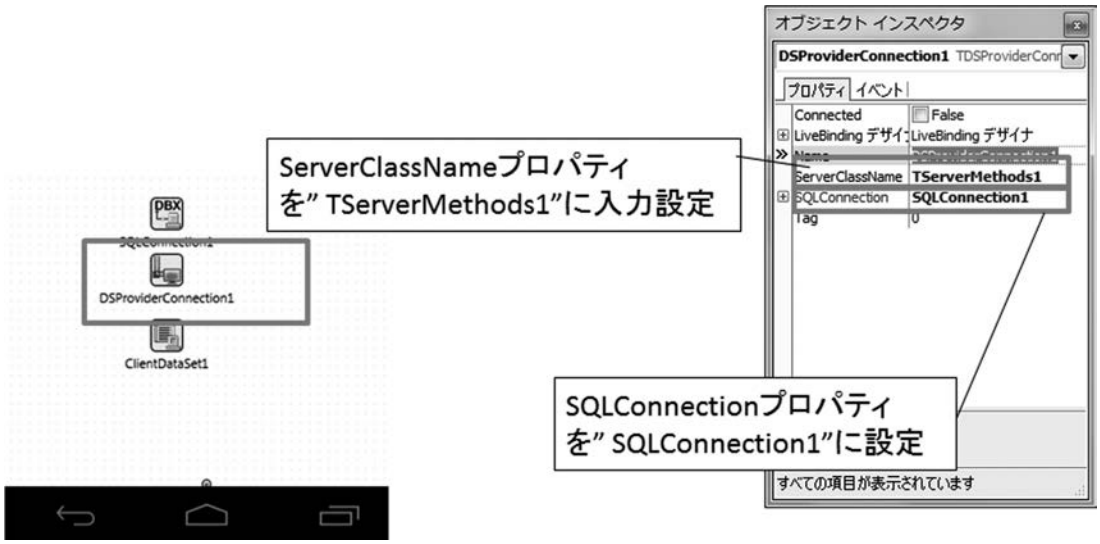


図35　ClientDataSetコンポーネント設定

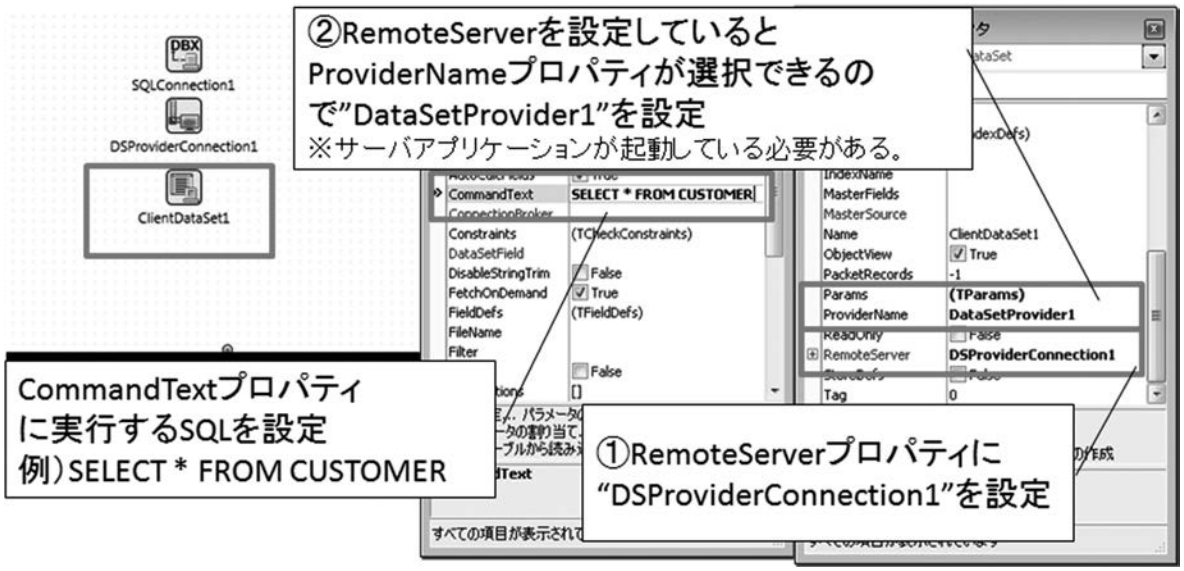


