

【セッションNo. 2】

新バージョンDelphi/400 XE7ご紹介

－ マルチデバイスデザイナー機能で開発効率アップ！ －

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.マルチデバイス開発とは

2.新バージョンDelphi/400 XE7

2-1.FireUIによるマルチデバイス開発機能

2-2.アップテザリングによるアプリ連携機能

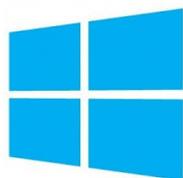
3.まとめ

1.マルチデバイス開発とは

1.マルチデバイス開発とは

- これまでシステム運用するデバイスはWindowsのPCだけが主な対象でしたが、ここ数年で対象のOS・デバイスの種類が急速に増加。

様々OS



Windows



OSX(Mac)



iOS



Android

様々デバイス



ノートPC



デスクトップPC



Windows
タブレット



iPhone



iPad



Android
スマートフォン



Android
タブレット

1.マルチデバイス開発とは

- モバイルを含めたこれからのシステム開発の課題

OS毎の異なる対応言語の習得

デバイス毎のアプリケーション開発

PCでしか対応できない・・・



iPhoneとiPadでも
画面設計が違う・・・

Microsoft
C#.net

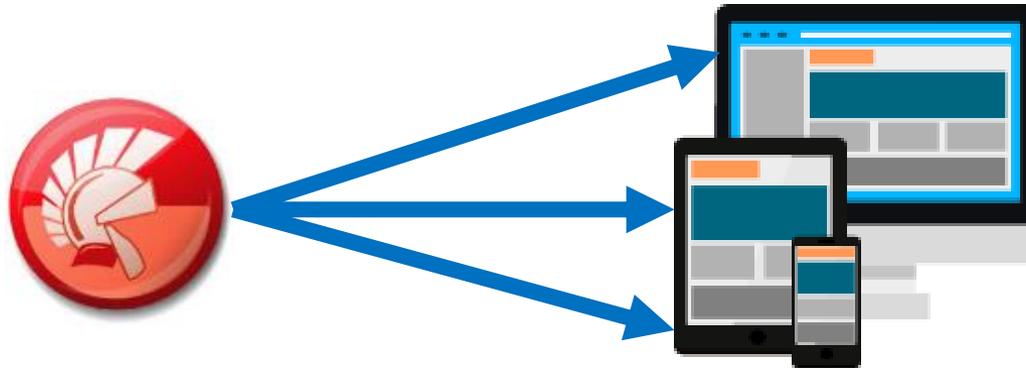


Androidはデバイスの
種類が無数がある・・・

使用するOS(および開発言語) × 使用するデバイスの種類

1.マルチデバイス開発とは

- マルチデバイス開発とは、
こうした様々なOS、デバイスに対応した統一開発



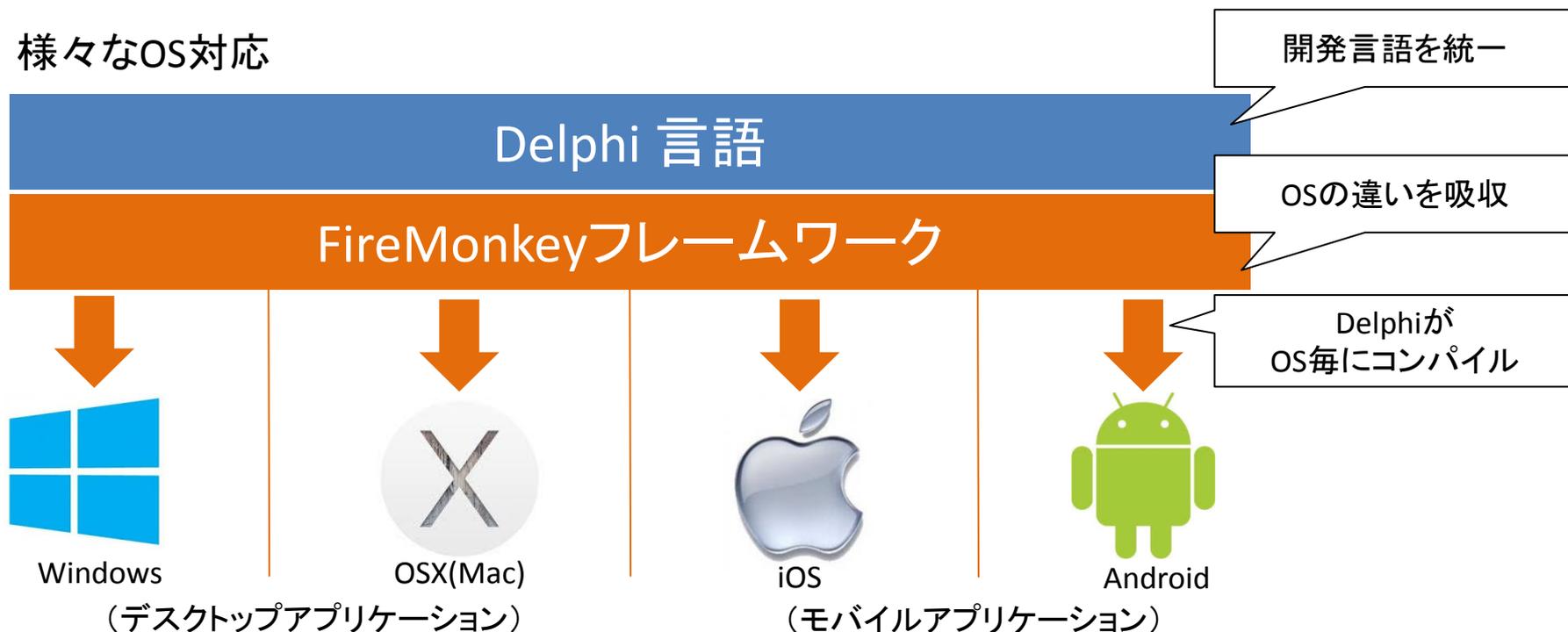
Delphiでは、Delphiスキルで全てのアプリケーション開発に
効率よく対応していける環境を目指しています

2.新バージョン Delphi/400 XE7

2.新バージョンDelphi/400 XE7

- Delphi/400 XE5では、Windows、OSX、iOS、Androidを対象としたネイティブアプリケーション開発がDelphi言語だけで可能。

様々なOS対応



XE5では画面設計が異なるアプリケーションは別々に開発が必要

2.新バージョン Delphi/400 XE7

- Delphi/400 XE7では、さらに進化して異なるデバイス毎の画面開発も1つのアプリケーションで対応可能。

様々なデバイス対応

1つのDelphiプログラム(ビジネスロジック)

デバイスの違いを細かく対応可能

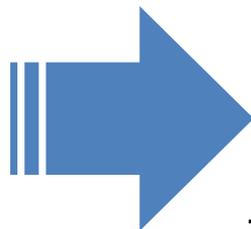
FireMonkeyマルチデバイスデザイナー(FireUI)



