Developer Camp

【C4】Delphi/C++テクニカルセッション

「多様化するスマートデバイスをビジネスアプリに活用するためのアーキテクチャと開発のヒント」

Embarcadero Technologies, Inc.
Manager, QA, Documentation, and Localization
新井 正広

はじめに

このセッションの内容

- キーワード
  - スマートデバイス
  - ビジネスアプリ
- 企業内で使われるアプリ
  - データとの連携 → データベース接続
- このセッションでフォーカスすること
  - データベース接続以外で「スマートデバイス利用」を念頭に「データと連携する」技術

スマートデバイス対応の方向性

Delphi / C++Builder ベースの典型的なアプリ

- 企業情報システム
  - Windows ベースのクライアント
    - キーボード、マウス、高解像度モニタ、ネットワークなどの入出力
  - 多機能なユーザーインターフェース
  - バックエンドのデータベースへの接続
  - 帳票などへの出力
- 科学技術計算など、CPU パワーが重要なもの
- 不特定多数のユーザーに配布するもの
  - インストーラ不要（実行形式ファイルのみ配布）

モバイルへの対応

- モバイルならではの活用方法はあるのか？
  - モバイルの「機能」を利用することが「目的」ではない
- 例えば
  - カメラ
  - GPS
  - （ほぼ）どこでもネットワークに接続（3G / 4G / LTE)
  - プッシュ通知

モバイル展開のアイデア

- 在庫管理システムの端末
  - 商品情報をバーコード撮影
  - 不良品の写真を撮影
  - 商品の情報を、画像とともに登録

- 自動車保険(事故現場のレポート)
  - 事故現場、車両の写真
  - 位置情報
  - 相手の運転免許の写真
  - 携帯ネットワーク経由で事故情報のレポート

モバイル展開のアイデア

- 営業支援システム
  - 訪問先の位置情報を利用して
  - 訪問記録の登録
    - 手入力するのではなく、位置情報から絞り込み
  - 近所の案件情報の検索
    - ついでに訪問できる案件は？
  - 緊急案件を会社から通知

モバイル展開のアイデア

- どのような機能を利用できるかリサーチ
- どのような機能を組みこくことが妥当か検証

モバイルへの対応

- Delphi / C++Builder でモバイル対応！
- では、同じ機能をそのまま移植？
  - 画面サイズの違い
  - キーボード、マウスなどの入力デバイスの違い
  - データベースへの接続
    - クライアントライブラリの有無
  - ネットワーク環境の違い
  - 帳票への出力は？
- モバイル端末上で同じユースケースを考えるのか？

どのような機能を実装するにせよ。。。

- UI とロジック、データアクセスの分離
  - データモジュールを利用する
    - デスクトップ向けとモバイル向けは UI の構成が大きく異なる
    - アプリケーション間での共通機能の共有
- 複数のフォームファクターのサポート
  - フォームの継承

マルチプラットフォーム環境向けのアプリケーションの構造

- バックエンドとの接続は「データモジュール」へ
  - データモジュール自体は初期の Delphi からある仕組み
  - 同じプラットフォームでもデバイスごとに UI を最適化
- テストプロジェクトの作成も視野に
- Prototype からの置き換え