図36　LiveBindingの設定

図37　LiveBindingのリンク機能

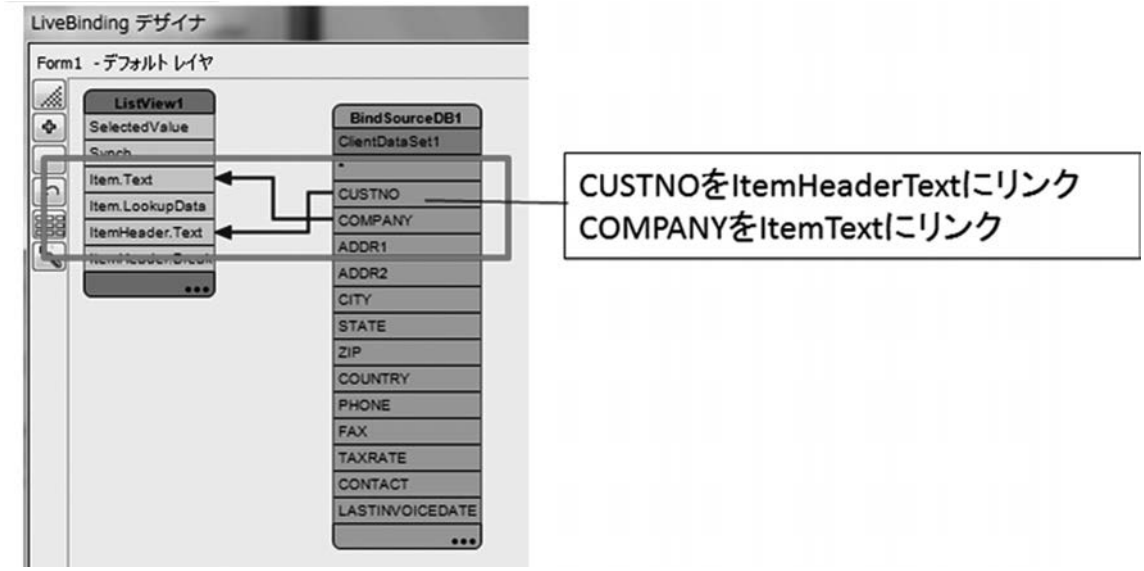
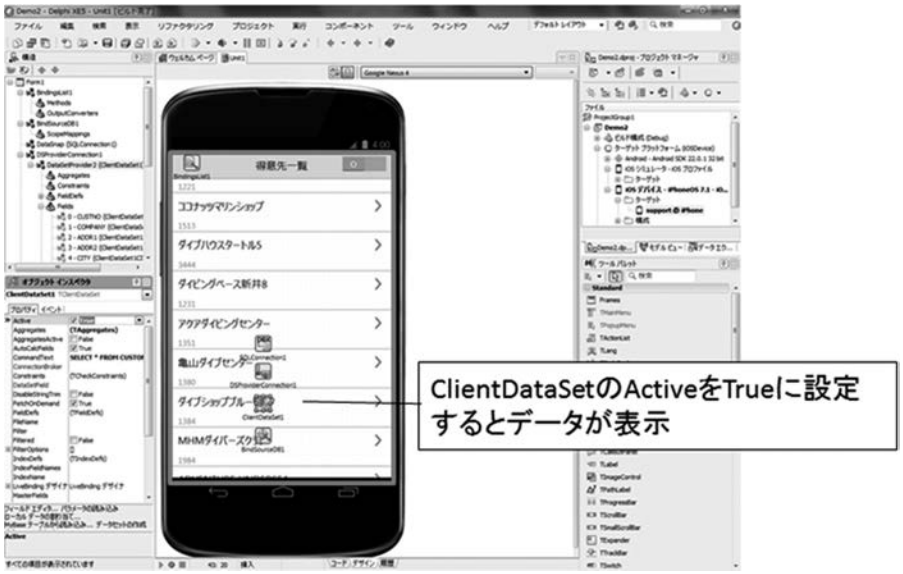