2.新バージョンDelphi/400 XE7

- Delphi/400 XE7のFireUI開発

従来の開発



FireUI開発

異なる画面パターン毎に
別々のアプリケーションとして開発

異なる複数の画面パターンも含めて
1つのアプリケーションとして開発



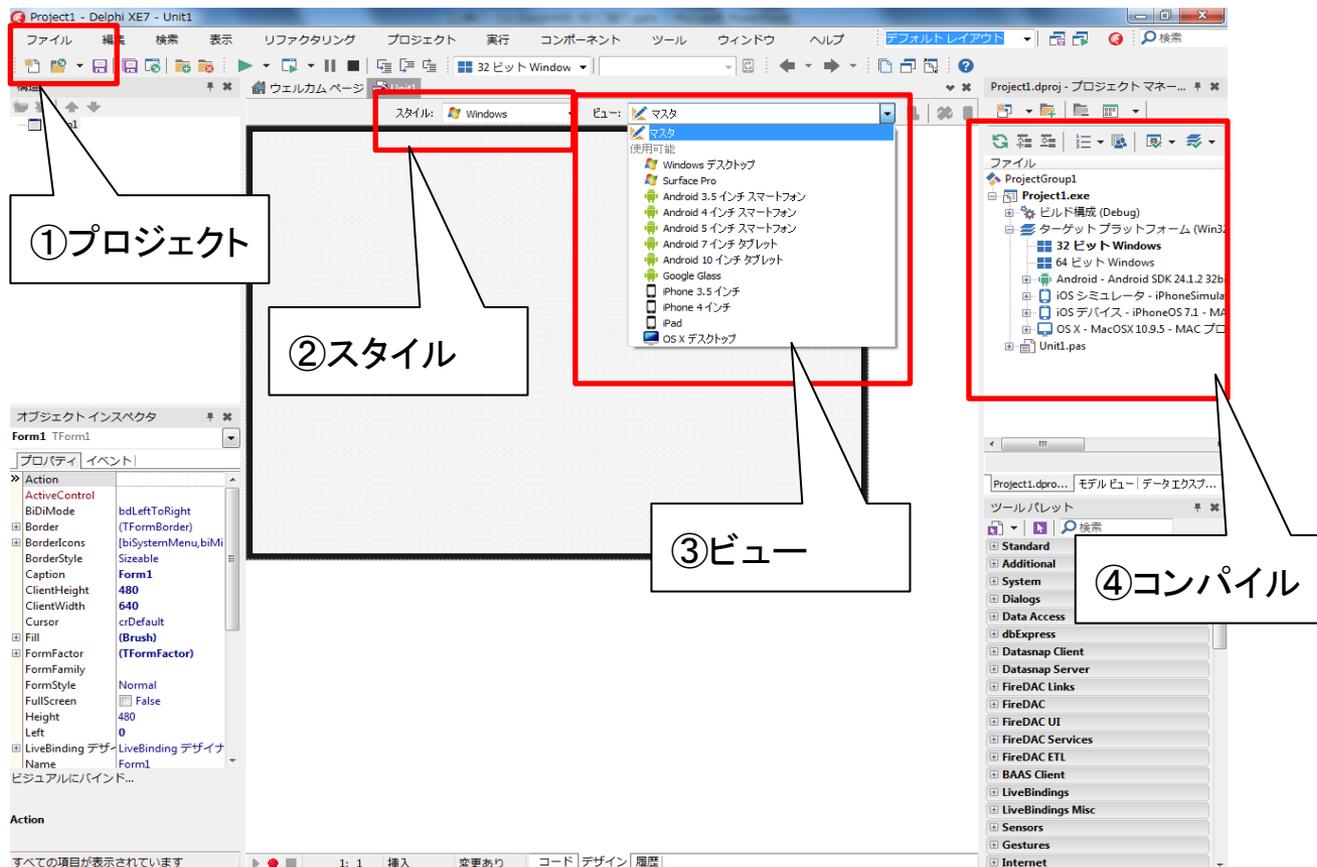
OSの違いも、画面設計の違いも、1つのアプリケーション開発で対応！

2-1.FireUIによるマルチデバイス開発機能

2-1.FireUIによるマルチデバイス開発機能

- マルチデバイスデザイナー (FireUI)

