【セッションNo. 3】

Delphi/400技術セッション

実践！iOS / Android ネイティブ機能開発
～バーコード読み取り、署名、オフライン処理～

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.スマートデバイスのネイティブ機能

2.Delphi/400ネイティブ機能の開発テクニック
- 2-1. カメラを使ったバーコード読み取り機能
- 2-2. タッチ操作を使った画面署名機能
- 2-3. オフラインでのローカルデータ保存

3.まとめ

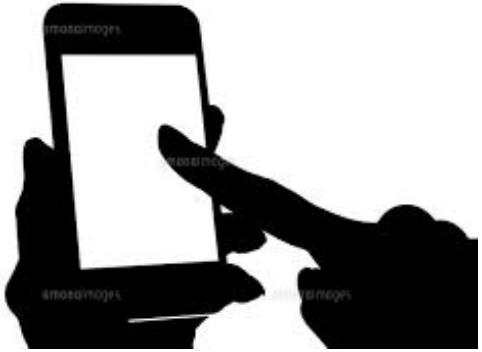
1.スマートデバイスのネイティブ機能

1.スマートデバイスのネイティブ機能

- スマートデバイスが持つデバイスネイティブ機能
  スマートデバイスはPCと違い、デバイス自体が様々な
  機能（ネイティブ機能）を搭載しています。
  カメラ撮影、タッチ操作、GPSといった機能が代表的ですが、
  Delphi/400ではこうしたネイティブ機能を活用したアプリケーションを
  簡単に開発することができます。

カメラ撮影

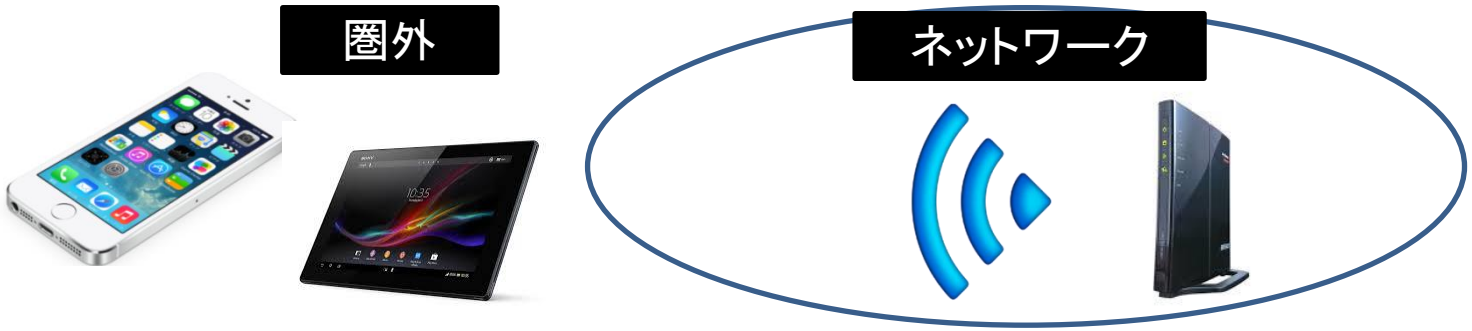
タッチ操作



GPS

1.スマートデバイスのネイティブ機能

- スマートデバイスが持つデバイスネイティブ機能
  またスマートデバイスはPCと違い、持ち運びに優れた携帯性が特徴です。
  しかし、稼働環境としては移動を前提とすることが多く、
  常にネットワークに接続されている保証がないことも大きなポイントです。



Webアプリケーションであれば、オフラインでWebサーバに
アクセスできなくなりますが、ネイティブアプリケーションであれば、
デバイスの内部ファイルにデータを保存することも可能です。

1.スマートデバイスのネイティブ機能

- スマートデバイスが持つデバイスネイティブ機能
  本セッションでは、こうしたデバイスネイティブ機能を応用した
  実践で役立つプログラミングテクニックをご紹介します。

カメラ	• カメラを使ったバーコード読込
タッチ操作	• タッチ操作による画面署名
内部ファイル	• オフラインでのローカルデータ保存

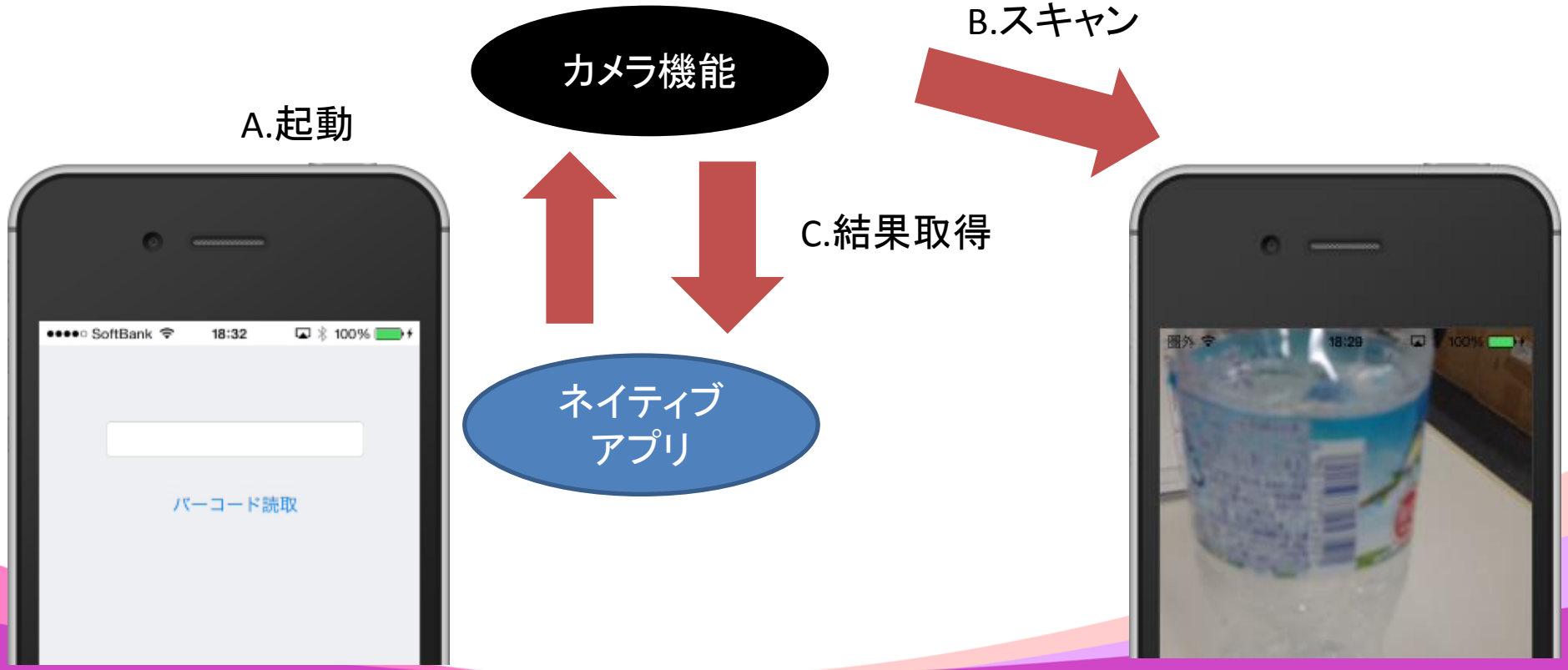
- GPS位置情報を使った地図連携は、XE5の標準サンプルに付属しています。

2.Delphi/400 ネイティブ機能の開発テクニック

2-1. カメラを使ったバーコード読み取り機能

2-1. カメラを使ったバーコード読み取り機能

- カメラ機能を使ったバーコード読み取りの仕組み
  スマートデバイスではカメラ機能を利用して、
  バーコードやQRコードを読み取り、値を取得します。
  （PCのように、バーコードリーダーの外部接続は不要）



