【セッションNo．3】

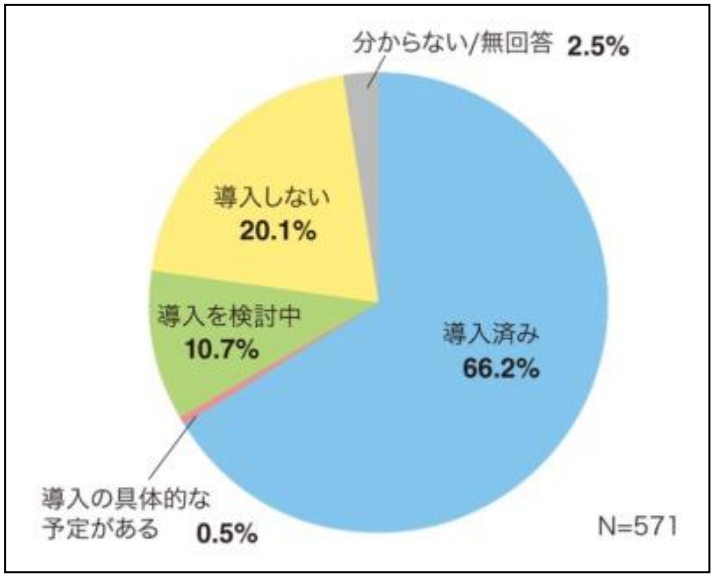
Delphi/400技術セッション
ステップアップ！
モバイルアプリケーション開発

株式会社ミガロ.
RAD事業部　技術支援課
吉原　泰介

はじめに

モバイル機器の企業利用は数年前に比べると格段に増え、すでに7割近くの企業で導入が進んでいます。

同時にモバイルアプリケーションの需要・自社開発も増加しており、Delphi/400テクニカルサポートでもお問い合わせが増えてきました。本セッションでは、テクニカルサポートの技術ノウハウから簡単かつ実用的なモバイルアプリケーションの開発テクニックをご紹介いたします。



出典：日経コミュニケーション
企業ネット/ICT利活用実態調査
（モバイル機器導入）

【アジェンダ】

1. アップテザリングによるモバイル機器活用

2.モバイルアプリケーションの帳票実装

3.まとめ

1.アップテザリングによるモバイル機器活用

1.アップテザリングによるモバイル機器活用

モバイル活用が広がるアップテザリング

アップテザリング（アプリケーションテザリング）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テザリング機能。
この機能を使用するとVCLとFireMonkeyのアプリケーション間でも連携ができる為、
既存のデスクトップアプリケーションと連携するモバイルアプリケーションが開発可能。

アップテザリングの前提

同じネットワークに両機器がつながっているか、
または両機器がBluetoothで通信できることが必要

1.アップテザリングによるモバイル機器活用

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で連携することが多かった。

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

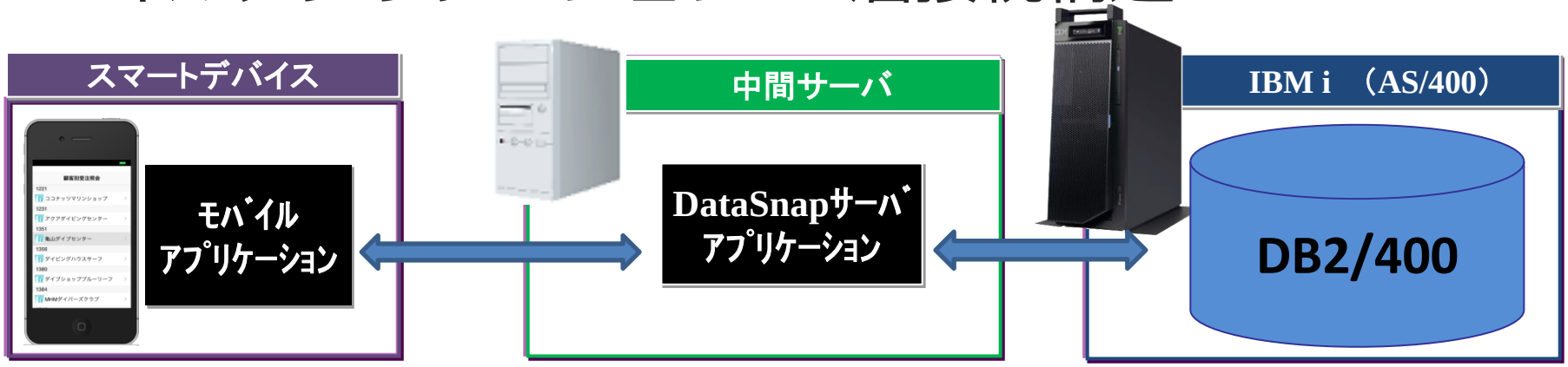
専用のバーコードリーダーやPOSを用意して、
PCのアプリケーションから登録・更新

アップテザリングで代用！

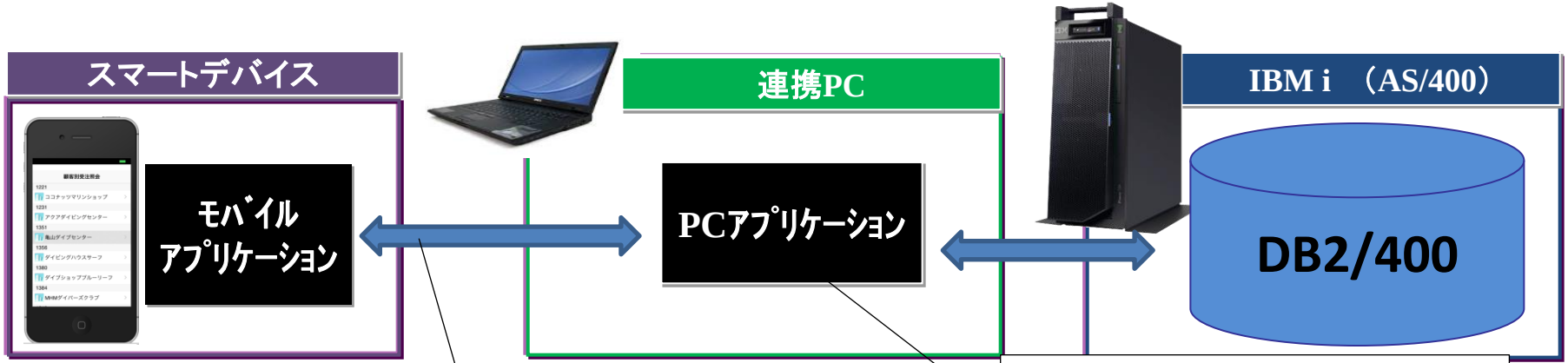
写真撮影やバーコード読取結果を
スマートフォンアプリからPCアプリに
送信して登録・更新
（アプリで自動化が可能）

1.アップテザリングによるモバイル機器活用

モバイルアプリケーションの3層接続構造



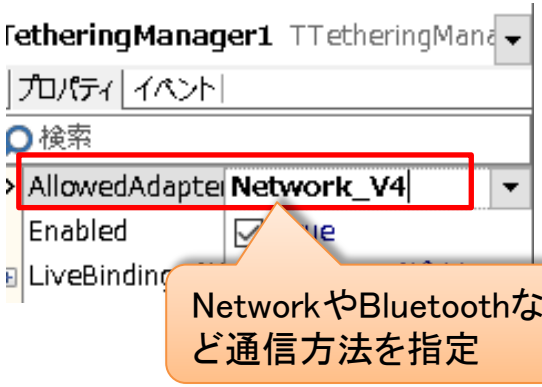
アプリケーションテザリングの3層接続構造



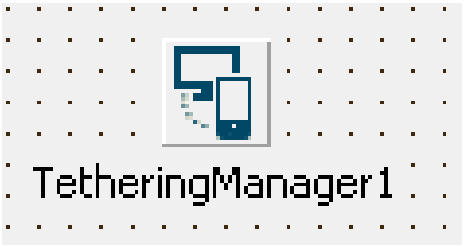
1.アップテザリングによるモバイル機器活用

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
TTetheringManagerとTTetheringAppProfileコンポーネントを配置