XE7の開発環境



2-1.FireUIによるマルチデバイス開発機能

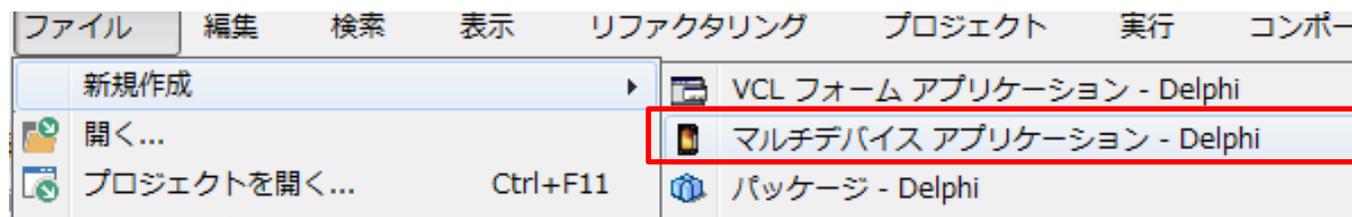
• FireUIの機能① プロジェクト

XE5でのプロジェクト作成メニュー



デスクトップアプリと
モバイルアプリは
別々の開発

XE7でのプロジェクト作成メニュー

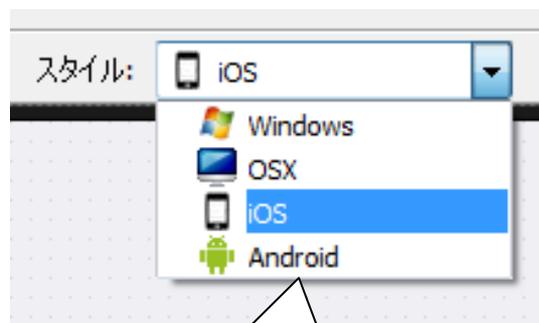


マルチデバイスアプリと
して統合開発

2-1.FireUIによるマルチデバイス開発機能

- FireUIの機能② スタイル

スタイルを切り替えるだけで
OS独自の画面表示で設計可能



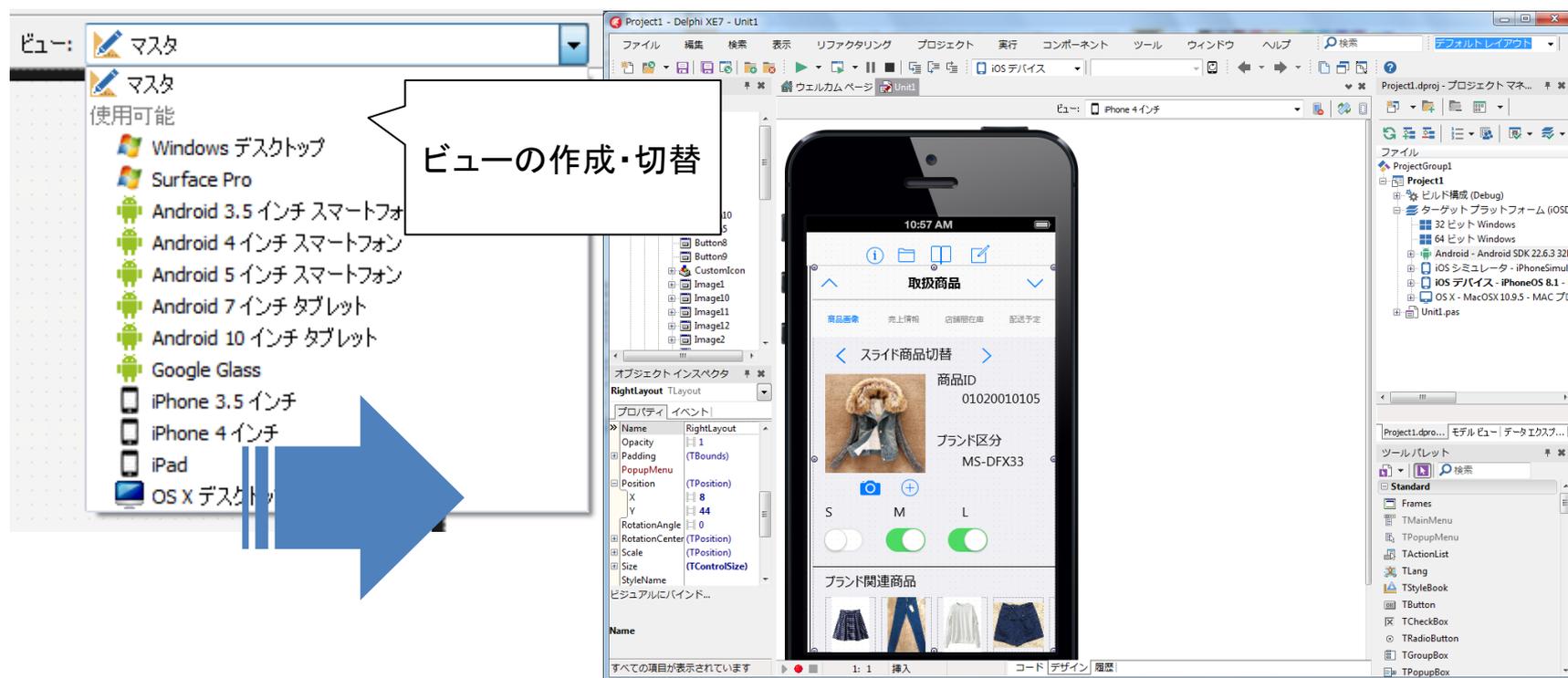
スタイルの切替



2-1.FireUIによるマルチデバイス開発機能

• FireUIの機能③ ビュー

デバイス毎にビューを作成することができるため、
同じプログラムでデバイス毎に画面を調整・変更することが可能

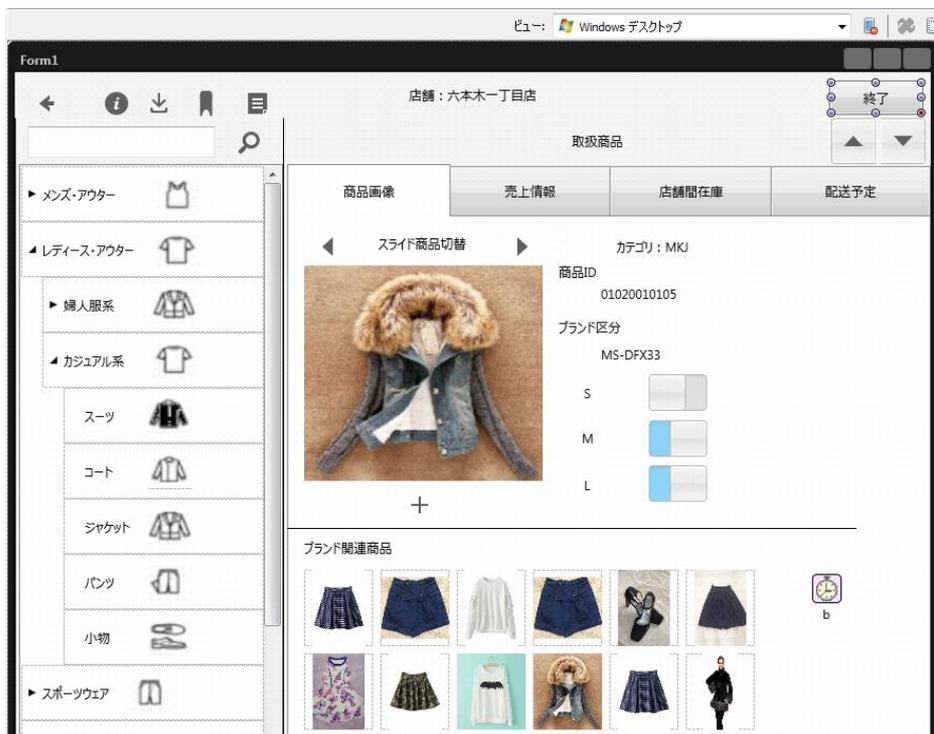


2-1.FireUIによるマルチデバイス開発機能

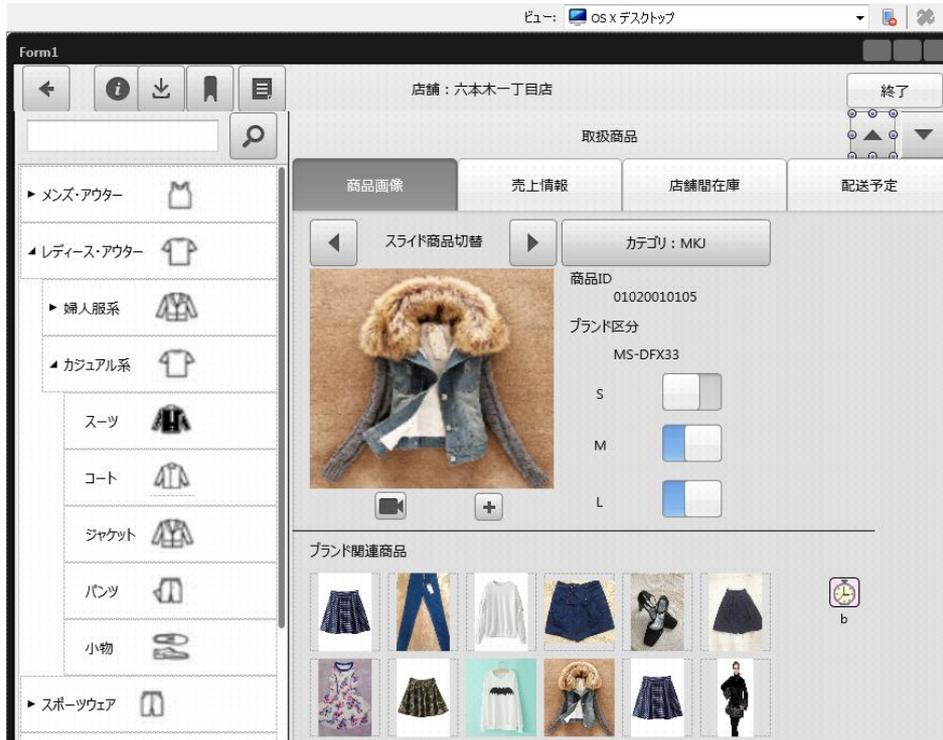
• FireUIの機能③ ビュー

OS毎の画面設計例

Windows



OSX(Mac)

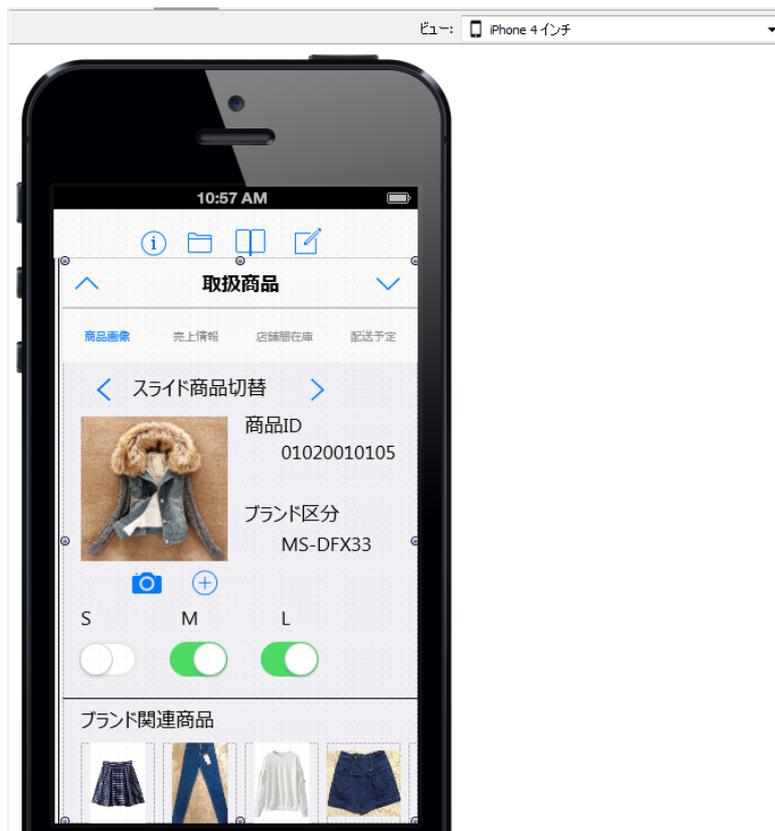


2-1.FireUIによるマルチデバイス開発機能

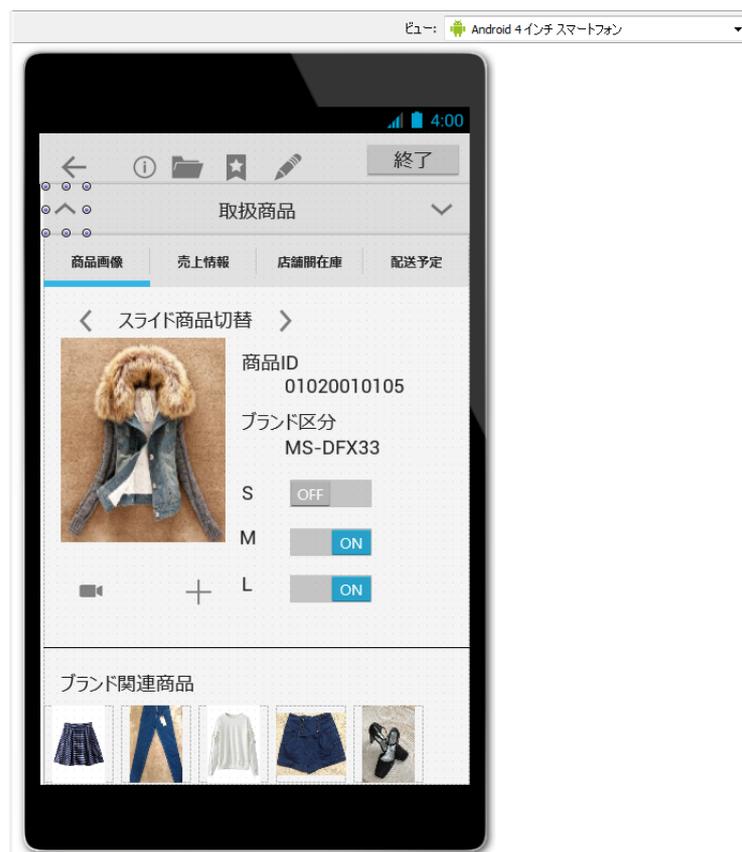
- FireUIの機能③ ビュー

OS毎の画面設計例

iOS (iPhone)



Android (スマートフォン)



2-1.FireUIによるマルチデバイス開発機能

- FireUIの機能③ ビュー
OS毎の画面設計例

iOS (iPad)



Android (タブレット)