2-1. カメラを使ったバーコード読み取り機能

- バーコード読み取り機能の実装に便利なコンポーネント

TMSSoftWare社のバーコード読み取りコンポーネント（無償）
【ZBarSDK】 ※iOS専用
http://www.tmssoftware.com/site/blog.asp?post=280

tmssoftware.com

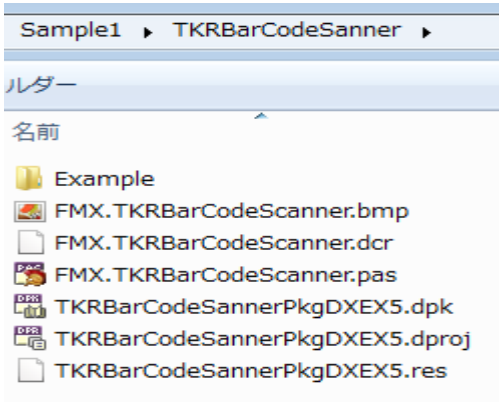
ただしZBarSDKコンポーネントはiOS専用です。
Androidで使用する場合は、これをカスタマイズした
フリーソースとして公開されているTKRBarCodeSannerコンポーネントが
便利です。

【TKRBarCodeSanner】 ※iOS / Android可能
（iOS使用時はZBarSDKもインストールが必要 ）
http://www.file-upload.net/download-8601754/TKRBarCodeSanner.zip.html

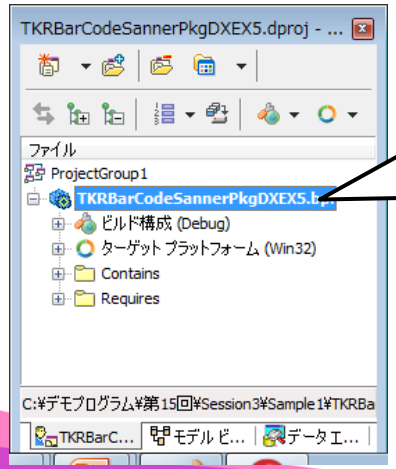
今回はこのTKRBarCodeSannerコンポーネントを使用します。

2-1. カメラを使ったバーコード読み取り機能

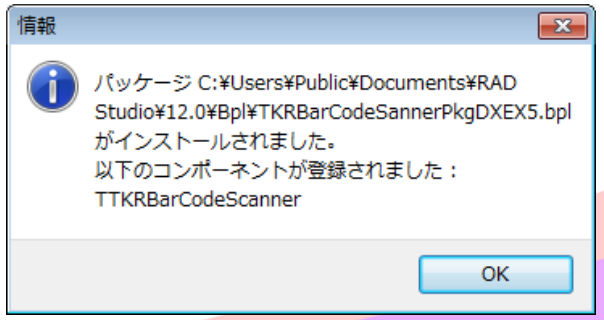
- TKRBarCodeSannerコンポーネントのインストール①
  TKRBarCodeSanner.zipをダウンロードして展開します。



[ファイル|プロジェクトを開く]よりTKRBarCodeSannerPkgDXEX5.dpkを開きます。

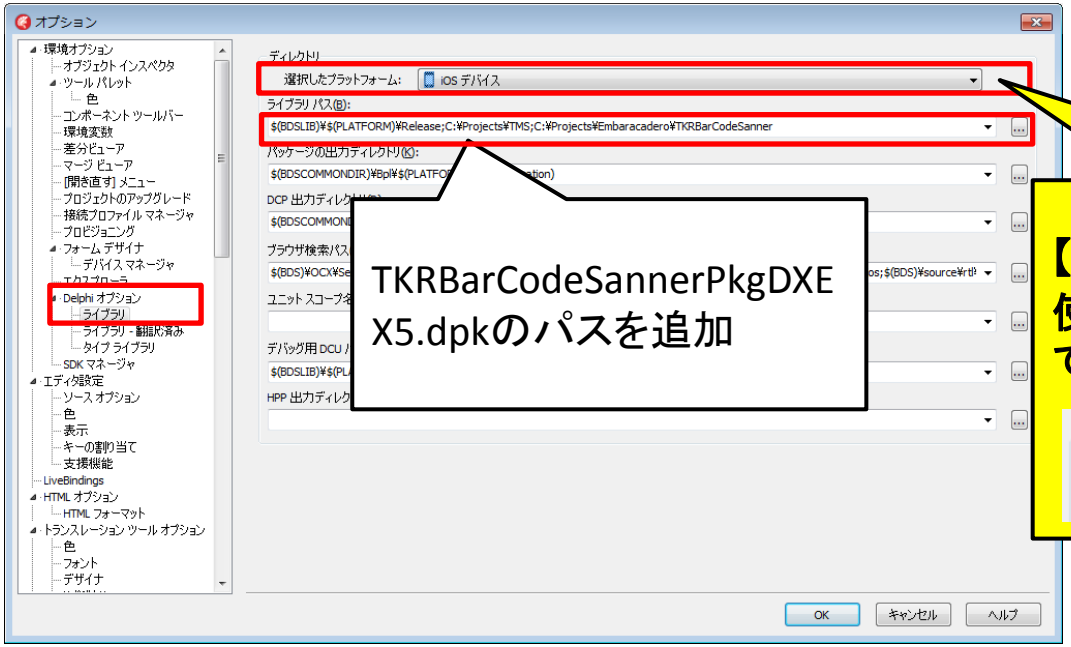


プロジェクトマネージャで
右クリックからコンパイル
とインストールを実行



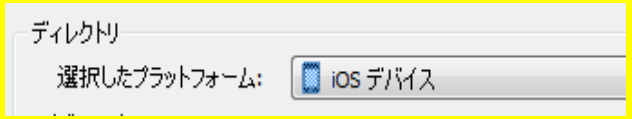
2-1. カメラを使ったバーコード読み取り機能

- TKRBarCodeSannerコンポーネントのインストール②
  [ツール|オプション]のライブラリでライブラリパスに
  TKRBarCodeSannerPkgDXEX5.dpkを開いたパスを追加します。