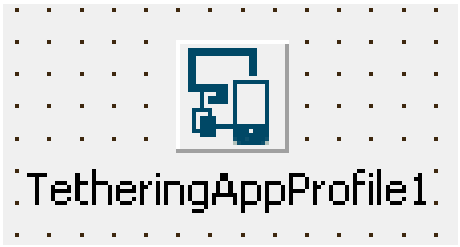


TTetheringManagerコンポーネント



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

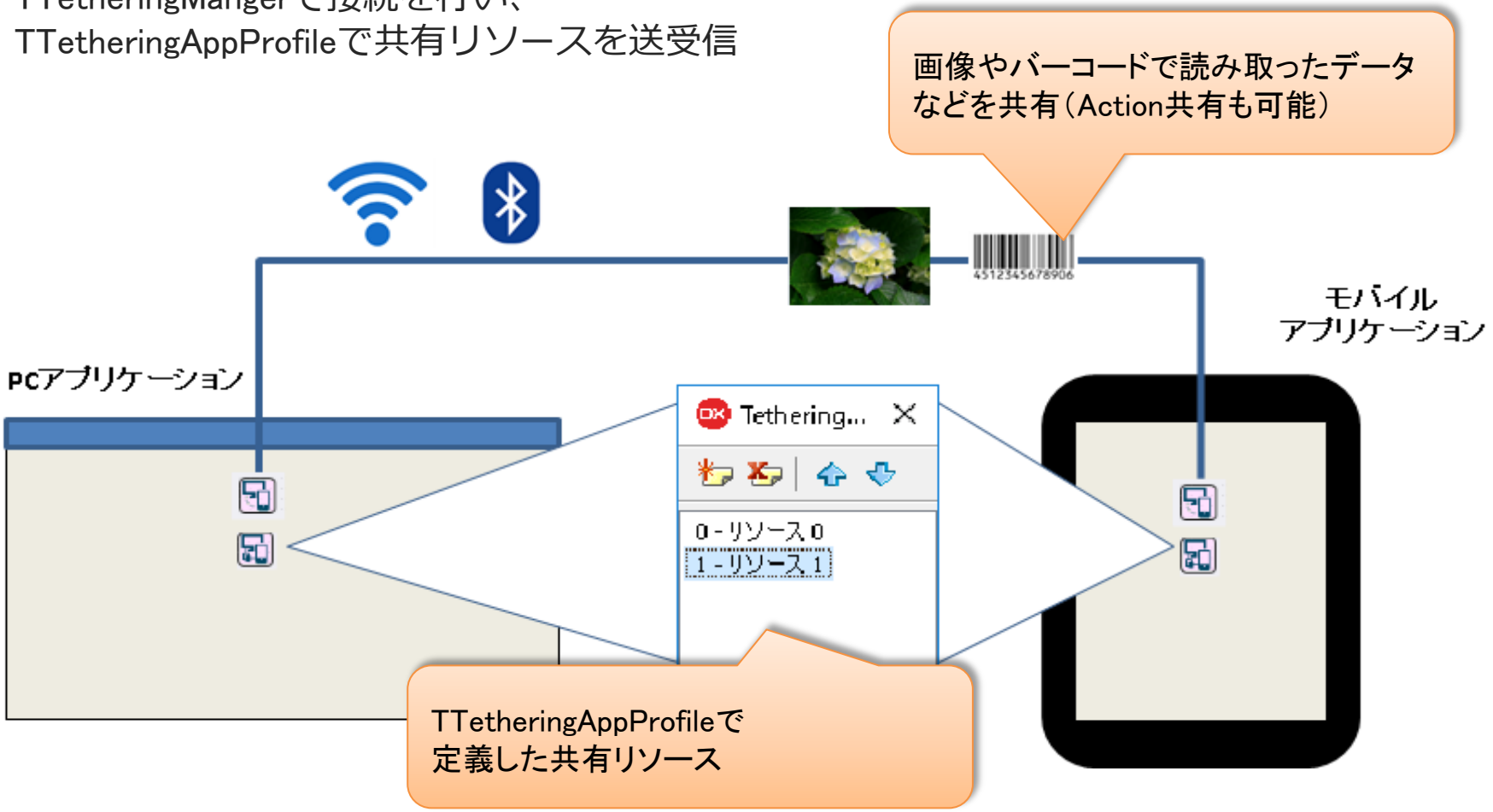


テザリングで接続した
アプリケーション間で共有する
リソースの制御

1.アップテザリングによるモバイル機器活用

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



1.アップテザリングによるモバイル機器活用

PCとスマートフォンのアプリケーション連携例

拡張するアプリケーション

PCアプリケーション

モバイルアプリケーション

読取バーコード
送信

撮影写真
送信

1.アップテザリングによるモバイル機器活用

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

TTetheringManager
（接続管理用）

TTetheringAppProfile
（リソース共有用）

TActionList
（写真撮影用）

TTMSFMXZBarReader
（バーコード読取用）
TMSのフリーコンポーネント

TImage
（撮影写真表示用）

TButton × 2　下に配置
（撮影＆バーコード読取用）

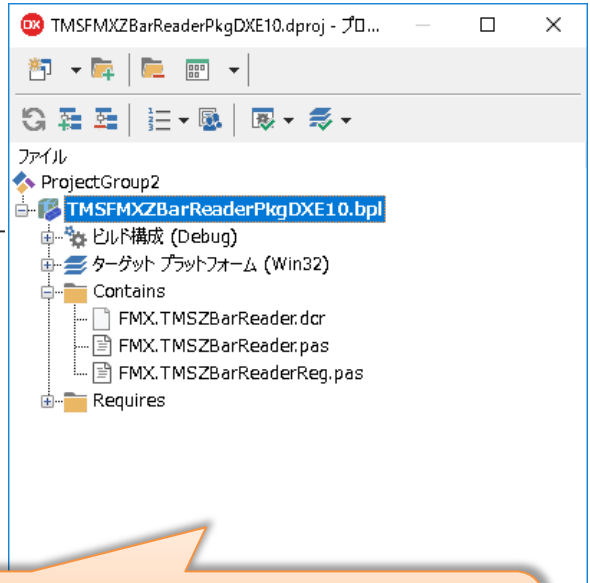
1.アップテザリングによるモバイル機器活用

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/freetools.asp#TTMSFMXZBarReader

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。（Android開発の場合はこのコンポーネントで読み替えて下さい）
http://www.file-upload.net/download-8601754/TKRBarCodeSanner.zip.html