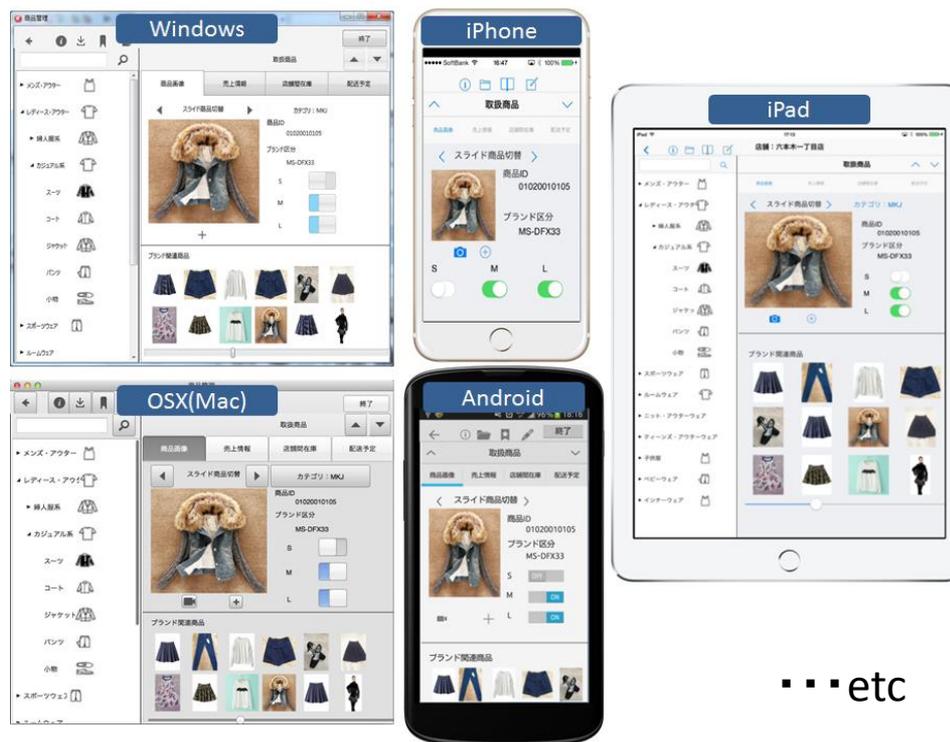
2-1.FireUIによるマルチデバイス開発機能

• FireUIの機能④ コンパイル

作成したプログラムはプロジェクトマネージャで対象のOSを選択して、デバイス毎のビューと一緒に自動でコンパイルが可能



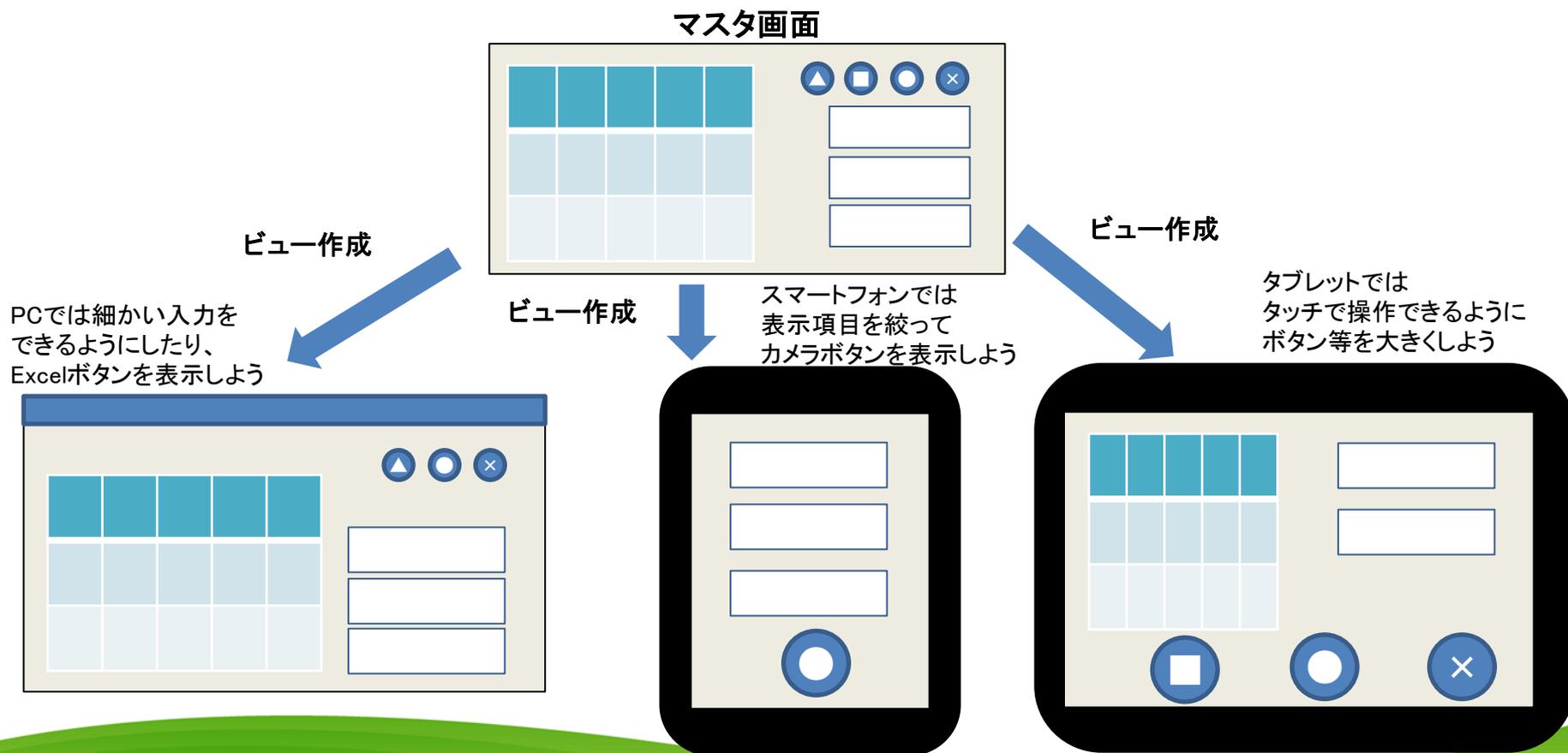
OS毎にコンパイル



2-1.FireUIによるマルチデバイス開発機能

- FireUIの仕組み

マスタ画面をベースとして継承し、OSやデバイス毎のビュー(画面)を1つのアプリケーションで開発できます。



2-1.FireUIによるマルチデバイス開発機能

- デバイスにおける画面設計差異のポイント

例) デバイスにおける設計の違い

画面に表示できる項目数が異なる

マウスとタッチでは操作性の考慮が異なる

デバイスによって項目位置の優先が異なる

PCアプリケーション

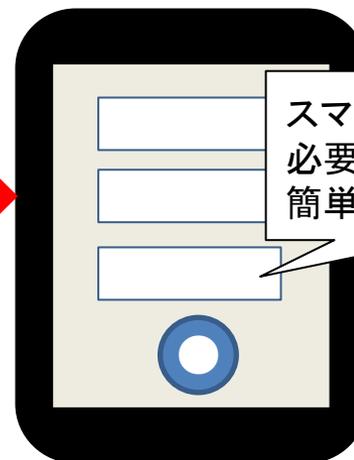
PCでは、多くの情報を表示しながら細かく操作する



同じ設計では
使いづらい

スマートフォンアプリケーション

スマートフォンでは
必要な情報を表示して
簡単に操作する



2-1.FireUIによるマルチデバイス開発機能

- OSにおける画面設計差異のポイント



例) OSにおける設計の違い

iOSでは「戻るボタン」が必要、「閉じる」ボタンは使用できない

Androidでは「戻るボタン」は不要、「閉じる」ボタンは使用できる

OS・デバイスの違いを把握した画面設計は非常に重要です。
FireUIはこうした違いを簡単に画面設計で調整できます。

2-1.FireUIによるマルチデバイス開発機能



- FireUIによるマルチデバイス開発例
1つのプログラムでPC向けとスマートフォン向けに作成するアプリケーション

PCアプリケーション

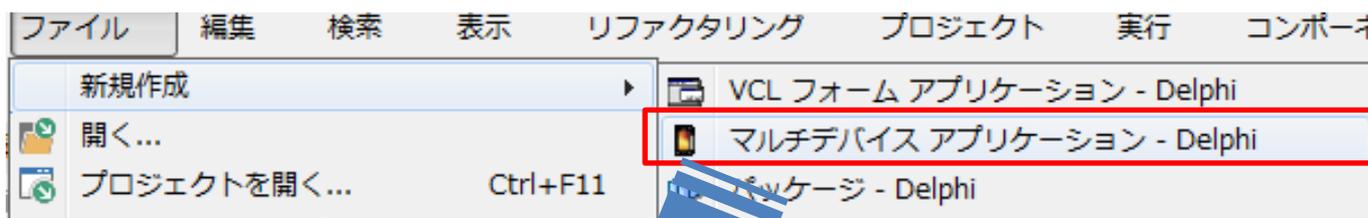
The screenshot shows a PC application window titled 'Form1'. It displays a '商品一覧' (Product List) on the left and a '商品詳細' (Product Details) form on the right. The product list includes items like 'いーろ・は・す', 'ボルヴィック', 'エビアン', 'クリスタルガイザー', 'おいしい水', and 'コントレックス'. The details form shows fields for '商品ID', '商品名', '在庫数', and '商品画像' (Product Image). The '商品ID' is 0000000001, '商品名' is 'いーろ・は・す', and '在庫数' is 100. A product image of a water bottle is shown. Buttons for '更新' (Update) and '閉じる' (Close) are at the bottom.