図38　IBM iデータの表示



ソース2

スイッチ切り替え処理

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

図39　iOS/Androidアプリの実行

図40　カスタマイズ例

図41　デバイスの違い

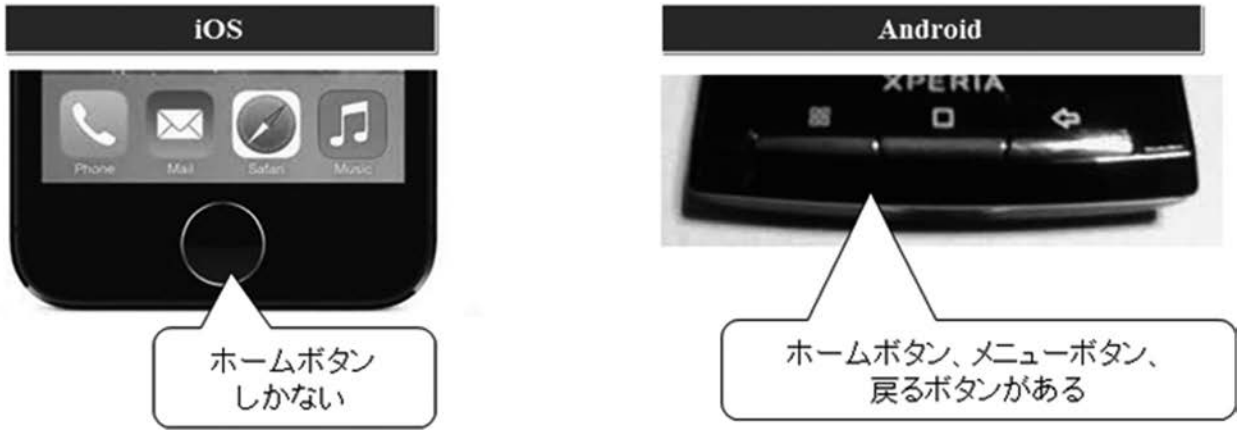


図42　ファイル配置の設定

ソース3

ファイルパスの取得例（iOS)

```
MediaPlayer1.FileName := GetHomePath +PathDelim + 'Documents' + PathDelim + 'Alerm.mp3';
```

ファイルパスの取得例（Android)

```
MediaPlayer1.FileName := TPath.Combine(TPath.GetDocumentsPath, 'Alerm.mp3');
```