使い方はZbarSDKとほぼ同じです。
XE5当時に作られているものなのでXE7以降ではソースの修正が必要です。



Berlinまでしか対応書いてませんが
Tokyoでも動きました

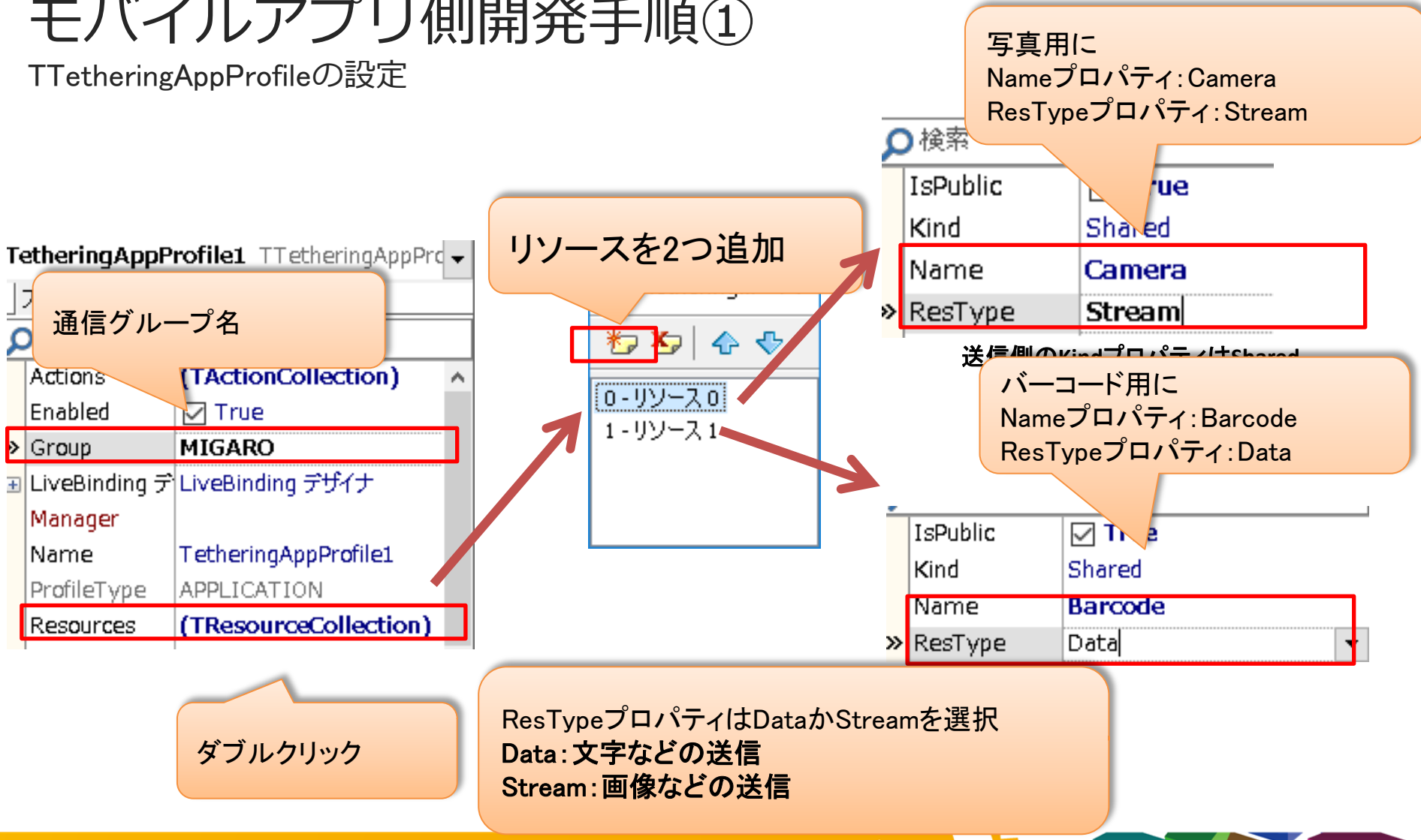
```
interface

uses
  System.Classes
  {$IFDEF IOS}
  ,FMX.TMSZBarReader
  {$ENDIF}
  {$IFDEF ANDROID}
  ,FMX.Platform, FMX.Helpers.Android, System.Rtti, FMX.Types, System.SysUtils,
  Androidapi.JNI.GraphicsContentViewText, Androidapi.JNI.JavaTypes,
  FMX.StdCtrls, FMX.Edit, Androidapi.Helpers   // Androidapi.Helpers 追加
  {$ENDIF}
  ;
```

```
{$IFDEF ANDROID}
function TTKRBarCodeScanner.HandleAppEvent(AAppEvent: TApplicationEvent;
  AContext: TObject): Boolean;
var
  aeBecameActive : TApplicationEvent;  //----- 追加
begin
  aeBecameActive := TApplicationEvent.BecameActive;  //----- 追加
```

1.アップテザリングによるモバイル機器活用

モバイルアプリ側開発手順①

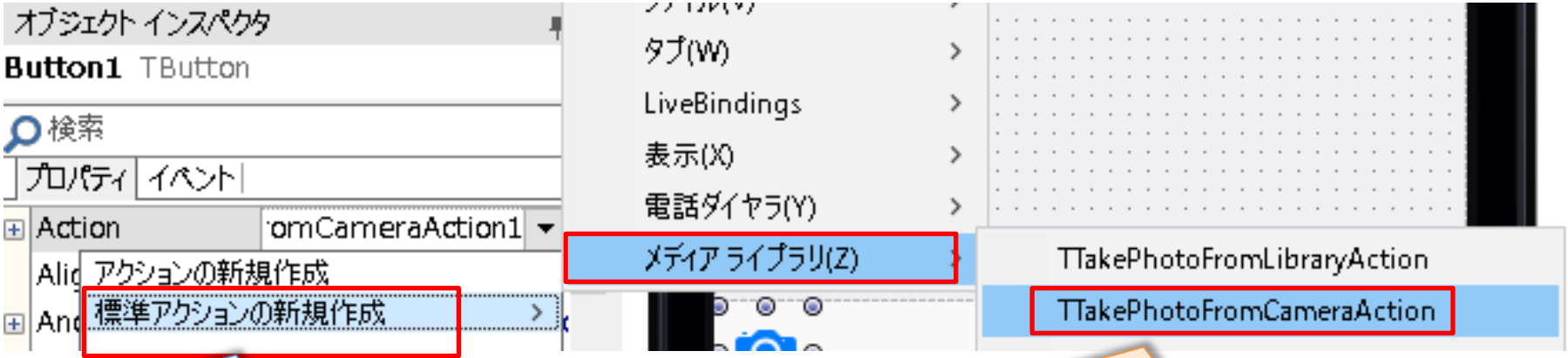
TTetheringAppProfileの設定



1.アップテザリングによるモバイル機器活用

モバイルアプリ側開発手順②

Actionの設定（写真撮影ボタン）



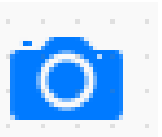
ButtonのActionプロパティで
標準アクションの新規追加

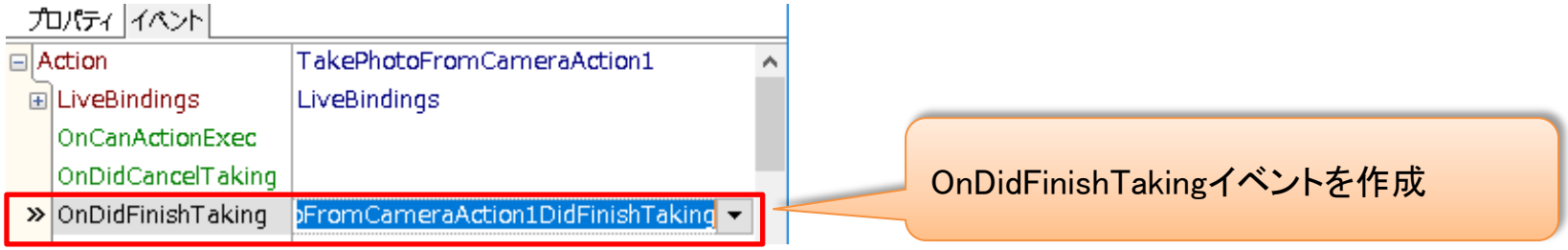
TTakePhotoFromCameraAction
を選択

1.アップテザリングによるモバイル機器活用

モバイルアプリ側開発手順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（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;
```

1.アップテザリングによるモバイル機器活用

モバイルアプリ側開発手順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;
```