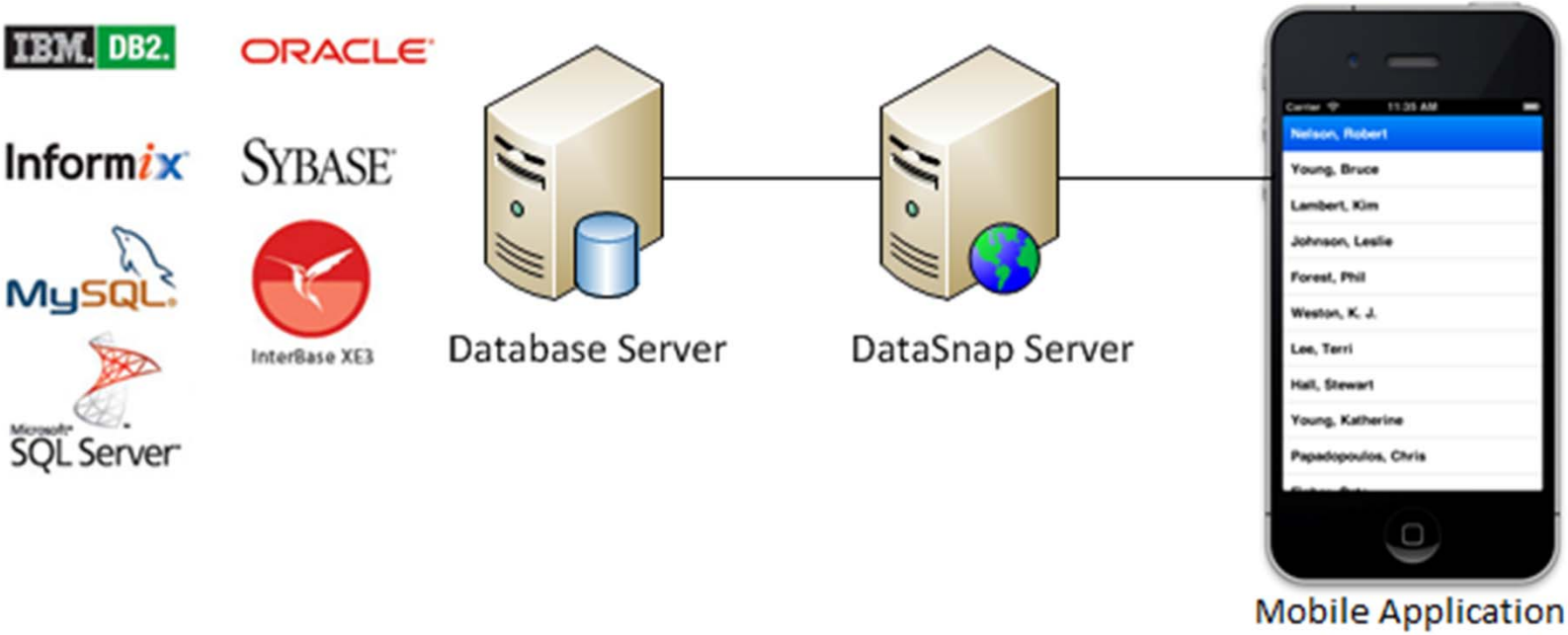
「データと連携する」技術の一般的な選択肢

「スマートデバイス」を念頭に「データと連携」するための選択肢

- DataSnap
  - バックエンドデータベースをDataSnapでカプセル化
- アプリケーション テザリング
  - アプリケーション間で、アクション・データを共有
    - リモートマシンで公開されているアクションの呼び出し
    - データ・ストリームの送受信
- REST (Representational state transfer)
  - Google Apps, Salesforce など、主要なプラットフォームがREST での接続をサポート
- BaaS (Backend As Service) サービスの活用