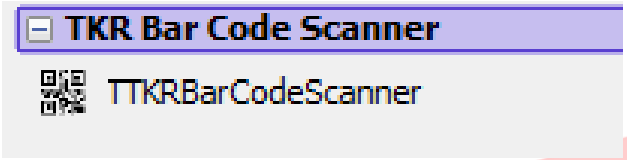


TKRBarCodeSannerPkgDXE
X5.dpkのパスを追加

【ポイント】
使用するプラットフォームを選択しておく必要があります



コンポーネントの登録が完了！



2-1. カメラを使ったバーコード読み取り機能

- バーコード機能の実装手順①
  フォームに次のコンポーネントを配置
  TKRBarCodeSanner 、TEdit、TButton

2-1. カメラを使ったバーコード読み取り機能

- バーコード機能の実装手順②

TButtonのクリックイベントにプログラムを実装

A.カメラ起動　B.スキャン

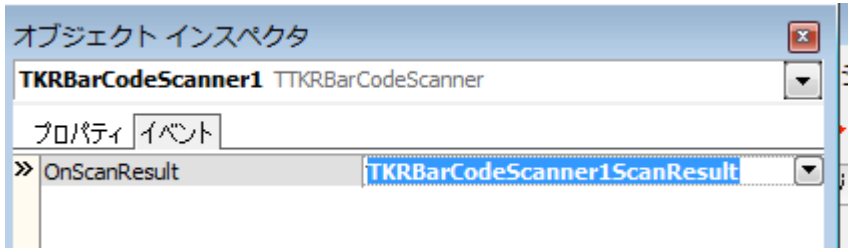
OnClick処理（バーコードスキャン）

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  TKRBarCodeScanner1.Scan;    //バーコードスキャンを実行
end;
```

2-1. カメラを使ったバーコード読み取り機能

- バーコード機能の実装手順③

TKRBarCodeSannerのスキャン結果イベントにプログラムを実装



C.結果取得

OnScanResult処理（スキャン結果）

```
procedure TForm1.TKRBarCodeScanner1ScanResult(Sender: TObject; AResult: string);
begin
  Edit1.Text := AResult;      //読み取ったコードをEditにセット
end;
```

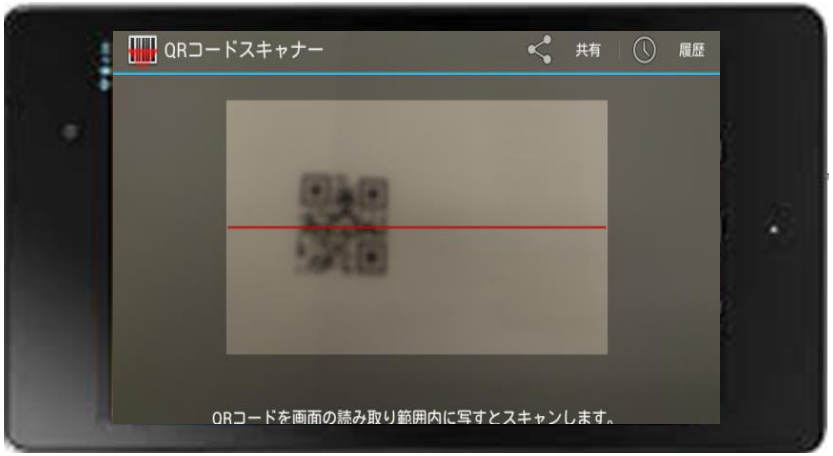
2-1. カメラを使ったバーコード読み取り機能

- バーコード機能の実行

A.カメラ起動　　B.スキャン　　C.結果取得

2-1. カメラを使ったバーコード読み取り機能

- 補足
  もちろんAndroid での実行やQRコードの読み取りも可能です。



【QRコード】
マトリクス型2次元コードで、Quick Responseコードという名の通り、高速読み取りを重視した2次元コードです。
情報量が多いのでURLなどに使われたりもします。

2.スマートデバイスアプリケーションの種類

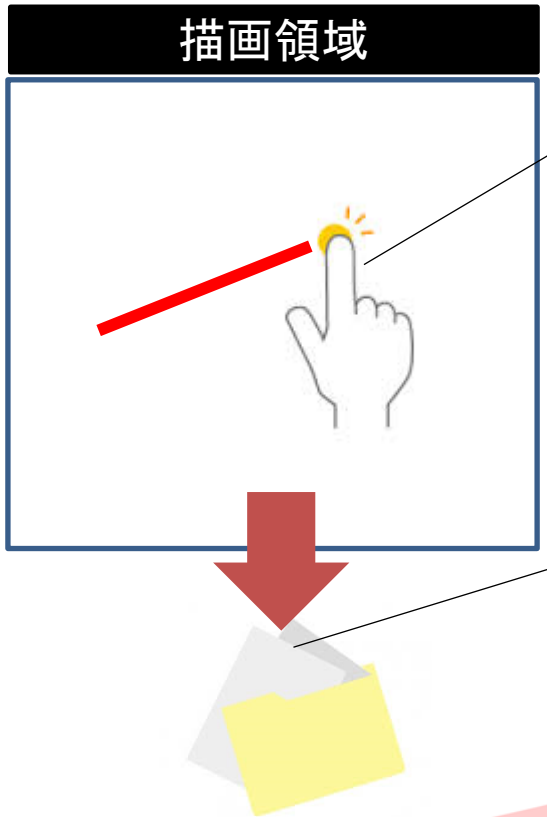
- バーコード機能の応用
  取得したバーコード値を使えば、バーコードとIBM i のデータの連携が可能。

2.Delphi/400 ネイティブ機能の開発テクニック

2-2. タッチ操作を使った画面署名機能

2-2. タッチ操作を使った画面署名機能

- タッチ操作を利用した署名の仕組み
  スマートデバイスではディスプレイのタッチ操作の機能を利用して、
  署名を行う場合、描画領域を用意してタッチの軌跡で表現します。



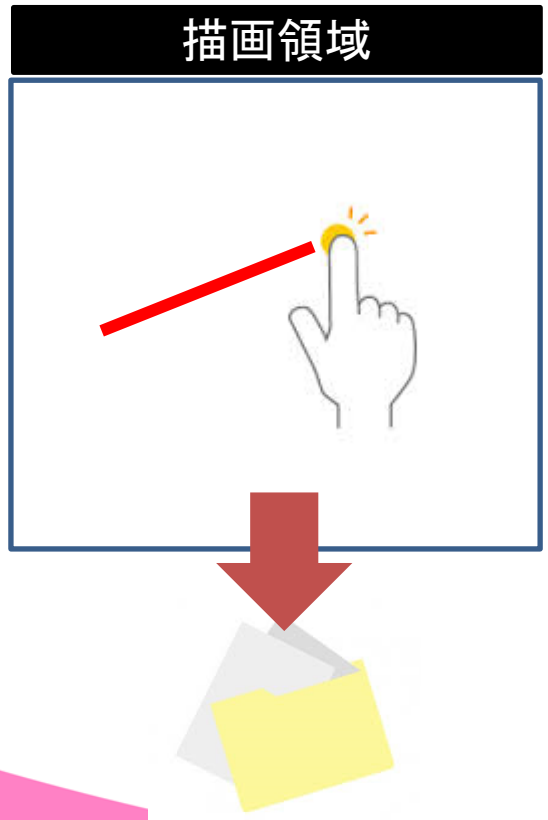
①タッチ操作でマウスのように軌跡を表現

マウスとしてハンドリング可能

②署名が終わったらスクリーンショットで画像に保存

2-2. タッチ操作を使った画面署名機能

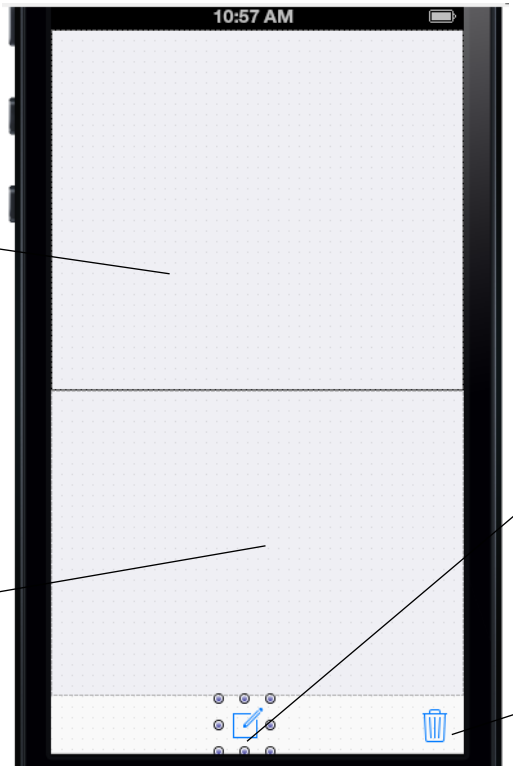
- タッチ操作を利用した署名機能実装のポイント
  機能の実装としては、軌跡情報制御、描画制御、初期・終了制御、
  タッチ操作制御、署名画像生成で構成を組み立てます。



A.軌跡情報制御	座標を管理する
B.描画制御	座標から描画する
C.初期・終了制御	変数を生成・破棄する
D.タッチ操作制御	操作で座標登録する
E.署名画像生成	軌跡から画像生成する

2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順①
  フォームに次のコンポーネントを配置
  TTRoundRect、Timage、TButton



描画領域
TRoundRect

画像生成確認用
（実際の実装では不要）
TImage

署名画像生成
TButton

署名クリア
TButton

2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順②

A.軌跡情報制御

必要なクラス・変数・手続きの宣言

軌跡情報クラスの宣言