1.アップテザリングによるモバイル機器活用

モバイルアプリ側開発手順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;
```

1.アップテザリングによるモバイル機器活用

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

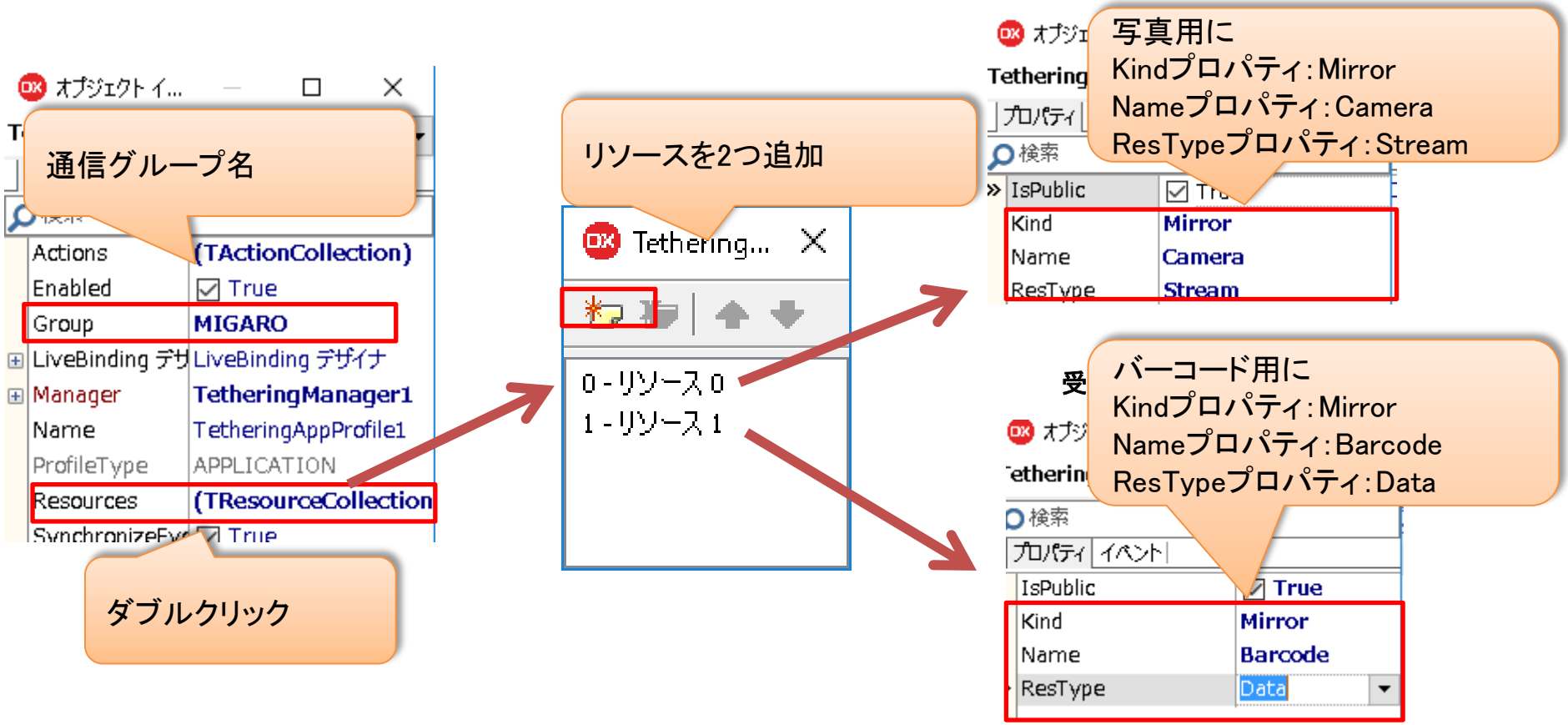
TTetheringManager
（接続管理用）

TTetheringAppProfile
（リソース共有用）

1.アップテザリングによるモバイル機器活用

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）



1.アップテザリングによるモバイル機器活用

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.TetheringAppProfile1ResourcesOResourceReceived
  (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;
```

1.アップテザリングによるモバイル機器活用

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;
```

プログラム完成

1.アップテザリングによるモバイル機器活用

PCとスマートフォンのアプリケーション連携拡張例

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！簡単に拡張が可能です。

PCアプリケーション

モバイルアプリケーション

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。

1.アップテザリングによるモバイル機器活用

補足：接続状況を表示する

TLabel （接続表示用）
TTimer （監視用）

OnTimerイベント（接続表示）

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Label1.Visible := (TetheringManager1.RemoteProfiles.Count > 0);
end;
```

接続カウントがあれば
Labelを表示

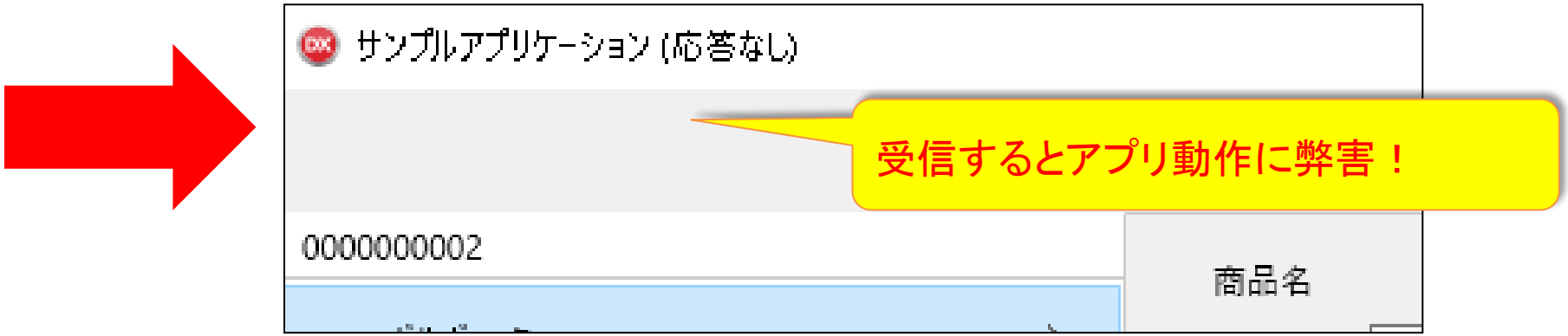
1.アップテザリングによるモバイル機器活用

補足：受信処理が重いとどうなってしまうか？

OnResourceReceivedイベント(撮影写真を受信)

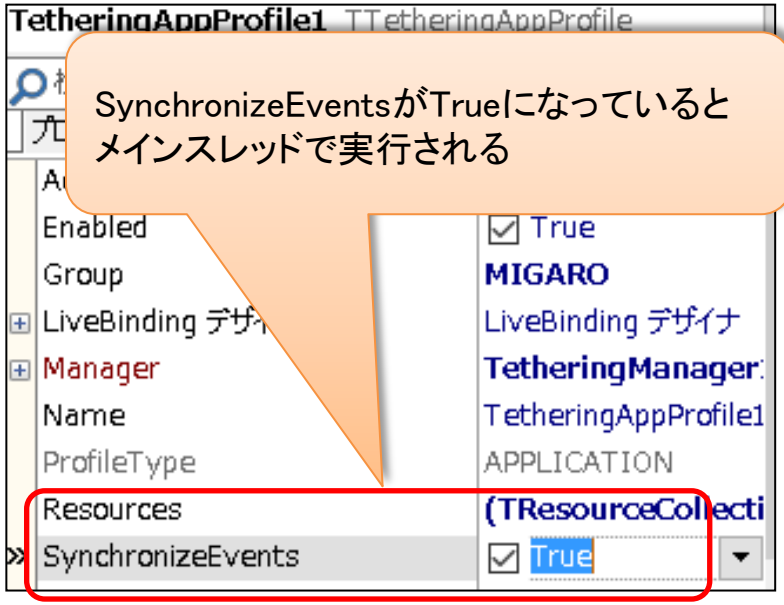
```
procedure TForm1.TetheringAppProfile1ResourcesOResourceReceived
  (const Sender: TObject; const AResource: TRemoteResource);
begin
  Sleep(10000);
end;
```

受信処理が重い場合



1.アップテザリングによるモバイル機器活用

補足：別スレッドで実行



SynchronizeEventsがTrueになっていると
メインスレッドで実行される

Falseにしておけば別スレッドになり、メインスレッドは
止まらずに使えて便利！
10.2 Tokyo以前は、自分でスレッドを考慮が必要

2.モバイルアプリケーションの帳票実装

2.モバイルアプリケーションの帳票実装

モバイルアプリケーションで扱う帳票形式

モバイルで帳票を扱う場合、一般的にPDFが使用されることが多い。
PDFであれば実際の紙媒体の代わりに画面で確認したり、
別ソフト（AirPrint等）を経由して実際にプリンタへの印刷もできる。