DataSnap

- 既存のデータベースに接続するのであれば、開発は最も容易



アプリケーション テザリング

アプリケーション テザリング

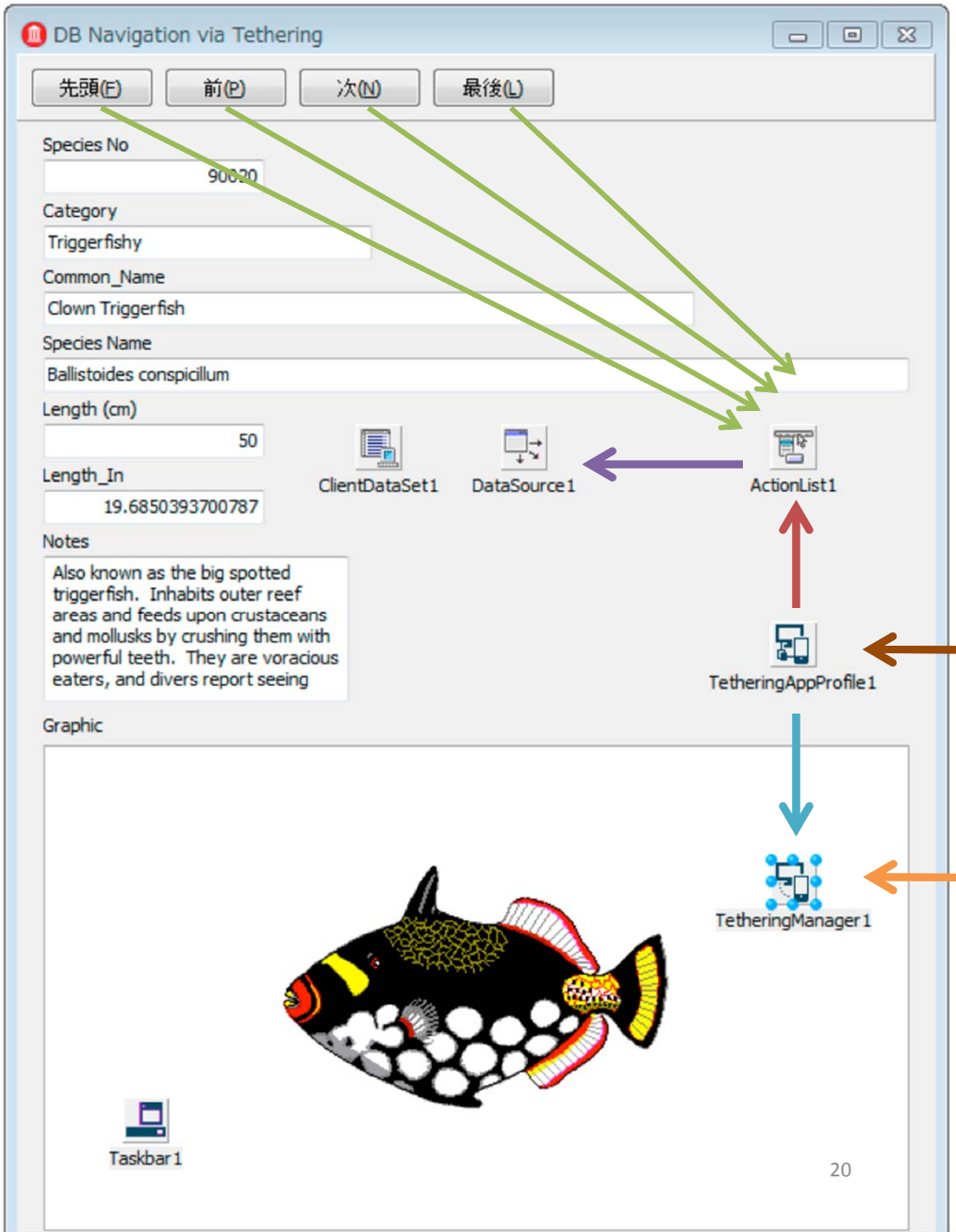
- アプリケーション間で
  - アクションを共有
  - データ・ストリームの送受信
- 利用方法
  - デスクトップアプリケーションをリモート制御
  - デスクトップアプリケーションへの入力支援
    - カメラなど
    - バーコードの読み取り
  - モバイル端末とのデータの共有