スマートフォンアプリケーション(iPhone)

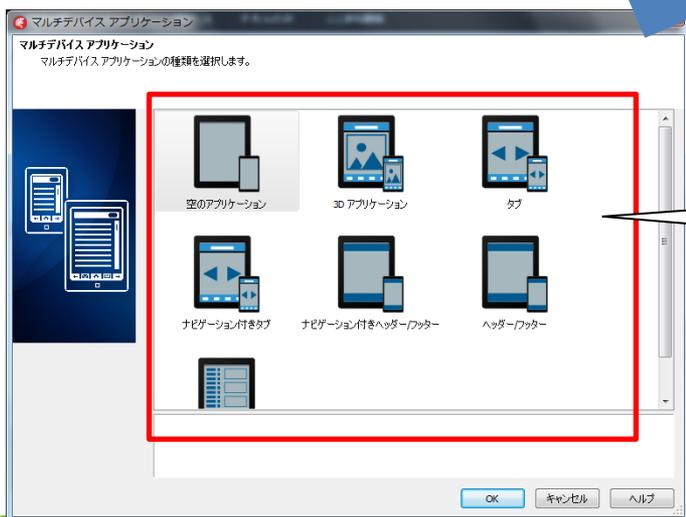


2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順①
プロジェクトを作成する



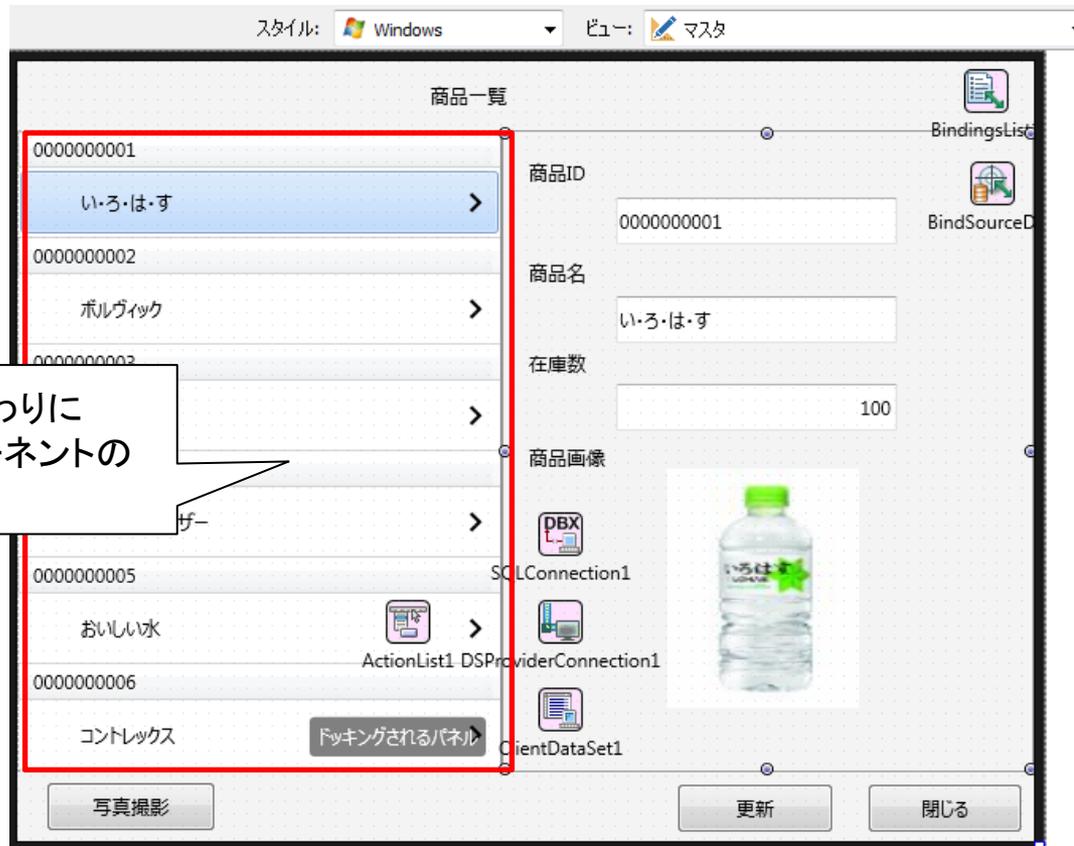
マルチデバイスアプリとして新規作成する



テンプレートがいくつか用意されている

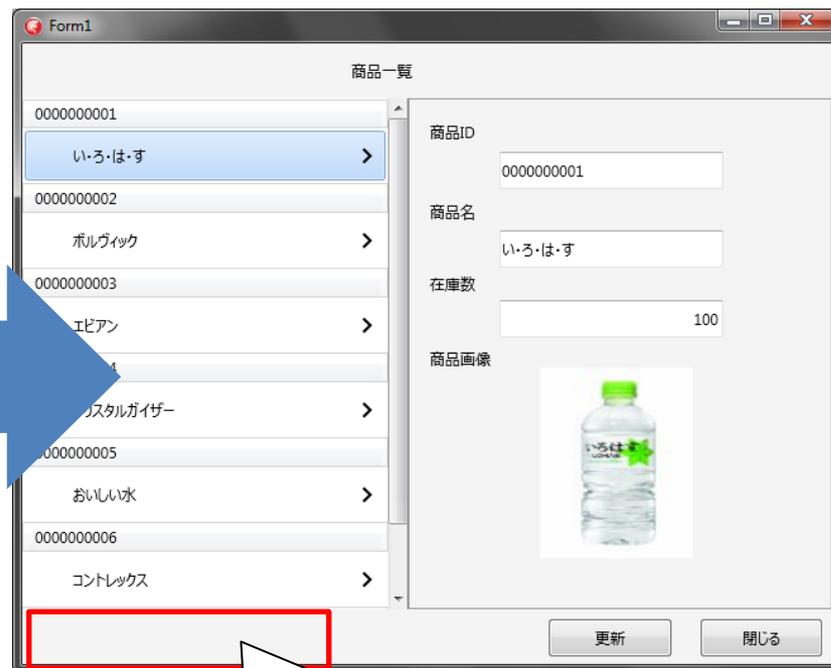
2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順②
マスタ画面を作成する



2-1.FireUIによるマルチデバイス開発機能

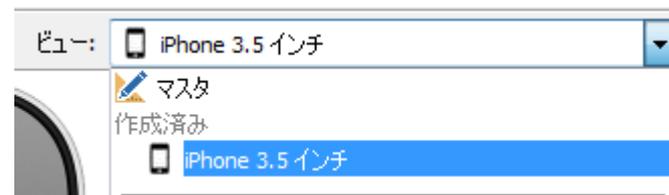
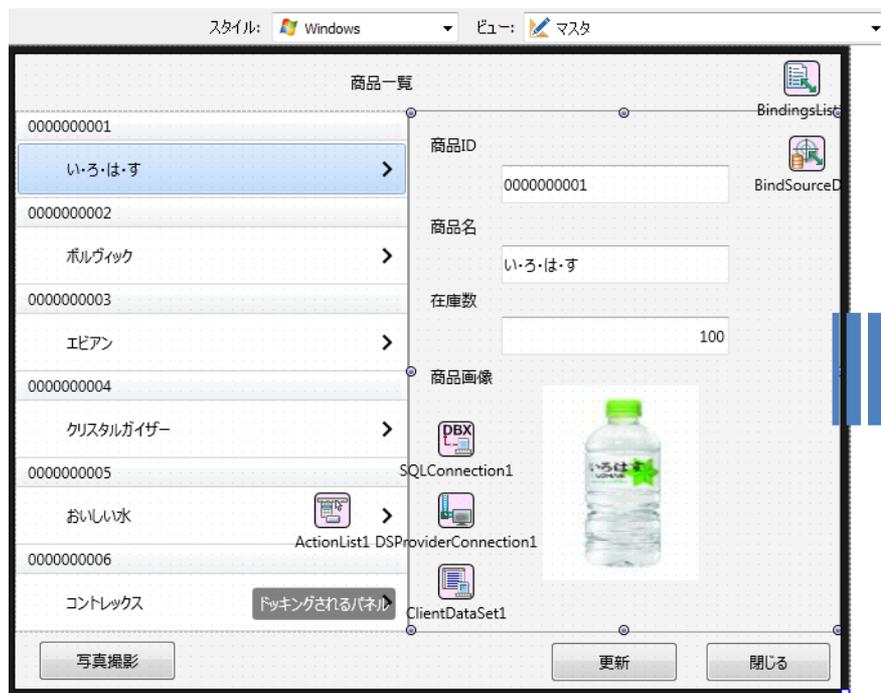
- FireUIによるマルチデバイス開発手順③
Windowsで実行する



Windowsでは扱えない
カメラボタンは自動で非表示
になる

2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順④
FireUIで iPhone向けに画面を変更する



2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順⑤
iPhone向けに画面を調整する

画面に項目が多すぎる

「閉じる」ボタンも不要

オブジェクトインスペクタ

MultiView1 TMultiView

プロパティ イベント

HelpType htContext
HitTest True
LiveBindings LiveBindings
Locked False
MasterButton
Mode Drawer
Name

ポイント！

TMultiViewの
ModeプロパティをDrawer
で設定(リストが折り畳まれる)