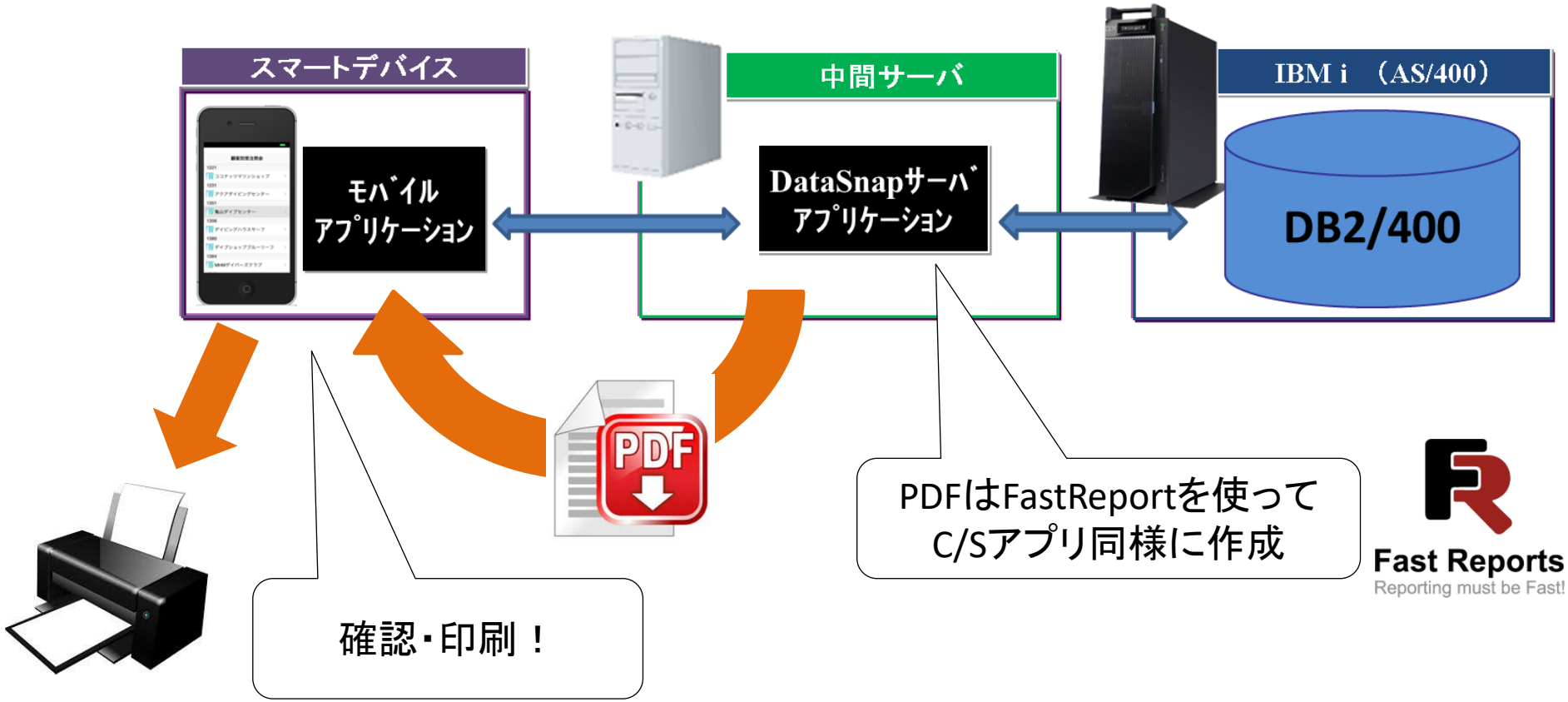
例）iOSであればPDFを
AirPrintからプリンタに印刷可能

2.モバイルアプリケーションの帳票実装

モバイル帳票実装の仕組み・構成

PDFファイルの作成は中間サーバ（DataSnapサーバ）で行う。

中間サーバ上では、FastReportが使用できるので開発も簡単。



2.モバイルアプリケーションの帳票実装

モバイルの帳票実装で必要となるプログラム機能

A.　PDFを作成するサーバ機能

【プログラムのポイント】

- **FastReportでPDFを作成して、Stream形式でPDFを返却する。**

B.　PDFを表示するモバイル機能

【プログラムのポイント】

- **iOSとAndroidでPDFの扱いの違いを考慮する。**
  （保存先や表示方法）

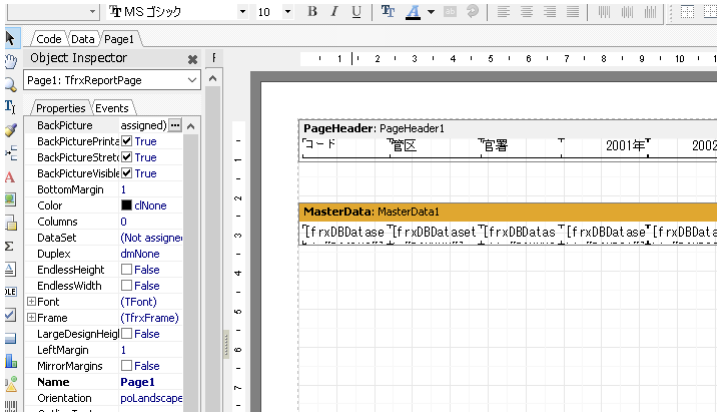
2.モバイルアプリケーションの帳票実装

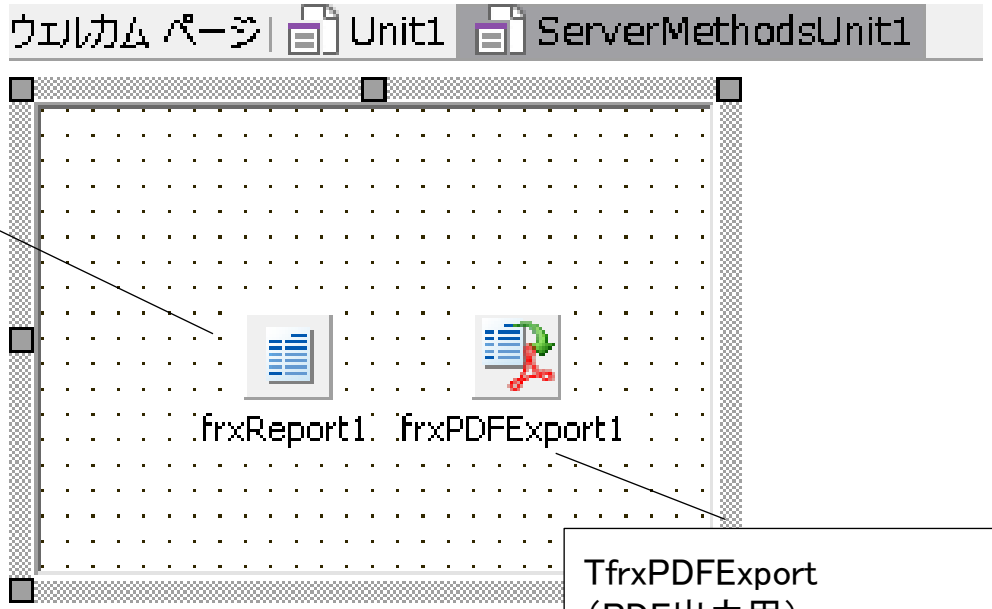
A.PDFを作成するサーバ機能 開発手順①

DataSnapサーバアプリ側設計画面（ServerMethods）

TfrxReport
（FastReport印刷用）

※帳票レイアウトや設定は
C/Sアプリ同様に作成しておく





TfrxPDFExport
（PDF出力用）

2.モバイルアプリケーションの帳票実装

A.PDFを作成するサーバ機能 開発手順②

サーバ機能（関数）の作成

PDF出力関数

```
//定義
 public
    function OutPdf(): TStream; //PDF出力用関数
 ・・・
//実装部
function TServerMethods1.OutPdf: TStream;
var
  msPDF: TStream;  //PDF返却用のStream変数
begin
  msPDF := TMemoryStream.Create;  //返却用にStream変数の生成（TMemoryStream）
  frxReport1.PrepareReport();     //FastReportの帳票作成
  frxPDFExport1.Stream := msPDF;  //frxPDFExportの出力Streamに変数を割り当てる
  frxReport1.Export(frxPDFExport1);//PDFの作成
  msPDF.Position := 0;             //ポジションの調整
  Result := msPDF;                 //Streamをモバイルに返却する
end;
```