図43　アイコンの設定

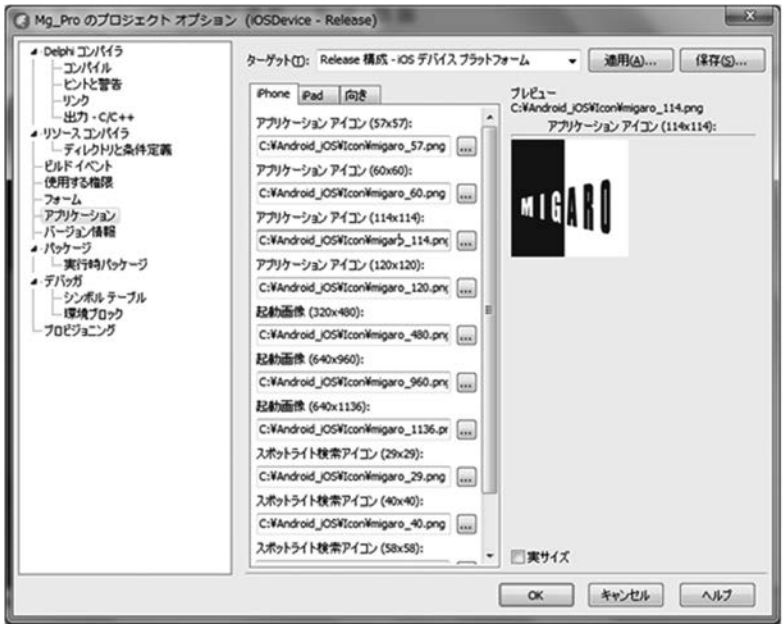


図44　デバイス向きの設定

図45　セキュリティ権限の設定

図46　バーコードの連携

図47　位置情報GPSの連携

図48　音声レコーディング連携

図49　通知機能連携

図50　配布の種類

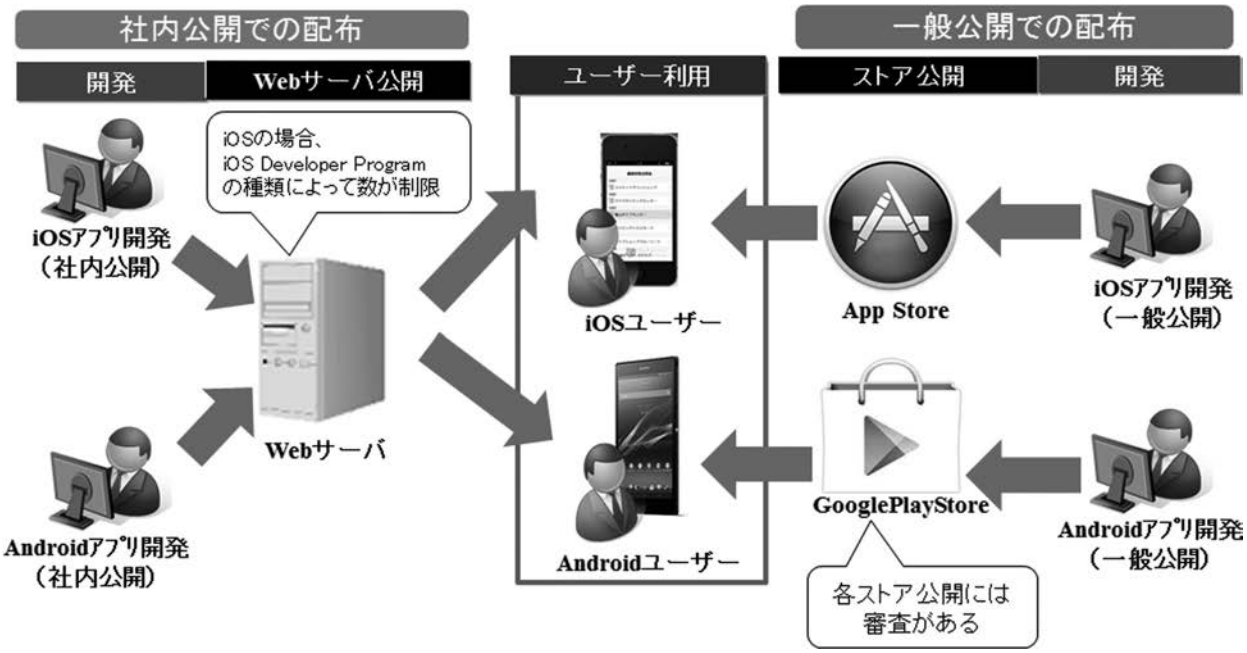


図51　メリット・デメリット

	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）

ソース4

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/Sample.plist">
   アプリケーションダウンロード</a><br>
</form>
```

ソース5

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="./Sample.apk" type="application/vnd.android.package-archive">アプリケーションダウンロード</a><br>
</form>
```

ソース6

ダウンロード用plist例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>items</key>
        <array>
                <dict>
                        <key>assets</key>
                        <array>
                                <dict>
                                        <key>kind</key>
                                        <string>software-package</string>
                                        <key>url</key>
                                        <string>https://Webサーバ/Sample.ipa</string>
                                </dict>
                        </array>
                        <key>metadata</key>
                        <dict>
                                <key>bundle-identifier</key>
                                <string>com.hogehoge.hogehoge</string>
                                <key>kind</key>
                                <string>software</string>
                                <key>title</key>
                                <string>Sample</string>
                        </dict>
                </dict>
        </array>
</dict>
</plist>
```

図52　今後の開発方針

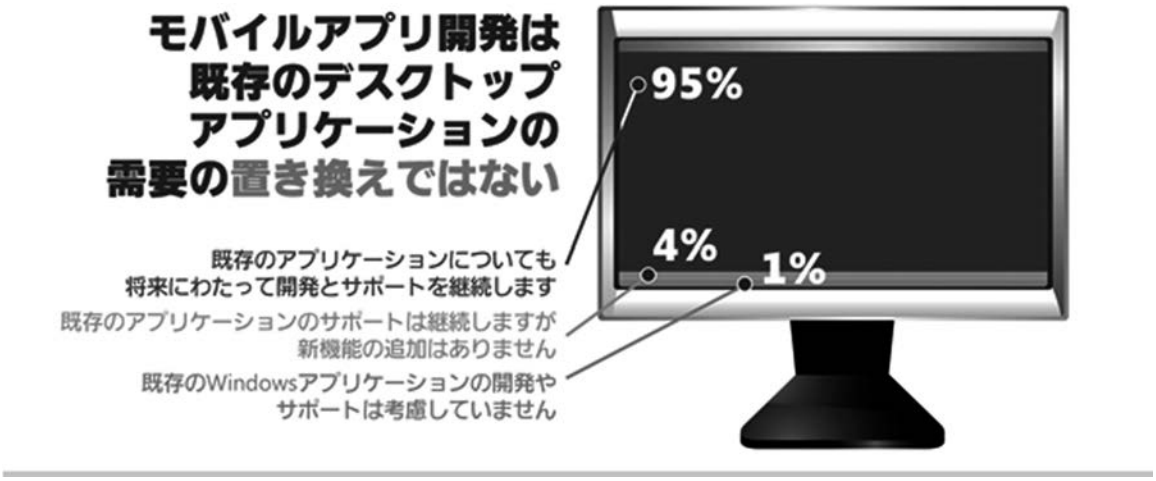


図53　PCアプリケーションとスマートデバイスアプリケーションの特徴の違い

PCアプリケーションの特徴	スマートデバイス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	どこでもすぐに使用できる
画面が大きく見やすい	タッチで感覚的に操作できる
ネットワークが安定している	カメラ、GPS、センサー等が利用できる
携帯性が低い	細かいキーボード入力は不向き