すべての項目が表示されています

画面設計をiPhone用に調整

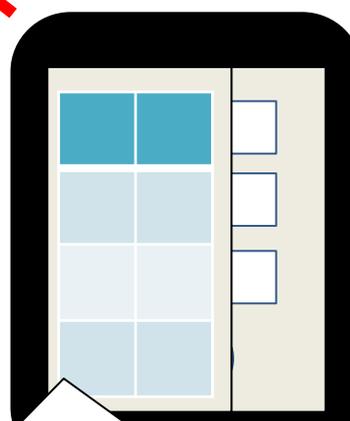
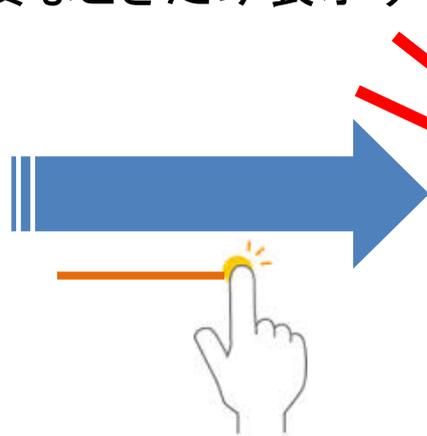
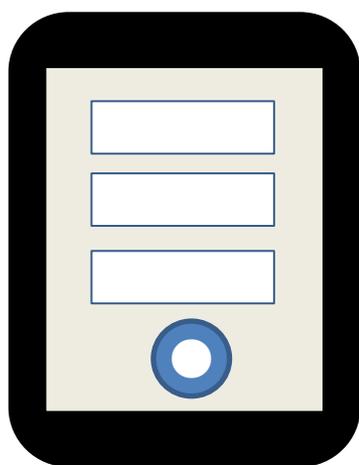
「閉じる」ボタンは
VisibleをFalseに設定

2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順⑤
iPhone向けに画面を調整する

TMultiViewコンポーネントとは？

スライド操作などで必要なときだけ表示する領域を簡単に実装可能



リストが表示される



Windows8のスライド表示させるチャームメニューなどの動作が代表的です。

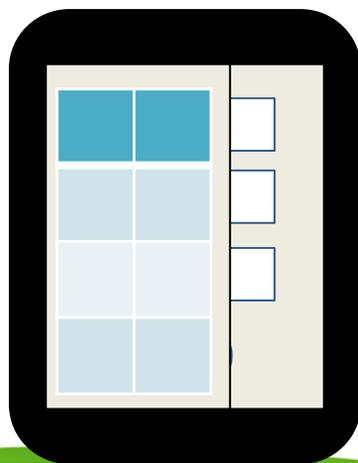
2-1.FireUIによるマルチデバイス開発機能

補足

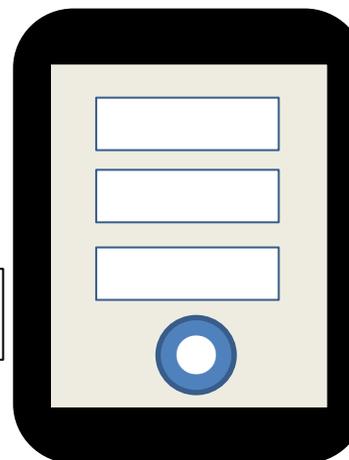
- FireUIによるマルチデバイス開発手順⑥
iPhone向けに画面を調整する

OnItemClickイベント(リストでのマルチビュー制御)

```
procedure TForm1.ListView1ItemClick(const Sender: TObject;  
  const AItem: TListViewItem);  
begin  
  //リスト選択されたらマルチビューを閉じる  
  MultiView1.HideMaster;  
end;
```

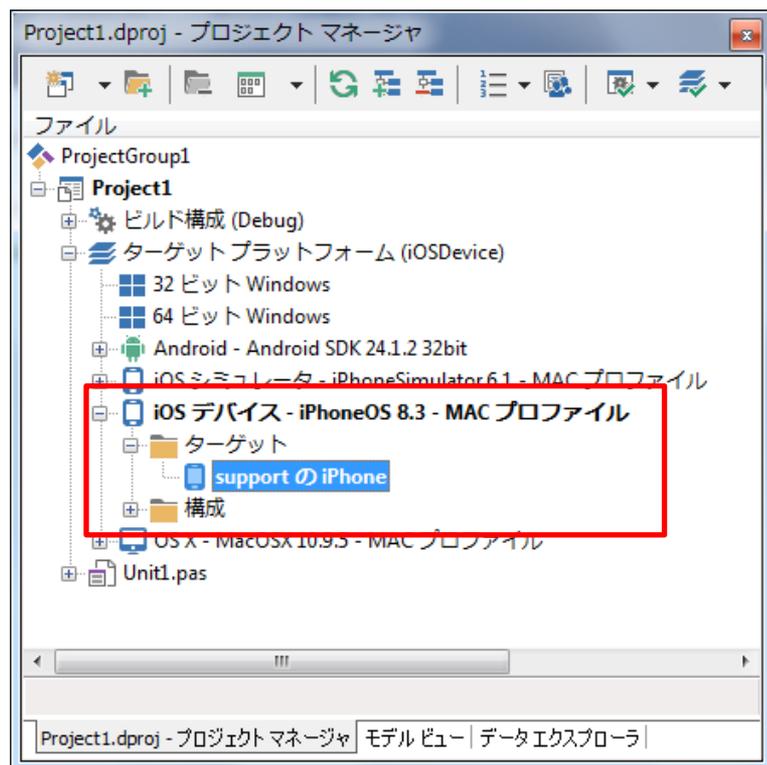


リストをタッチすると
リストを閉じる



2-1.FireUIによるマルチデバイス開発機能

- FireUIによるマルチデバイス開発手順⑦
iPhone向けにコンパイルする



コンパイル



マルチビューで
画面設計も解決

2-1.FireUIによるマルチデバイス開発機能

- アプリケーションの完成

1つのプログラムからFireUIでデバイス毎に画面を調整して最適なアプリが完成！



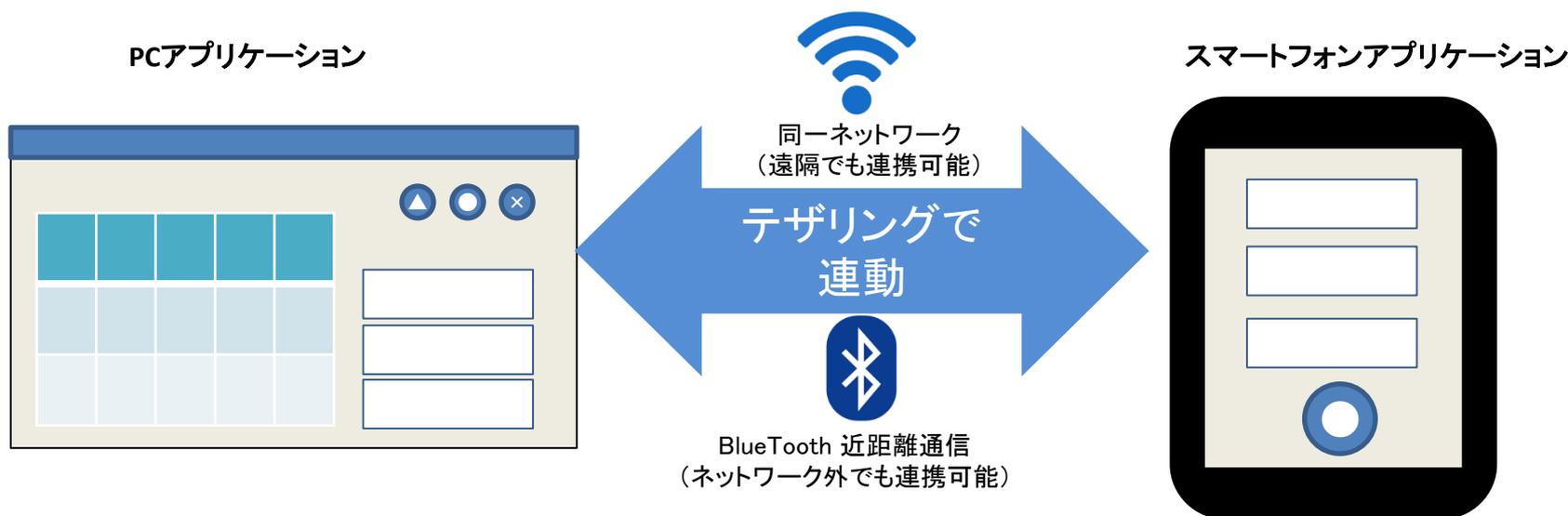
もちろんAndroidやタブレット向けの画面を作成することも同様に可能です。

2-2.アップデートザリングによるアプリ連携機能

2-2. アップテザリングによるアプリ連携機能

• アップテザリングとは？

同じネットワークやBluetooth上のアプリケーション間でデータや処理を共有して連携することができるXE7の新機能
(ルータとしての通信テザリングとは少し異なります)