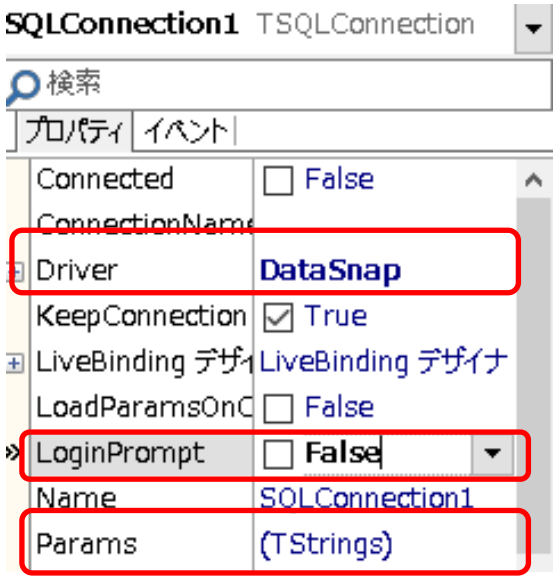
【ポイント】
DataSnapの受け渡しで
TMemoryStreamは使用できない
のでTStreamを使う

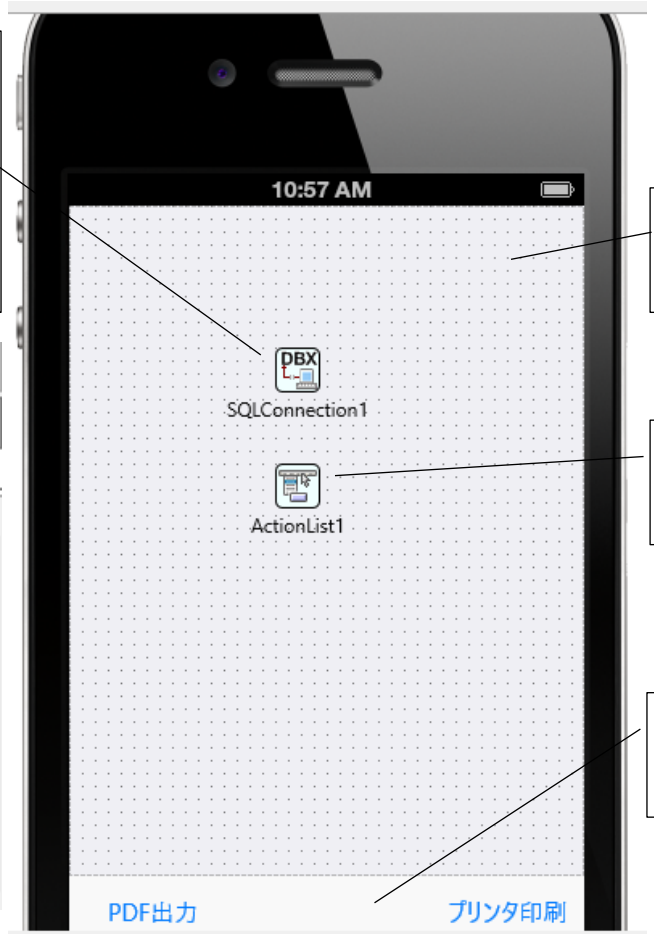
2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順①

iOSアプリケーション設計画面

TSQLConnection
（サーバ機能呼び出し用）
Driverプロパティ：DataSnap
LoginPromptプロパティ：False
Paramsプロパティ：
HostNameにサーバのIPアドレス





TWebBrowser
（PDF表示用）

TActionList
（プリンタ印刷用）

TToolBarおよびTButton×2
（処理ボタン用）

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順②

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

```
procedure TForm1.Button1Click(Sender: TObject);
const
  BufSize = $F000;
var
  smCon: TServerMethods1Client;  //DataSnapサーバ用
  msPDF: TMemoryStream;          //PDF受け取り用
  stPDF: TStream;                //PDFStream読み込み用
  sPath: String;                 //PDF保存先
  iBytesRead: Integer;           //Buffer計算用
  pbBuffer: PByte;               //Buffer読み込み用
begin
  try
    //DataSnapサーバに接続
    SQLConnection1.Open;
    smCon := TServerMethods1Client.Create(SQLConnection1.DBXConnection);
```

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順③

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

```
//PDF出力関数を呼び出して受け取ったStreamをMemoryStreamに転送
 msPDF := TMemoryStream.Create;
 GetMem(pbBuffer, BufSize);
 stPDF := smCon.OutPdf;
 stPDF.Seek( 0, TSeekOrigin.soBeginning );
 stPDF.Position := 0;
 repeat
   iBytesRead := stPDF.Read(pbBuffer^, BufSize);
   if iBytesRead > 0 then
     msPDF.WriteBuffer(pbBuffer^, iBytesRead);
 until iBytesRead < BufSize;
 msPDF.Seek(0, TseekOrigin.soBeginning);

 //PDF保存先の設定
 sPath := TPath.Combine(TPath.GetDocumentsPath,  'PrintXX.pdf');

 //念のため同名ファイルを削除してから保存
 DeleteFile(sPath);
 msPDF.SaveToFile(sPath);
```

【ポイント】
DataSnapからはTStreamで送られてくるので用意したTMemoryStreamに移し替える

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順④

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

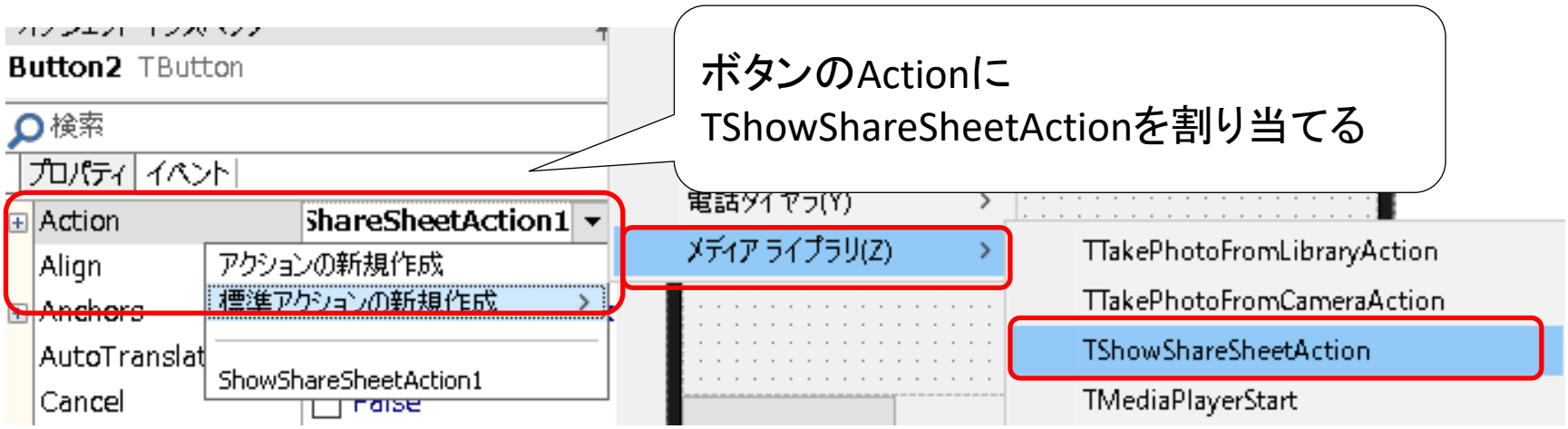
```
    //PDFを表示
    WebBrowser1.Navigate('file://' + sPath);

  finally
    //破棄
    msPDF.Free;
    smCon.Free;
  end;
end;
```