```
type
  //座標クラス
  TSigCapRec = Record
    CurPos   : TPointF;
    PosState : Byte;
  end;
```

グローバル変数・手続きの宣言

```
private
   { private 宣言 }
   Signature :  TList<TSigCapRec>;  //軌跡情報持用（座標クラスの配列）
   DrawFlg   :   Boolean;           //描画制御用
   procedure Addpoint(const aX, aY: Single; const aState:Byte);  //座標追加手続き
```

2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順③

A.軌跡情報制御

座標位置手続きの実装

座標位置追加手続き

```
procedure TForm1.Addpoint(const aX, aY: Single; const aState: Byte);
var
  P :TSigCapRec;
//  H,M,S,Sm:Word;
begin
  //カーソル位置とステータスを軌跡クラスに保持
  P.CurPos   := PointF(aX, aY);
  P.PosState := aState;

  //軌跡情報が空であればステータスを0に変更
  if Signature.Count - 1 < 0 then P.PosState := 0;

  //ステータスが1以外であれば座標を追加
  if P.PosState <> 1 then
  begin
    Signature.Add(P);
  end
```

2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順④

A.軌跡情報制御

座標位置手続きの実装

座標位置追加手続き（前ページ続き）

```
//ステータスが1で軌跡距離が規定より大きければ座標を追加
  else if P.CurPos.Distance(Signature.Last.CurPos) > 8.0 then
  begin
    Signature.Add(P);
  end;
//軌跡情報を使って描画領域に描画
  RoundRect1.Repaint;
end;
```

2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順⑤

B.描画制御

軌跡描画処理イベントの実装

OnPaint処理(軌跡描画)

```
procedure TForm1.RoundRect1Paint(Sender: TObject; Canvas: TCanvas;
  const ARect: TRectF);
var
  P : TSigCapRec;
  P1 , P2 : TPointF;
begin
  // 軌跡情報が空であれば描画しない
  if not (Signature.Count - 1 > 0) then Exit;

  //描画設定
  Canvas.Stroke.Dash  := TStrokeDash.sdSolid; //実線
  Canvas.Stroke.Thickness := 4;               //太さ
  Canvas.Stroke.Color := TAlphaColorRec.Red;  //色
```

2-2. タッチ操作を使った画面署名機能

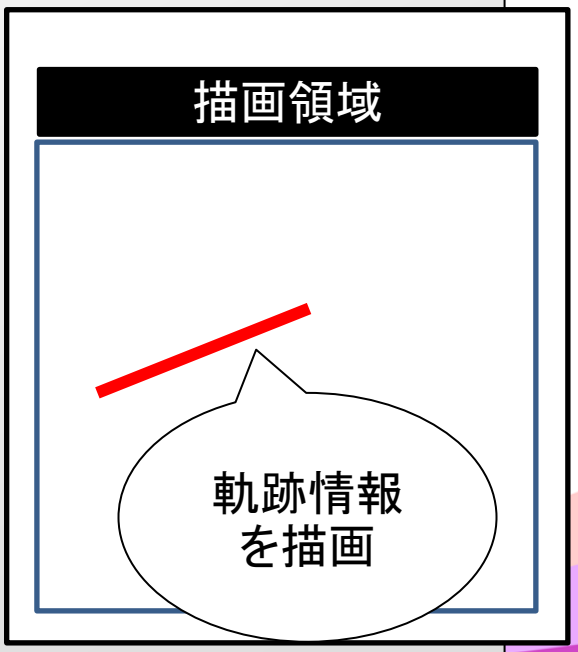
- 署名機能の実装手順⑥

B.描画制御

軌跡描画処理イベントの実装

OnPaint処理(軌跡描画　前ページ続き)

```
//軌跡情報の配列を回して描画
for P in Signature do
begin
  case P.PosState of
    //ステータスが0であれば初期値に設定
    0: P1 := P.CurPos;
    //ステータスが1であれば初期位置から描画して、次の初期位置に設定
    1: begin
         P2 := P.CurPos;
         Canvas.DrawLine(P1, P2, 1, Canvas.Stroke);
         P1 := P.CurPos;
       end;
    //ステータスが2であれば初期位置から描画のみ
    2: begin
         P2 := P.CurPos;
         Canvas.DrawLine(P1, P2, 1, Canvas.Stroke);
       end;
    end;
  end;
end;
```



2-2. タッチ操作を使った画面署名機能

- 署名機能の実装手順⑦

C.初期・終了制御

画面初期処理・終了処理の実装

OnCreate処理（画面初期）

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Signature := TList<TSigCapRec>.Create;  //軌跡情報の生成
end;
```

OnDestroy処理（画面終了）

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
  FreeAndNil(Signature);  //軌跡情報の破棄
end;
```

2-2. タッチ操作を使った画面署名機能

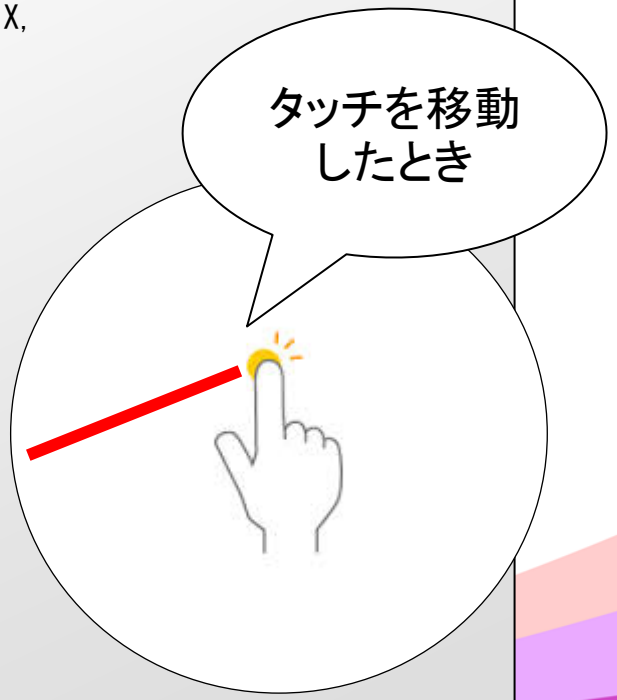
- 署名機能の実装手順⑧

D.タッチ操作制御

タッチ操作(マウス)の制御実装A

OnMouseMove処理(タッチ操作制御)