アップテザリングの前提

同じネットワークに両機器つながっているか、
または両機器がBluetoothで通信する必要があります。

2-2.アップテザリングによるアプリ連携機能

- アップテザリング連携例

例えば2-1.で作成したWindowsアプリケーションは、
せっかく画像を更新する機能があるのに、カメラ機能が使えない事が残念



PCでは通常
写真撮影機能
が使えない！

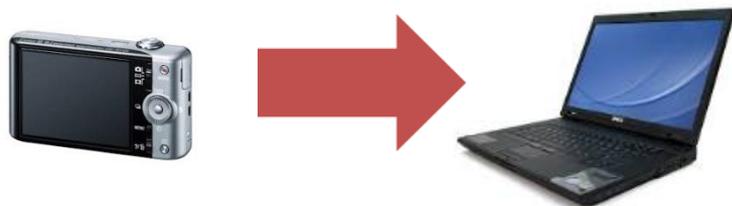
2-2.アップテザリングによるアプリ連携機能

• アップテザリング連携例

これまで写真撮影やバーコード読取を行う業務では専用機器を用意して連携する方法が一般的

商品写真を登録する

デジタルカメラで撮影して、SDカードをPCにアップして、ようやくサーバへ登録・更新



バーコードを読み取って登録する

専用のバーコードリーダーやPOSを用意して、PCのアプリケーションから登録・更新



スマートデバイスがあれば代用実現



写真撮影やバーコード読取結果をスマートフォンアプリからPCアプリに送信して登録・更新
(アプリで自動化が可能)

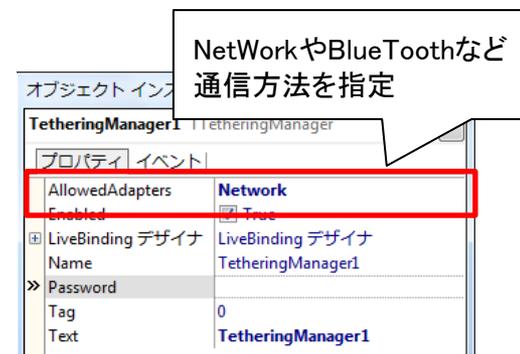
2-2. アップテザリングによるアプリ連携機能

- アップテザリング用のコンポーネント
アップテザリングを使用する場合には、通信をする両方のアプリケーションに TTetheringMangerとTTetheringAppProfileコンポーネントを配置

TTetheringManagerコンポーネント



ネットワーク上でテザリング
するための接続等の管理



TTetheringAppProfileコンポーネント



テザリングで接続した
アプリケーション間で共有する
リソースの制御

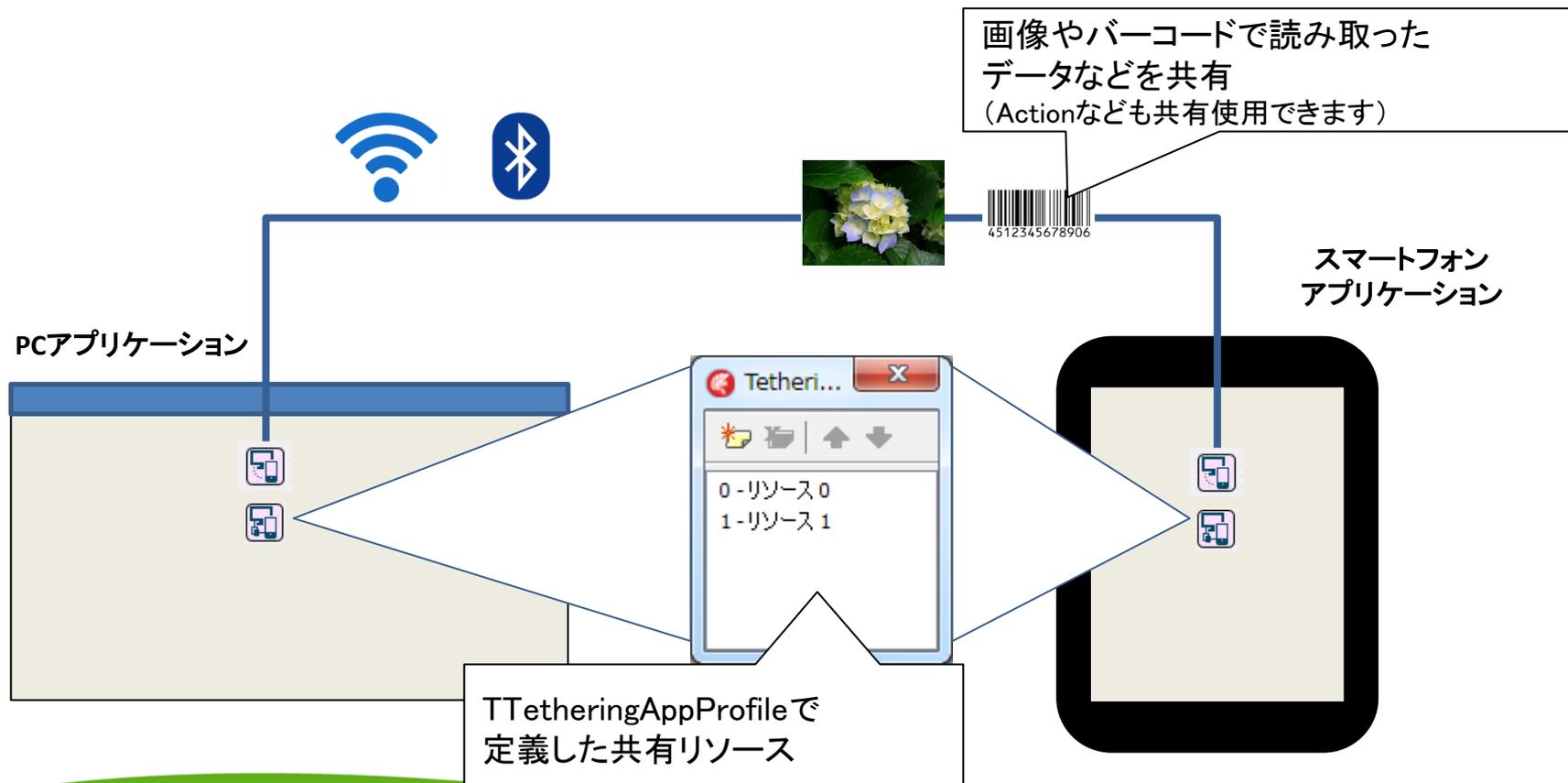


2-2. アップテザリングによるアプリ連携機能

- アップテザリング用のコンポーネントの使い方

TTetheringMangerで接続を行い、

TTetheringAppProfileで共有リソースを送受信



2-2. アップテザリングによるアプリ連携機能

- PCとスマートフォンのアプリケーション連携例
作成するアプリケーション



PCアプリケーション

商品一覧

0000000001	いろは・はす
0000000002	ボルグイック
0000000003	エビアン
0000000004	クリスタルガイザー
0000000005	おいしい水
0000000006	コントレックス

商品ID: 4902102091
商品名: いろは・はす
在庫数: 100
商品画像:

更新 閉じる

スマートフォンアプリケーション (iPhone)