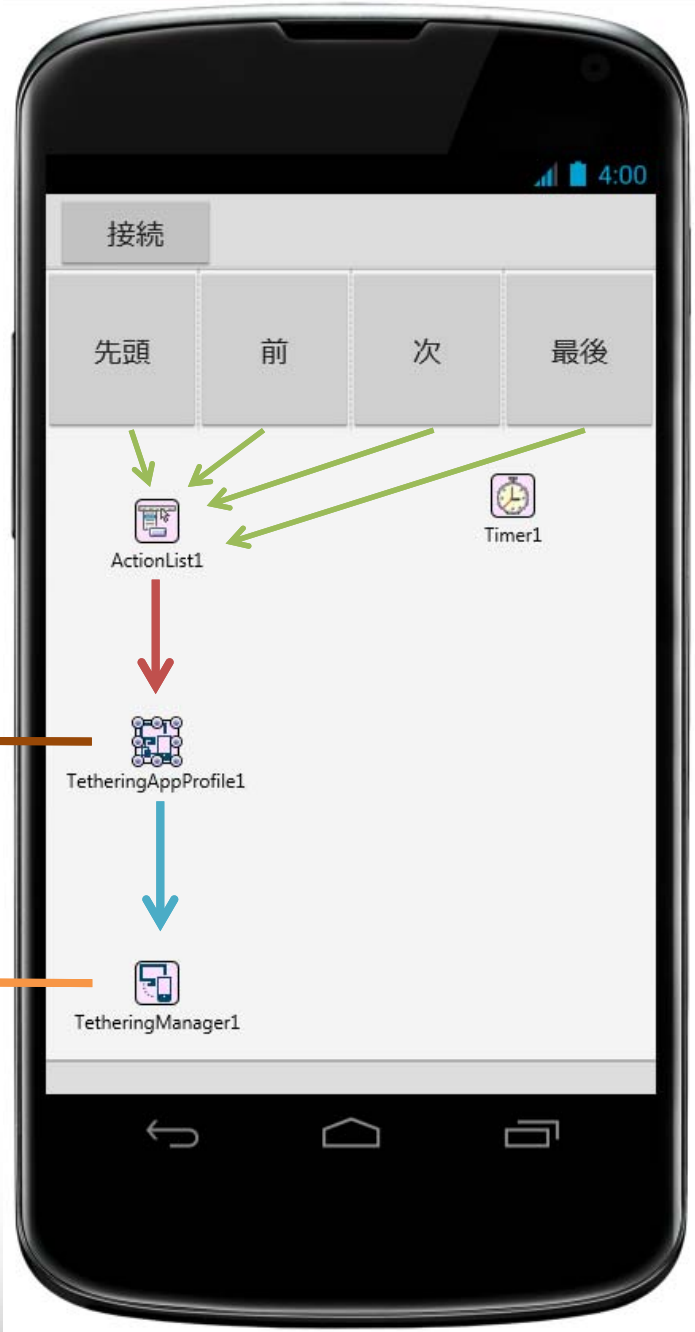
アプリケーション テザリング

- 同一サブネット上で動作するアプリ同士の連携
  - 連携するプラットフォームは問わない
  - 1:n 接続でも構わない

アプリケーション テザリング

- テザリングを通じた機能の公開
  - TTetheringManager
    - 同一サブネット上にある TTetheringManager を検出
    - パスワードによるアプリケーションの認証
  - TTetheringAppProfile
    - アプリケーションが所属するグループを定義
    - アプリケーションが公開／利用するアクションを定義
    - アプリケーションが公開／利用するリソースを定義
      - データ（文字列、数値など）
      - ストリーム


DB Navigation via Tethering
先頭(F)
前(P)
次(N)
最後(L)
Species No
90620
Category
Triggerfishy
Common_Name
Clown Triggerfish
Species Name
Ballistoides conspicillum
Length (cm)
50
Length_In
19.6850393700787
Notes
Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing
Graphic
ClientDataSet1
DataSource1
ActionList1
TetheringAppProfile1
TetheringManager1
Taskbar1


4:00
接続
先頭
前
次
最後
ActionList1
Timer1
TetheringAppProfile1
TetheringManager1

リモートアプリケーションの検出と接続

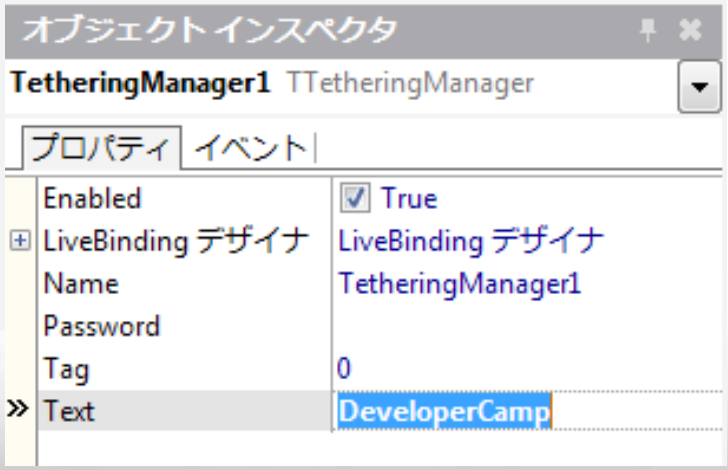
- グループ名を定義し自動的にペア設定
  - TTetheringAppProfile.Group プロパティを設定
  - TTetheringManager.AutoConnect() を呼び出し
- 接続パスワードを設定
  - TTetheringManager.Password プロパティを設定
  - 設計時に静的に設定 or 実行時に動的に設定
- 手動でリモートアプリケーションに接続

1) TTetheringManager を探す

```
procedure TFmxTetheringAppForm.RefreshManagers;
var
  i: Integer;
begin
  // 既に接続している TTetheringManager があれば、ペア設定を解除
  for i := TetheringManager1.PairedManagers.Count - 1 downto 0 do
    TetheringManager1.UnPairManager
      (TetheringManager1.PairedManagers[i].ManagerIdentifier);

  // 新たに TTetheringManager を探す
  TetheringManager1.DiscoverManagers();
end;
```