```
procedure TForm1.RoundRect1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Single);
begin
  //タッチ操作状態のときだけ処理
  if ssLeft in Shift then
  begin
    //描画フラグがOFFであれば、ステータス0で座標を追加して
    //描画フラグをONに設定
    if not DrawFlg then
    begin
      Addpoint(X, Y, 0);
      DrawFlg := True;
    end
    //描画フラグがONであれば、ステータス1で座標追加
    else
    begin
      Addpoint(X, Y, 1);
    end;
  end;
end;
```



2-2. タッチ操作を使った画面署名機能

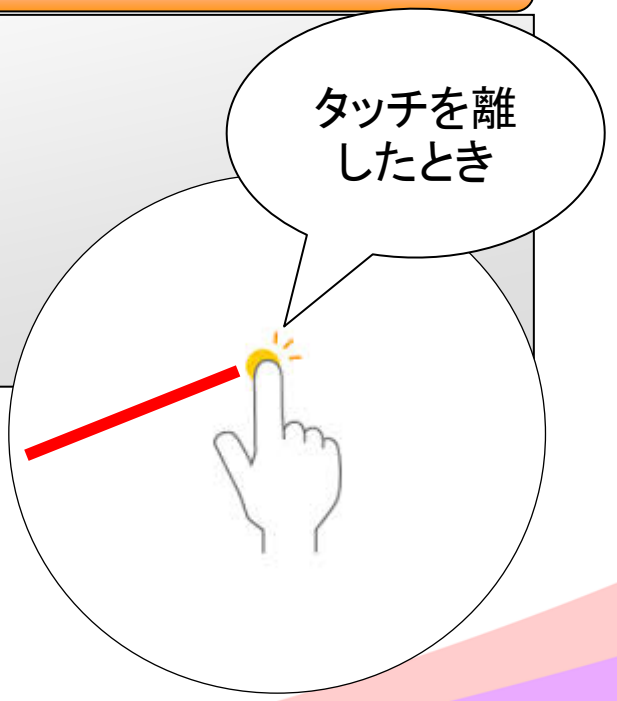
- 署名機能の実装手順⑨

D.タッチ操作制御

タッチ操作(マウス)の制御実装B

OnMouseUp処理(タッチ操作終了)

```
procedure TForm1.RoundRect1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Single);
begin
  //描画フラグをOFFに設定
  DrawFlg := False;
  //ステータス2で座標追加
  Addpoint(X, Y, 2);
end;
```



2-2. タッチ操作を使った画面署名機能

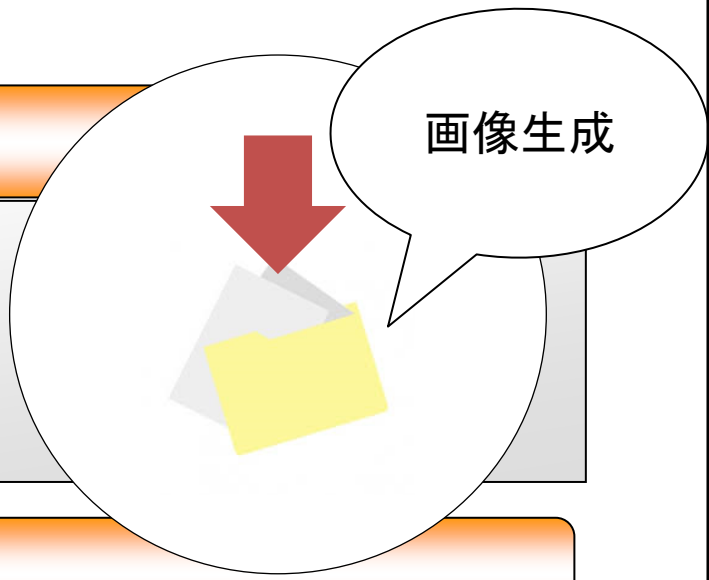
- 署名機能の実装手順⑩

E.署名画像生成

各Buttonクリックイベントの実装

OnClick処理(署名画像生成)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //スクリーンショットで画像を生成して
  //確認用Imageに表示
  Image1.Bitmap := RoundRect1.MakeScreenshot;
end;
```