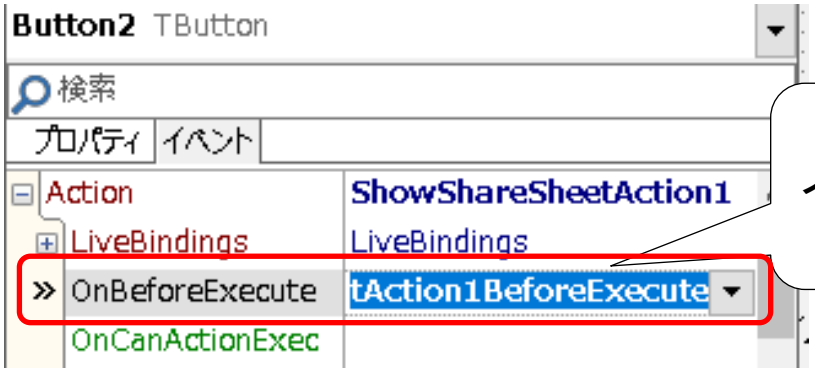
【ポイント】
iOSではPDFがブラウザ標準で表示できるのでTWebBrowserで表示する

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順⑤

モバイルアプリケーション側のプリンタ印刷ボタン





イベントはOnBeforeExecuteを作成

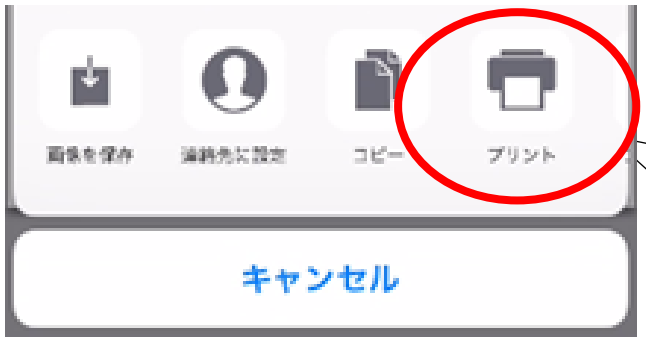
2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（iOS）開発手順⑥

モバイルアプリケーション側のプリンタ印刷ボタン

ShowShareSheetActionBeforeExecuteイベント(プリンタ出力ボタン)

```
procedure TForm2.ShowShareSheetAction1BeforeExecute(Sender: TObject);
begin
  ShowShareSheetAction1.Bitmap.Assign(WebBrowser1.MakeScreenshot);
end;
```



TWebBrowserに表示しているPDFをAirPrintへ印刷できるようにShareメニューを表示する

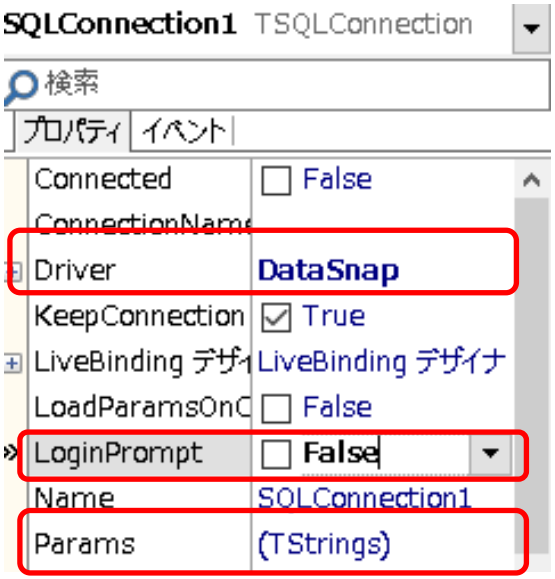
プログラム完成

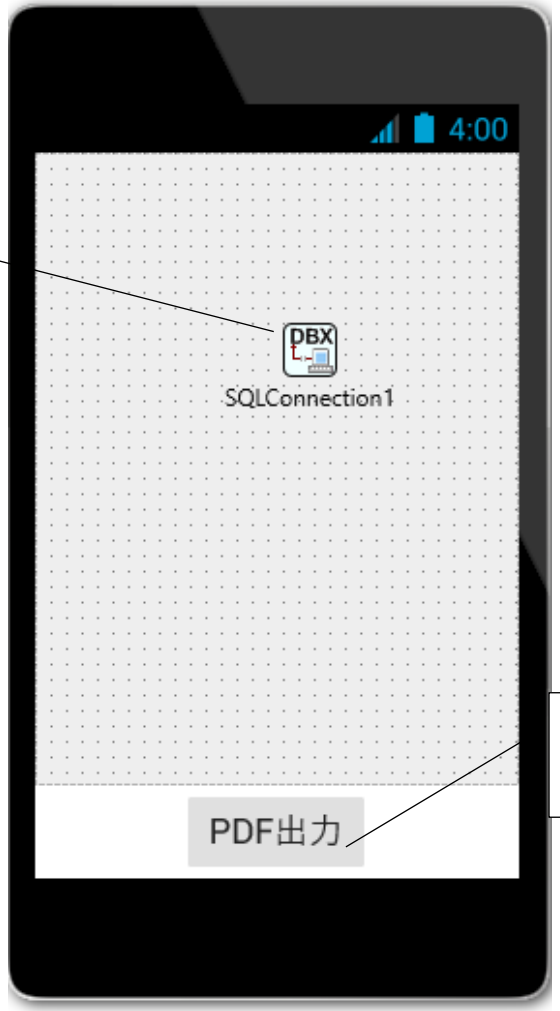
2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（Android）開発手順①

Androidアプリケーション設計画面

TSQLConnection
（サーバ機能呼び出し用）
Driverプロパティ：DataSnap
LoginPromptプロパティ：False
Paramsプロパティ：
HostNameにサーバのIPアドレス





TToolBarおよびTButton
（処理ボタン用）

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（Android）開発手順②

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

```
procedure TForm1.Button1Click(Sender: TObject);
const
  BufSize = $F000;
var
  smCon: TServerMethods1Client;  //DataSnapサーバ用
  msPDF: TMemoryStream;          //PDF受け取り用
  stPDF: TStream;                //PDFStream読み込み用
  sPath: String;                 //PDF保存先
  iBytesRead: Integer;           //Buffer計算用
  pbBuffer: PByte;               //Buffer読み込み用
  Intent: JIntent;               //PDF表示用Intent
begin
  try
    //DataSnapサーバに接続
    SQLConnection1.Open;
    smCon := TServerMethods1Client.Create(SQLConnection1.DBXConnection);
```

【ポイント】
TWebBrowserの代わりに
Intentを使って表示

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（Android）開発手順③

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

```
//PDF出力関数を呼び出して受け取ったStreamをMemoryStreamに転送
msPDF := TMemoryStream.Create;
GetMem(pbBuffer, BufSize);
stPDF := smCon.OutPdf;
stPDF.Seek( 0, TSeekOrigin.soBeginning );
stPDF.Position := 0;
repeat
  iBytesRead := stPDF.Read(pbBuffer^, BufSize);
  if iBytesRead > 0 then
    msPDF.WriteBuffer(pbBuffer^, iBytesRead);
until iBytesRead < BufSize;
msPDF.Seek(0, TseekOrigin.soBeginning);

//PDF保存先の設定
sPath := TPath.Combine(TPath.GetSharedDocumentsPath,  'PrintXX.pdf');

//念のため同名ファイルを削除してから保存
DeleteFile(sPath);
msPDF.SaveToFile(sPath);
```