2) TTetheringManager とペア設定する

```
procedure TFmxTetheringAppForm.TetheringManager1EndManagersDiscovery
  (const Sender: TObject;
   const RemoteManagers: TTetheringManagerInfoList);
var
  RemoteManager: TTetheringManagerInfo;
begin
  for RemoteManager in RemoteManagers do
  begin
    // 接続先の TTetheringManager のプロパティを確認してから接続
    if RemoteManager.ManagerText = 'DeveloperCampDemo' then
      TetheringManager1.PairManager(RemoteManager);
  end;
end;
```

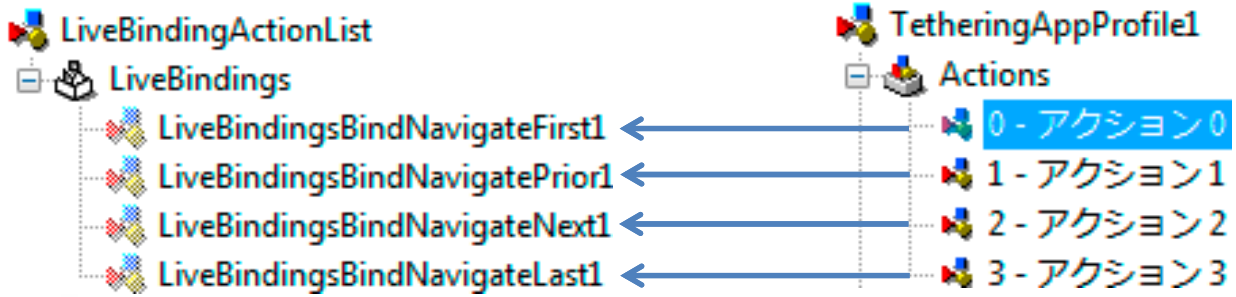


3) TTetheringAppProfile と接続する

```
procedure TFmxTetheringAppForm.TetheringManager1EndProfilesDiscovery
 (const Sender: TObject;
  const RemoteProfiles: TTetheringProfileInfoList);
var
  RemoteProfile: TTetheringProfileInfo;
begin
  // 接続先の TTetheringAppProfile のプロパティを確認してから接続する
  for RemoteProfile in RemoteProfiles do
  begin
    // リモート側の TTetheringAppProfile.Text は自分で事前に設定しておく
    if RemoteProfile.ProfileText = 'Glass 1' then
       TetheringAppProfile1.Connect(RemoteProfile);
  end;
end;
```

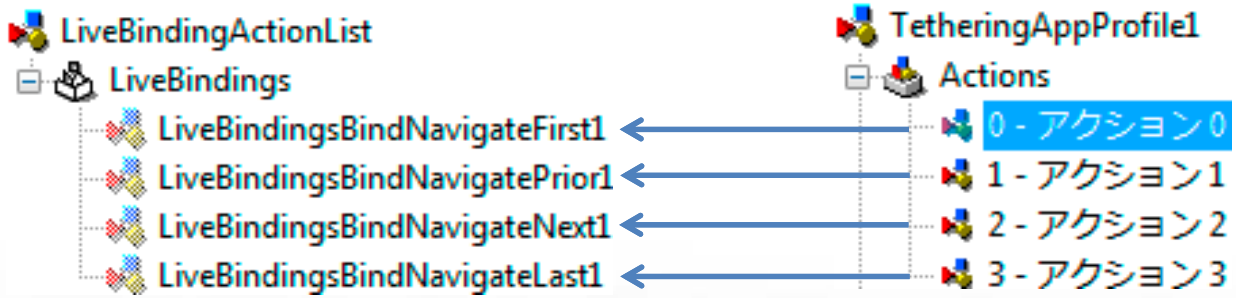
アプリケーション テザリング

- アクションの公開
  - アプリケーションでアクションを使用
  - TTetheringAppProfile に公開するアクションを定義
  - 既に定義されているアクションと TTetheringAppProfile で新たに定義したアクションを接続