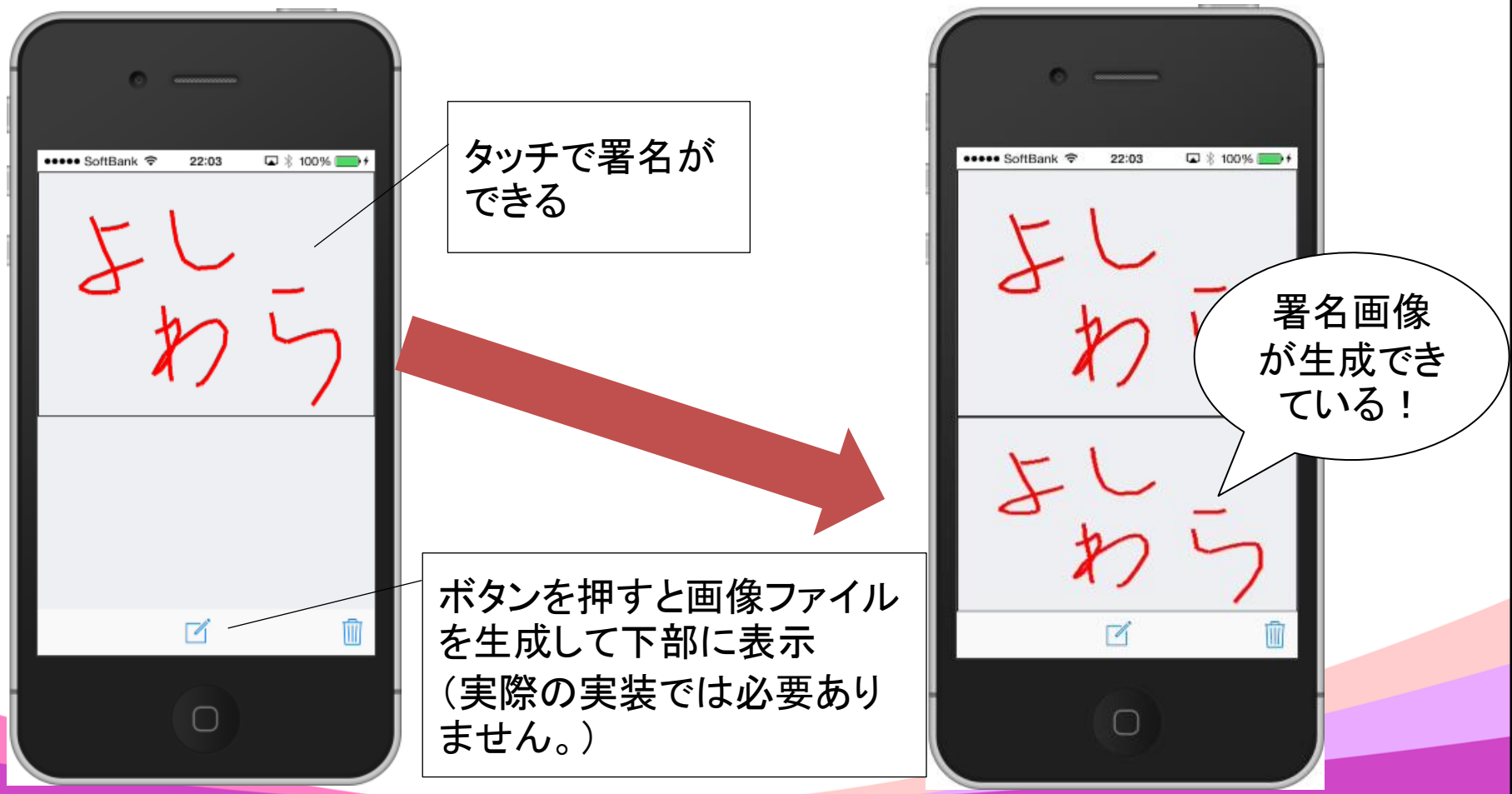


OnClick処理(署名初期化)　＜補足＞

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  Signature.Clear;          //軌跡情報の初期化
end;
```

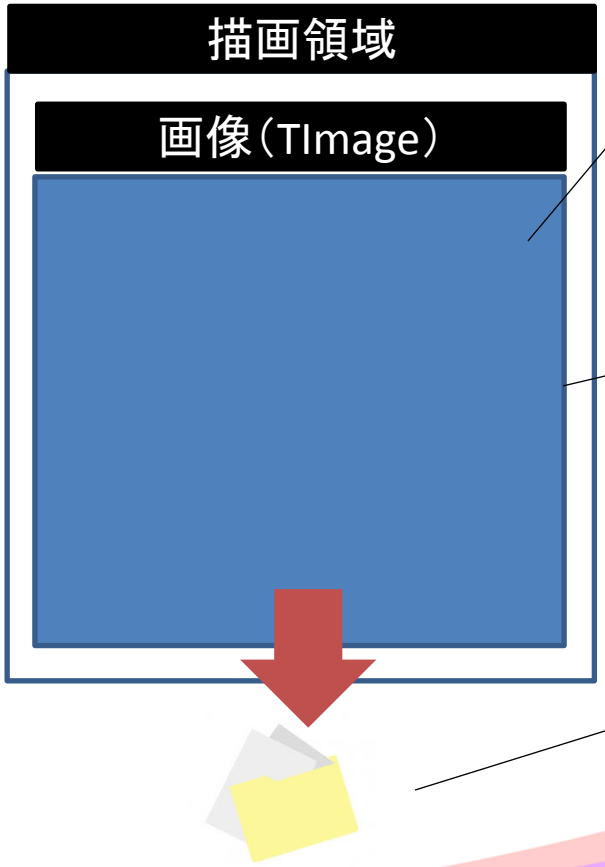
2-2. タッチ操作を使った画面署名機能

- 署名機能の実行



2-2. タッチ操作を使った画面署名機能

- タッチ操作を利用した署名の仕組みの応用
  応用すれば、写真や画像に対して署名やメモをする機能としても実装できます。画像はスクリーンショットとして生成しているので特別な制御も不要です。



例）
①撮影した画像を重ねて表示

②タッチ操作でマウスのように軌跡を表現

③メモが終わったらスクリーンショットで画像に保存

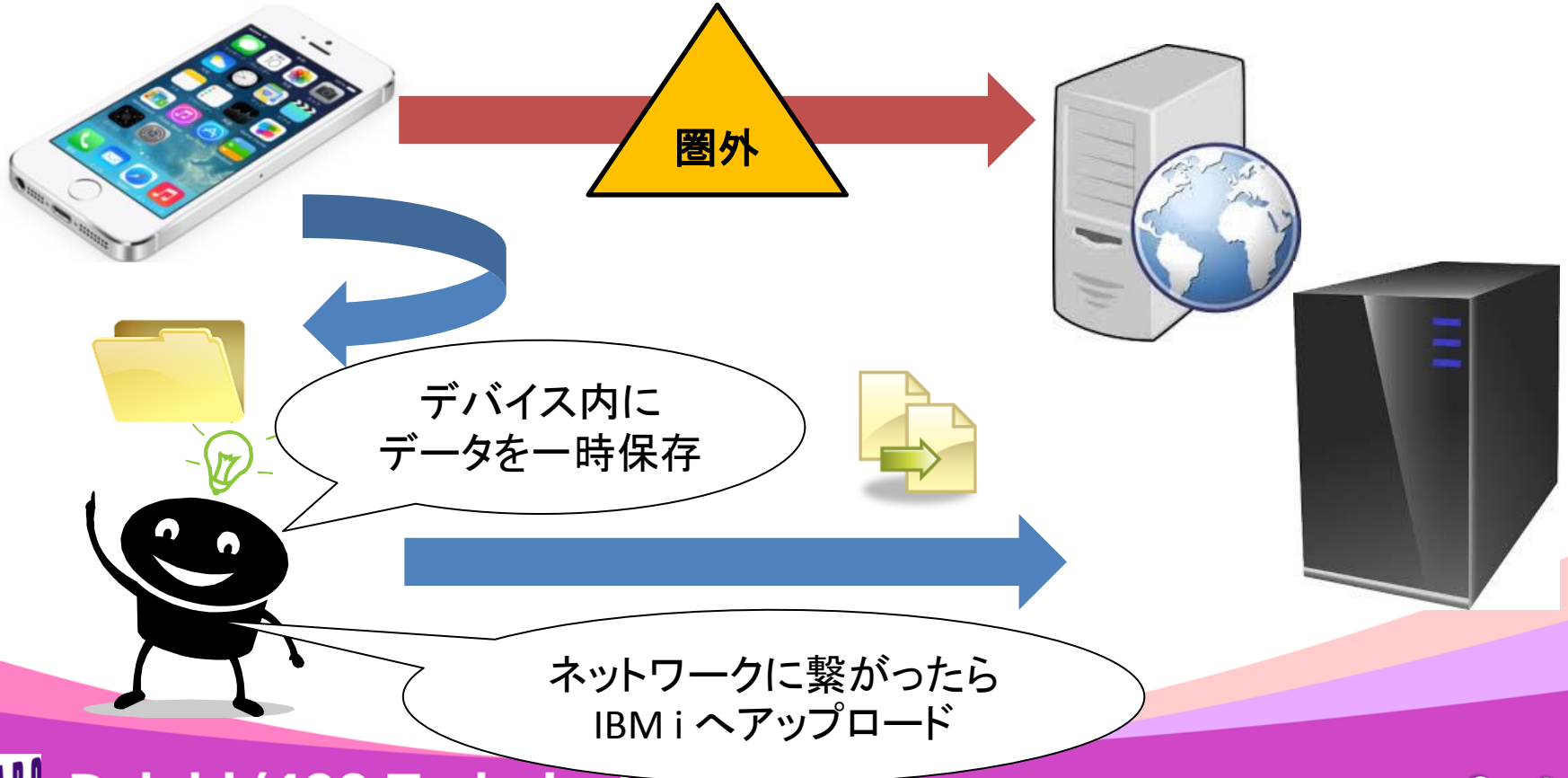
2.Delphi/400 ネイティブ機能の開発テクニック

2-3. オフラインでのローカルデータ保存

2-3. オフラインでのローカルデータ保存

- スマートデバイス内にローカルデータを保存する仕組み

  スマートデバイスは無線で接続するため、ネットワークに接続できない場合があります。ネイティブアプリケーションならデバイス内のファイルを操作できるので、ローカルデータとして保存しておくことができます。



2-3. オフラインでのローカルデータ保存

- ネイティブデバイスのローカル内でファイル管理
  Delphi/400ではアプリケーションとセットでネイティブのファイルを配置して利用することができます。
  （画像、動画、音源、データ、Iniファイルなど）

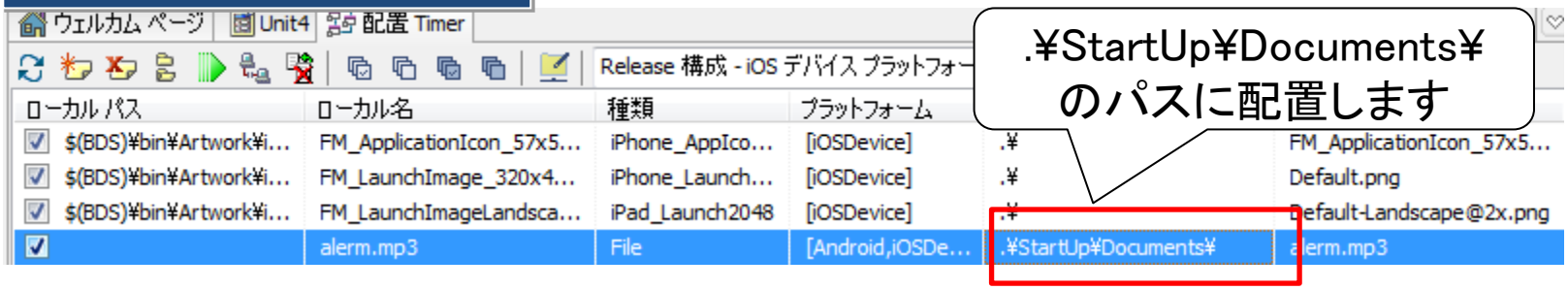
  ただし注意点もあります。
  スマートデバイス上ではWindowsのエクスプローラのように、自由な場所にファイルを作成・操作できるわけではありません。
  基本的には、操作できる領域は限定されています。

使用できる領域（パス）はデバイス（ iOSやAndroid）によって異なるので考慮が必要です

2-3. オフラインでのローカルデータ保存

- iOSでのファイル配置パス
  [プロジェクト|配置]から必要なファイルを追加することができます。

iOS



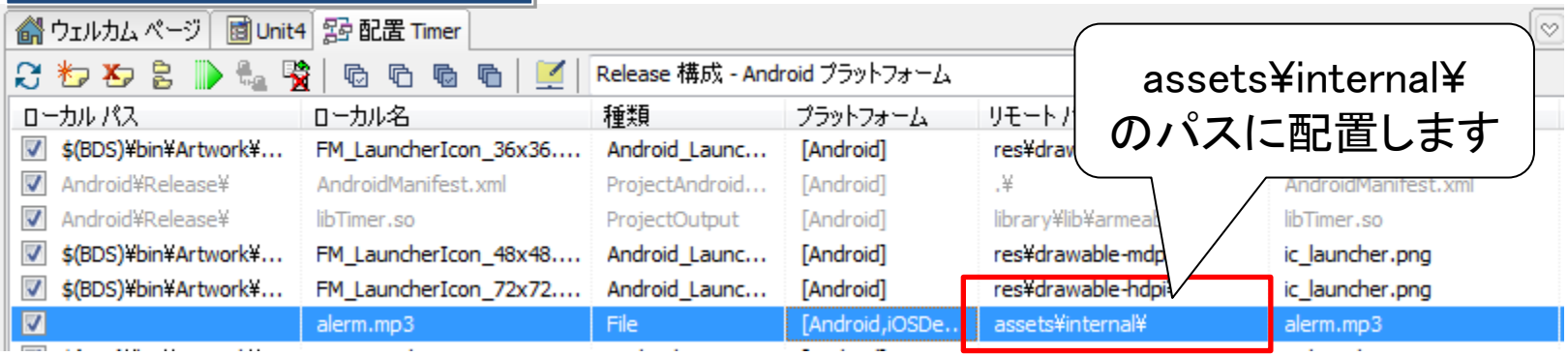
プログラム上でのネイティブファイルパス指定方法(iOS)

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

2-3. オフラインでのローカルデータ保存

- Androidでのファイル配置パス
  [プロジェクト|配置]から必要なファイルを追加することができます。

Android



プログラム上でのネイティブファイルパス指定方法（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

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装例

①IBM i から
データを取得

②ローカルでデータの
編集&保存

③ローカルのデータを
IBM i へ反映

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装例

A.IBM i データ取得

B.ローカルデータ保存

C.ローカルデータ確認

D.IBM i アップロード

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装①
  各ボタン処理の実装

A.IBM i データ取得

OnClick処理1(IBM i データの取得)

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //一度切断
  ClientDataSet1.Close;
  SQLConnection1.Connected := False;
  //表示切替
  BindSourceDB1.DataSet := ClientDataSet1;
  //IBM i へ接続
  SQLConnection1.Connected := True;
  //データを開く
  ClientDataSet1.Open;
  //DBラベルの変更
  Label1.Text := 'DB:IBM i';
  //完了メッセージ
  ShowMessage('IBM i からデータを取得しました');
end;
```