【ポイント】
DataSnapからはTStreamで送られてくるので用意したTMemoryStreamに移し替える

2.モバイルアプリケーションの帳票実装

B.PDFを表示するモバイル機能（Android）開発手順④

モバイルアプリケーション側のPDF出力ボタン

OnClickイベント(PDF出力ボタン)

```
 //PDFを表示
    Intent := TJIntent.Create;
    Intent.setAction(TJIntent.JavaClass.ACTION_VIEW);
    Intent.setDataAndType(StrToJURI('file://' + sPath),
                StringToJString('application/pdf'));
    SharedActivity.StartActivity(Intent);

  finally
    //破棄
    msPDF.Free;
    smCon.Free;
  end;
end;
```

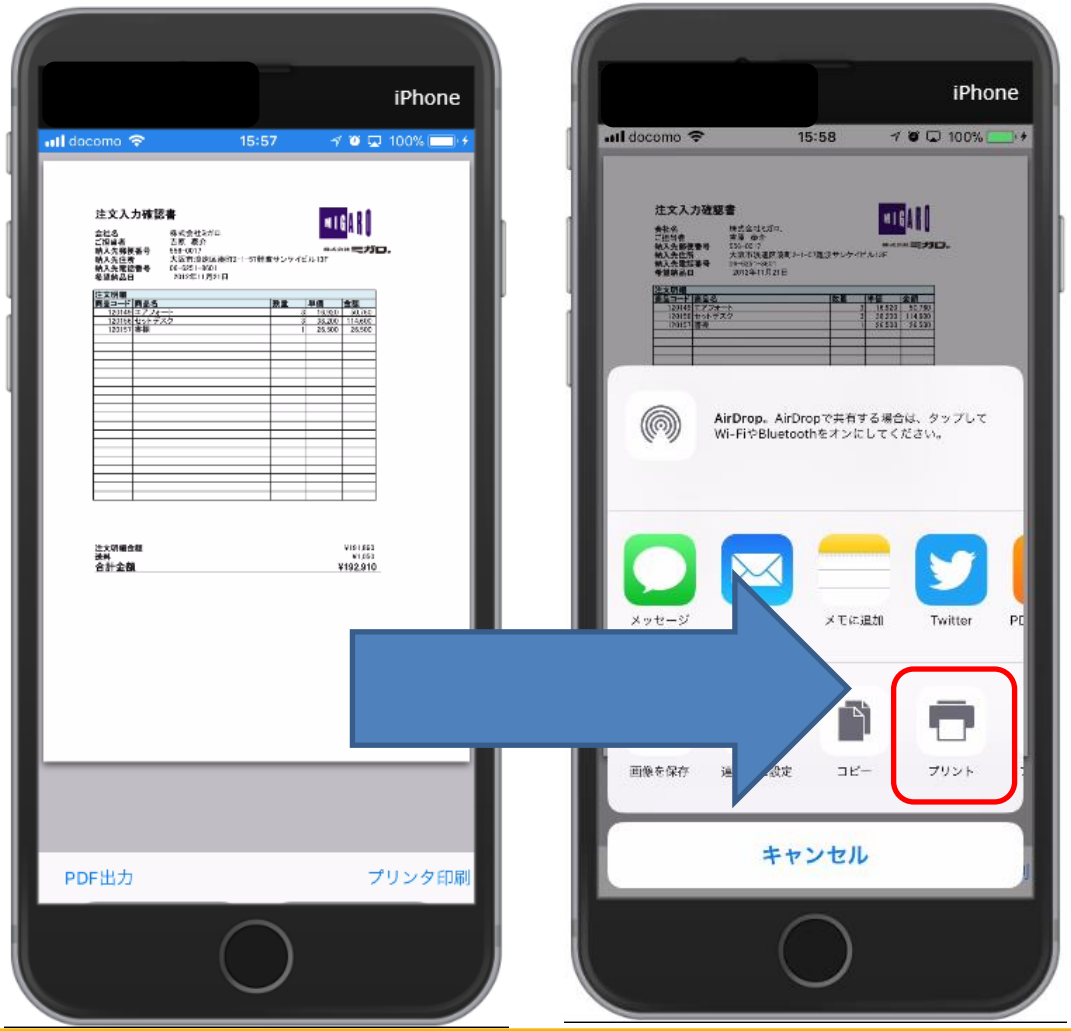
【ポイント】
AndroidではPDFがブラウザ標準で表示されないのでIntentを使って表示する。

プログラム完成

2.モバイルアプリケーションの帳票実装

アプリケーションの実行

iOSはAirPrintから印刷



AndroidはAcrobat上から印刷

2.モバイルアプリケーションの帳票実装

補足：iOSとAndroidのプログラムを同時に記述する例

OnClickイベント(PDF印刷ボタン)

```
procedure TForm1.Button1Click(Sender: TObject);
const
  BufSize = $F000;
var
  smCon: TServerMethods1Client;  //DataSnapサーバ用
  msPDF: TMemoryStream;          //PDF受け取り用
  stPDF: TStream;                //PDFStream読み込み用
  sPath: String;                 //PDF保存先
  iBytesRead: Integer;           //Buffer計算用
  pbBuffer: PByte;               //Buffer読み込み用
{$IFDEF ANDROID}
  Intent: JIntent;               //PDF表示用Intent
{$ENDIF}
begin
  try
    //DataSnapサーバに接続
    SQLConnection1.Open;
    smCon := TServerMethods1Client.Create(SQLConnection1.DBXConnection);
```

【ポイント】
IFDEF でプラットフォームを
分岐させる

2.モバイルアプリケーションの帳票実装

補足：iOSとAndroidのプログラムを同時に記述する例

OnClickイベント(PDF印刷ボタン)

```
//PDF出力関数を呼び出して受け取ったStreamをMemoryStreamに転送
    msPDF := TMemoryStream.Create;
    GetMem(pbBuffer, BufSize);
    stPDF := smCon.OutPdf;
    stPDF.Seek( 0, TSeekOrigin.soBeginning );
    stPDF.Position := 0;
    repeat
      iBytesRead := stPDF.Read(pbBuffer^, BufSize);
      if iBytesRead > 0 then
        msPDF.WriteBuffer(pbBuffer^, iBytesRead);
    until iBytesRead < BufSize;
    msPDF.Seek(0, TseekOrigin.soBeginning);

    //PDF保存先の設定
{$IFDEF IOS}
    sPath := TPath.Combine(TPath.GetDocumentsPath,  'PrintXX.pdf');
{$ENDIF}
{$IFDEF ANDROID}
    sPath := TPath.Combine(TPath.GetSharedDocumentsPath,  'PrintXX.pdf');
{$ENDIF}
```

2.モバイルアプリケーションの帳票実装

補足：iOSとAndroidのプログラムを同時に記述する例

OnClickイベント(PDF印刷ボタン)

```
//念のため同名ファイルを削除してから保存
    DeleteFile(sPath);
    msPDF.SaveToFile(sPath);
    //PDFを表示
{$IFDEF IOS}
    WebBrowser1.Navigate('file://' + sPath);
{$ENDIF}
{$IFDEF ANDROID}
    Intent := TJIntent.Create;
    Intent.setAction(TJIntent.JavaClass.ACTION_VIEW);
    Intent.setDataAndType(StrToJURI('file://' + sPath),
          StringToJString('application/pdf'));
    SharedActivity.StartActivity(Intent);
{$ENDIF}

  finally
    //破棄
    msPDF.Free;
    smCon.Free;
  end;
end;
```

3.まとめ

3.まとめ

- アップテザリング機能を組み込むことで、既存のアプリケーション とモバイルアプリケーションが簡単に連携できる。

- アップテザリングを使ったモバイルアプリケーションはDataSnap サーバも不要なので、モバイル開発入門として簡単に取り組める。

- モバイルアプリケーションの帳票機能は、FastReportを使えば C/Sアプリケーションと同様に開発できる。

- iOSとAndroidではPDFの扱い方が異なるので、実装には注意が必要。