読取バーコード
送信

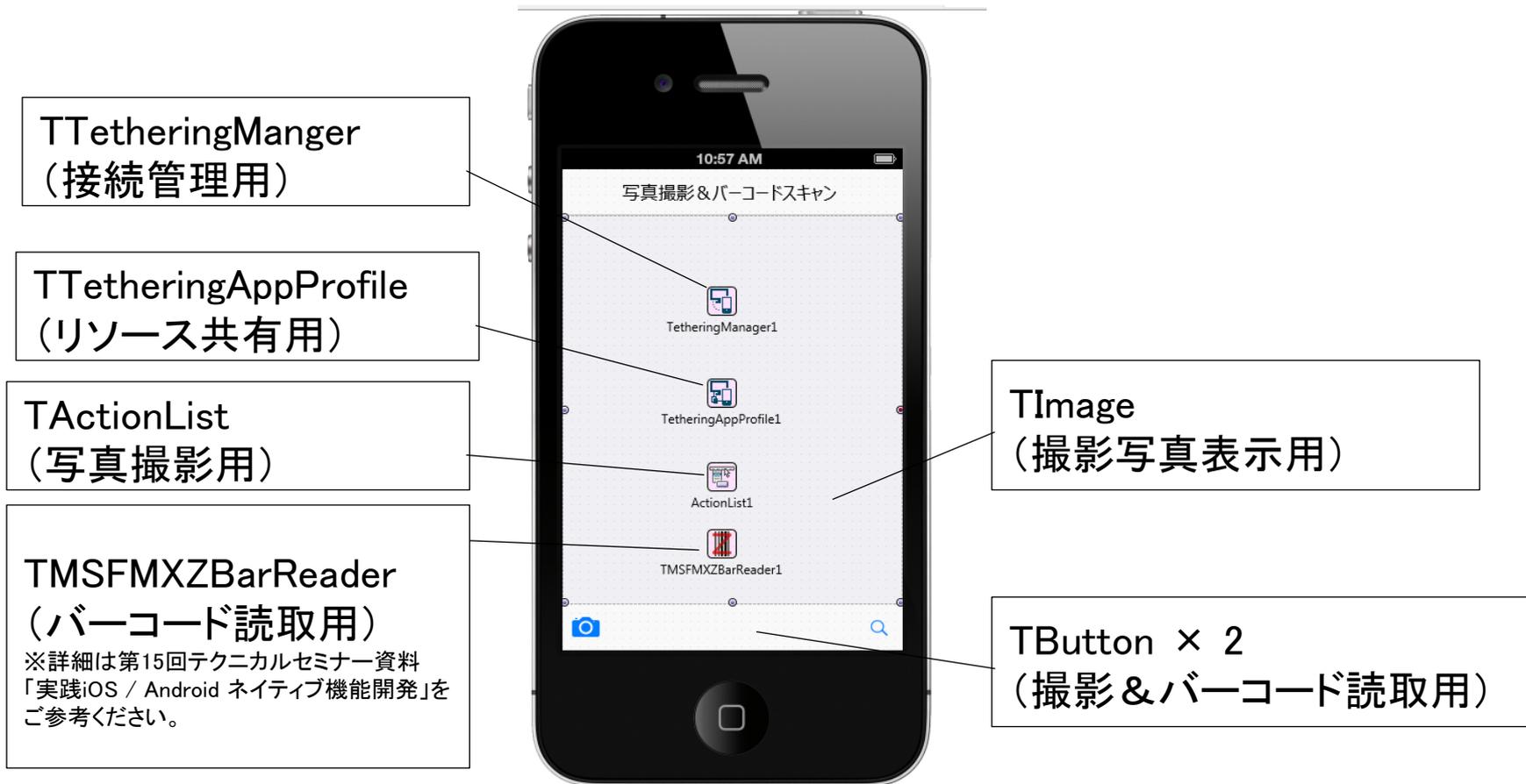


撮影写真
送信



2-2. アップテザリングによるアプリ連携機能

- PCとスマートフォンのアプリケーション連携例
iPhoneアプリ側画面設計



2-2.アップテザリングによるアプリ連携機能

- iPhoneアプリ側開発手順①
TTetheringAppProfileの設定

通信グループ名

リソースを2つ追加

ポイント!

写真用に
Nameプロパティ: Camera
ResTypeプロパティ: Stream

送信側のKindプロパティはShared

バーコード用に
Nameプロパティ: Barcode
ResTypeプロパティ: Data

ResTypeプロパティはDataかStreamを選択
Data: 文字などの送信
Stream: 画像などの送信

ダブルクリック

2-2. アップテザリングによるアプリ連携機能

- iPhoneアプリ側開発手順②
Actionの設定(写真撮影ボタン)



オブジェクトインスペクタ
Button1 TButton
プロパティ イベント |
Action
Align アクションの新規作成
Anchors 標準アクションの新規作成
AutoTransp True

(カテゴリなし)(S) ▶
編集(T) ▶
ウィンドウ(U) ▶
ファイル(V) ▶
タブ(W) ▶
表示(X) ▶
電話ダイヤラ(Y) ▶
メディア ライブラリ(Z) ▶
LiveBindings ▶

TTakePhotoFromLibraryAction
TTakePhotoFromCameraAction
TShowShareSheetAction

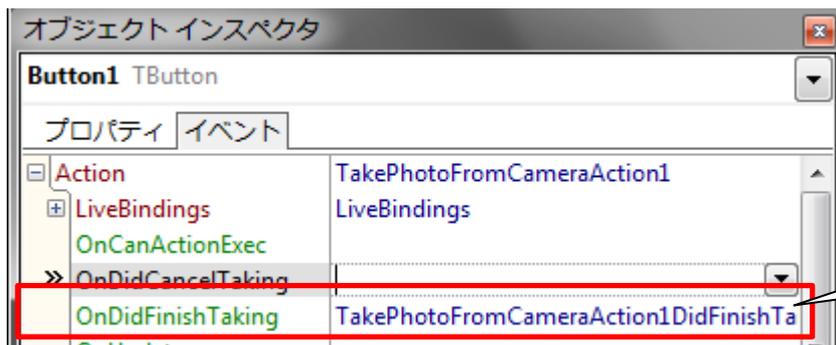
ButtonのActionプロパティで
標準アクションの新規追加

TTakePhotoFromCameraAction
を選択

2-2. アップテザリングによるアプリ連携機能

- iPhoneアプリ側開発手順③

Actionのイベントにプログラムを実装(写真撮影ボタン) 



OnDidFinishTakingイベントを作成

OnDidFinishTakingイベント(撮影写真を送信)

```
procedure TForm1.TakePhotoFromCameraAction1DidFinishTaking(Image: TBitmap);
var
  FStream: TMemoryStream;
begin
  FStream := TMemoryStream.Create; //写真用のStreamを作成
  image.SaveToStream(FStream); //撮影写真をStreamに格納
  TetheringAppProfile1.Resources.Items[0].Value := FStream; //共有リソースに送信
  Image1.Bitmap.Assign(Image); //画面に写真を表示
end;
```

2-2. アップテザリングによるアプリ連携機能

- iPhoneアプリ側開発手順④

バーコード撮影用のイベントにプログラムを実装(バーコード撮影ボタン) 

OnClickイベント(バーコード撮影起動)

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
    TMSFMXZBarReader1.Show; //バーコード撮影を起動  
end;
```

バーコード撮影用のイベントにプログラムを実装(TMSFMXZBarReader)

OnGetResultイベント(取得バーコード送信)

```
procedure TForm1.TMSFMXZBarReader1GetResult(Sender: TObject; AResult: string);  
Begin  
    //読み取ったバーコード値を共有リソースに送信  
    TetheringAppProfile1.Resources.Items[1].Value := AResult;  
end;
```

2-2.アップテザリングによるアプリ連携機能

- iPhoneアプリ側開発手順⑤

画面起動時のイベントにプログラムを実装

OnCreateイベント(テザリングで接続)

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    //起動時にテザリング接続を行う
    TetheringManager1.AutoConnect();
end;
```

2-2.アップテザリングによるアプリ連携機能

- PCとスマートフォンのアプリケーション連携例
PCアプリ側画面設計(2-1の画面に追加)



2-2.アップテザリングによるアプリ連携機能

• PCアプリ側開発手順①

TTetheringAppProfileの設定
(スマートフォン側と設定を合わせる)

通信グループ名

リソースを2つ追加

ポイント!

写真用に
Kindプロパティ: Mirror
Nameプロパティ: Camera
ResTypeプロパティ: Stream

受信側のKindプロパティはMirror

バーコード用に
Kindプロパティ: Mirror
Nameプロパティ: BarCode
ResTypeプロパティ: Data

ダブルクリック

2-2. アップテザリングによるアプリ連携機能

- PCアプリ側開発手順②

写真撮影リソースのイベントにプログラム実装

OnResourceReceivedイベント(撮影写真を受信)

```
procedure TForm1.TetheringAppProfile1Resources0ResourceReceived  
  (const Sender: TObject; const AResource: TRemoteResource);  
begin  
  AResource.Value.AsStream.Position := 0; //Streamのポジション  
  Image1.Bitmap.LoadFromStream(AResource.Value.AsStream); //画面に受信画像を設定  
  Image1.Repaint; //再描画  
end;
```

2-2.アップテザリングによるアプリ連携機能

- PCアプリ側開発手順③

バーコードリソースのイベントにプログラム実装

OnResourceReceivedイベント(取得バーコードを受信)

```
procedure TForm1.TetheringAppProfile1Resources1ResourceReceived  
  (const Sender: TObject; const AResource: TRemoteResource);  
begin  
  Edit1.Text := AResource.Value.AsString; //画面に受信値を設定  
end;
```

2-2. アップテザリングによるアプリ連携機能

- PCとスマートフォンのアプリケーション連携例
それぞれコンパイルを行い、アプリケーション連携が完成

PCアプリケーション

The screenshot shows a PC application window titled 'Form1' with a '商品一覧' (Product List) on the left and a '商品ID' (Product ID) field on the right. The product list includes items like 'いろいろは・す', 'ポリック', 'エビアン', 'クリスタルガイザー', 'おいしい水', and 'コントレックス'. The '商品ID' field contains '4902102091', the '商品名' (Product Name) is 'いろいろは・す', and the '在庫数' (Inventory) is '100'. A '商品画像' (Product Image) field shows a photo of a blue and white beverage bottle. Buttons for '更新' (Update) and '閉じる' (Close) are at the bottom.

スマートフォンアプリケーション (iPhone)

読取バーコード
送信



撮影写真
送信



こうしたアプリ連携はAndroidはもちろん、スマートデバイス間、PC間のアプリケーションでも活用することができます。

3.まとめ

- 新バージョンXE7ではFireUIによりOS毎だけでなく、デバイス毎の画面設計にも対応したマルチデバイス開発を実現
- マルチデバイス開発では、OS・デバイス毎の特徴をおさえてアプリケーションを設計することが重要
- アップテザリング機能を使うとネットワークやBlueToothを通じてアプリケーション間でデータや処理を共有できるため、スマートデバイスを専用機器の代用などに利用可能

ご清聴ありがとうございました