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装②
  各ボタン処理の実装

B.ローカルデータ保存

OnClick処理2(ローカルデータの保存)

```
procedure TForm1.Button12Click(Sender: TObject);
var
  sFolder: string;
begin
  //iOSの場合のパス設定
  {$IFDEF IOS}
  sFolder := GetHomePath + PathDelim + 'Documents';
  {$ENDIF IOS}

  //Androidの場合のパス設定
  {$IFDEF ANDROID}
  sFolder := TPath. GetDocumentsPath;
  {$ENDIF ANDROID}
```

【ポイント】
デバイス毎に違うコーディングをする場合は
{$IFDEF デバイス}
{$ENDIF デバイス}
の構文を使用します。

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装③
  各ボタン処理の実装

B.ローカルデータ保存

OnClick処理2(ローカルデータの保存　前ページ続き)

```
  //パスが存在しなければディレクトリを作成
  if not DirectoryExists(sFolder) then
    MkDir(sFolder);

  //保存ファイル名の指定 (pFileNameはグローバル変数)
  pFileName := sFolder + PathDelim + 'Sample.cds';

  //デバイスのファイルを保存
  ClientDataSet1.SaveToFile(pFileName, dfBinary);
  //完了メッセージ
  ShowMessage('デバイス内に保存しました');
end;
```

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装④
  各ボタン処理の実装

C.ローカルデータ確認

OnClick処理3(ローカルデータの確認)

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  //一度切断
  ClientDataSet1.Close;
  //参照先をローカルファイルに設定
  ClientDataSet2.FileName := pFileName;
  //表示を切り替え
  BindSourceDB1.DataSet := ClientDataSet2;
  //データを開く
  ClientDataSet2.Open;
  //DBラベルの変更
  Label1.Text := 'DB:ローカル';
  //完了メッセージ
  ShowMessage('デバイス内からデータを取得しました');
end;
```

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存の実装⑤
  各ボタン処理の実装

D.IBM i アップロード

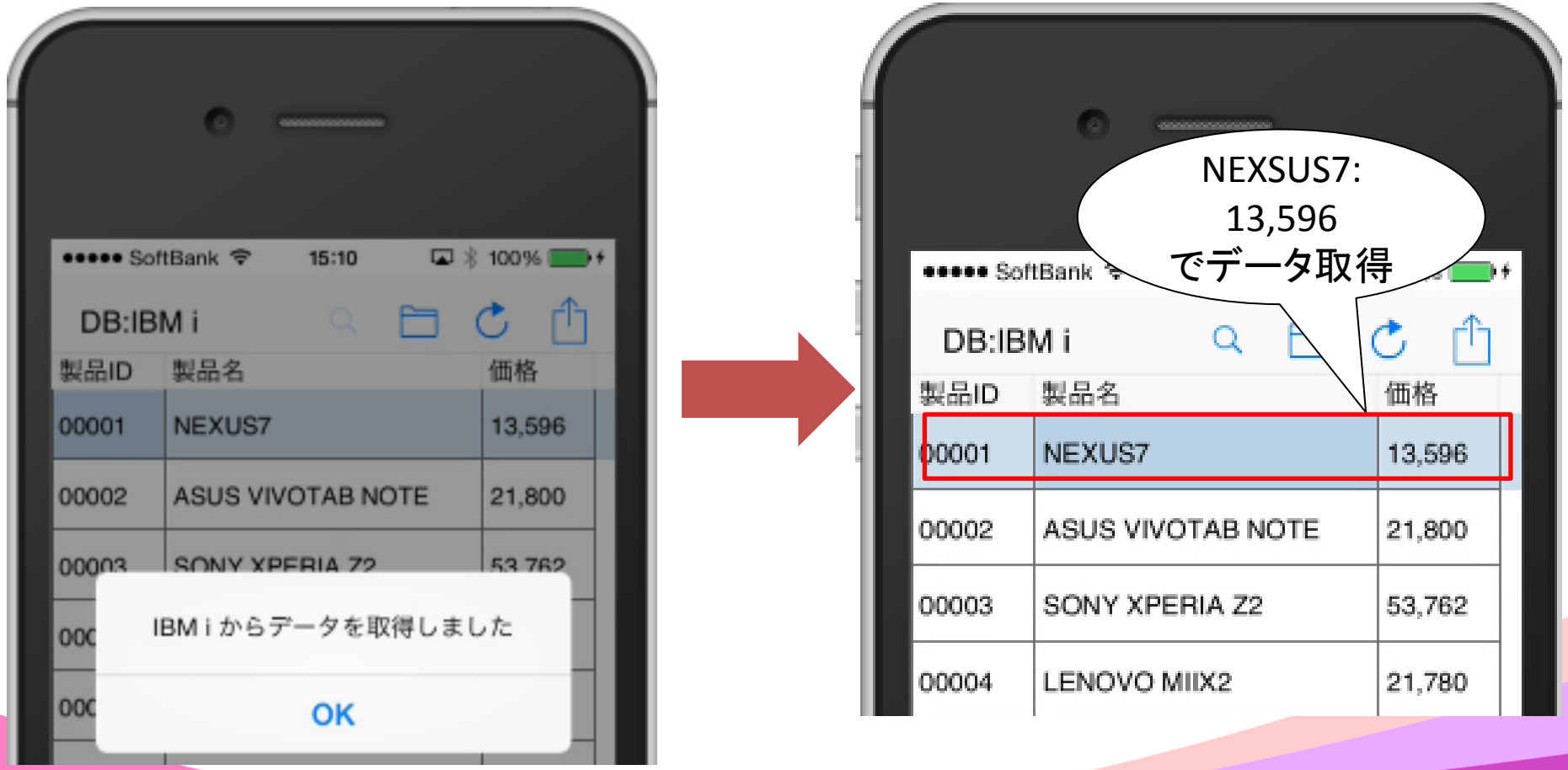
OnClick処理4(IBM i へのデータの反映)

```
procedure TForm1.Button4Click(Sender: TObject);
Begin
  //データを開く際にローカルデータをコピー
  ClientDataSet1.Close;
  ClientDataSet1.CreateDataSet;
  ClientDataSet1.Data := ClientDataSet2.Data;

  //IBM i へ変更データを一括反映
  ClientDataSet1.ApplyUpdates(-1);
  //完了メッセージ
  ShowMessage('IBM i にデータを反映しました');
end;
```

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存機能の実行
  IBM i からのデータ取得

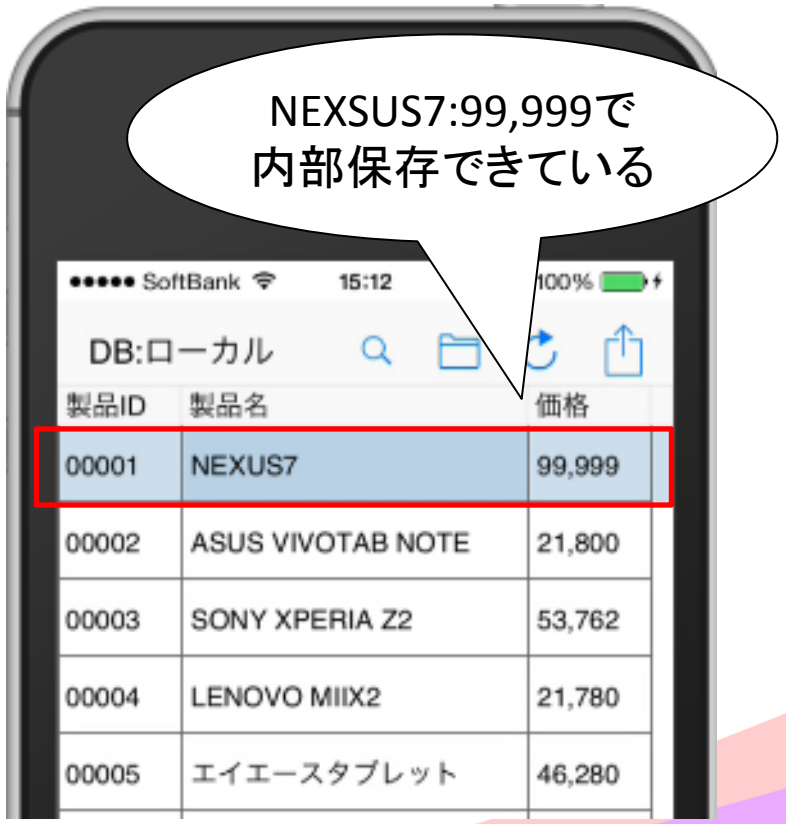


2-3. オフラインでのローカルデータ保存

- ローカルデータ保存機能の実行
  データを変更してデバイスローカルへ保存

2-3. オフラインでのローカルデータ保存

- ローカルデータ保存機能の実行
  デバイスローカルからデータを再取得（保存内容確認）



2-3. オフラインでのローカルデータ保存

- ローカルデータ保存機能の実行
  デバイスローカルからIBM i へデータのアップロード

3.まとめ

3.まとめ

- バーコード読み取りはカメラ機能を連動して実現できる
- 画面署名や画像メモはタッチ操作の軌跡を
  トレースして実現できる
- オフライン時のデバイスローカルへのデータ保存は
  デバイス毎の内部領域を利用して実現できる
- Delphi/400では様々なスマートデバイスネイティブ
  機能の連携が実現できる
  （他にも録音・再生や通知など多くの機能が利用できます）

ご静聴ありがとうございました