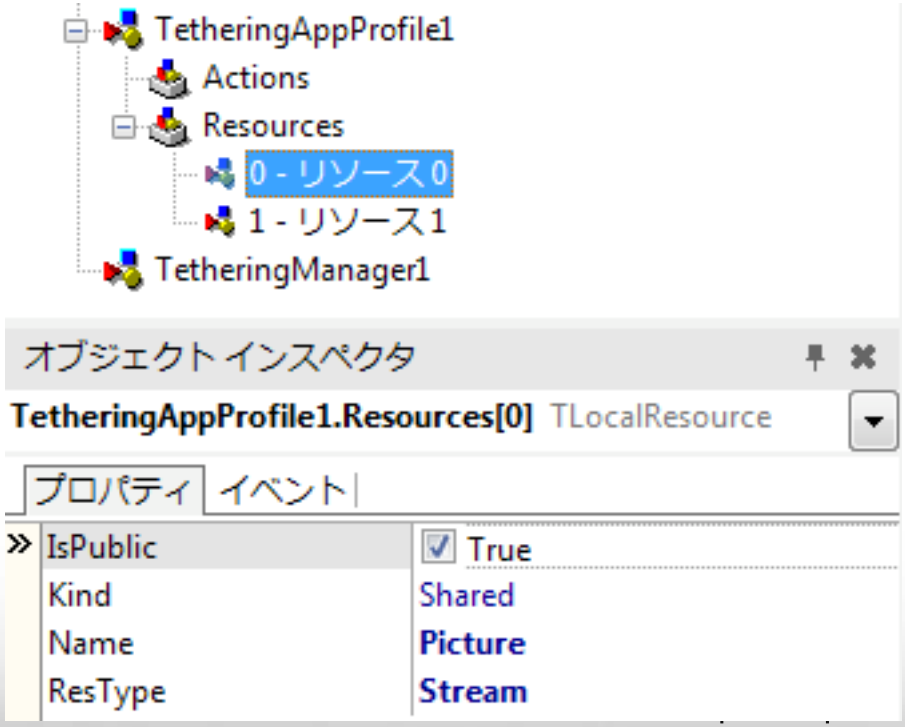
アプリケーション テザリング

- 公開されているアクションを利用
  - 公開するアクションに相当するアクションを定義
  - TTetheringAppProfile に利用するアクションを定義
    - Kind プロパティを Mirror に変更
  - 定義したアクションと TTetheringAppProfile に定義したアクションを接続



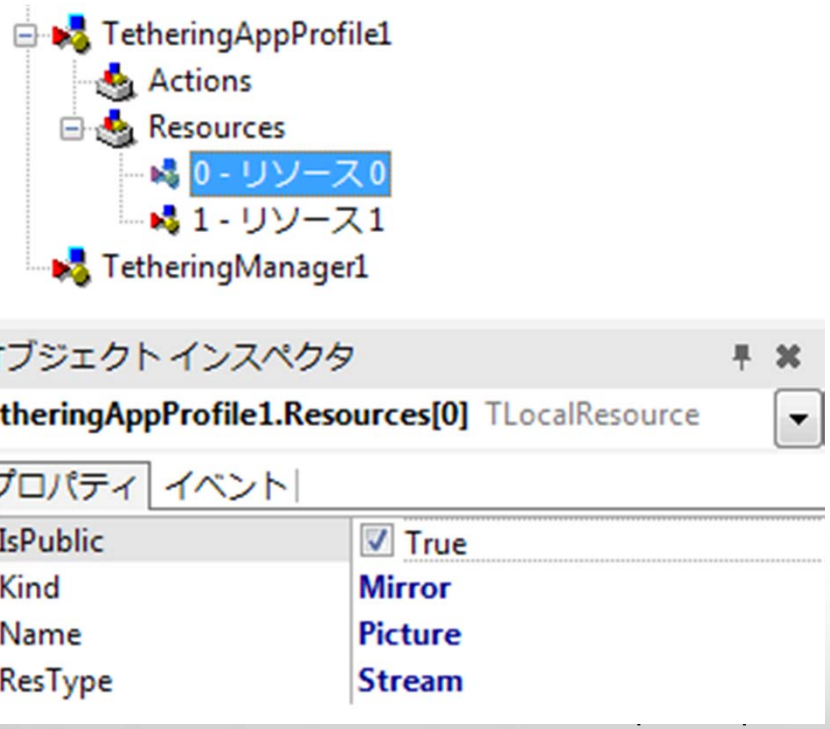
共有リソースの定義

- 共有できるリソース
  - データ
    - 文字列
    - 数値 (Integer, Double, Int64)
    - 真偽値 (Boolean)
  - ストリーム



共有されたリソースを取得する

- 同名のResources[] プロパティを作成する
  - Kind プロパティを Mirror に設定する
  - Kind プロパティ以外は同じ値に設定しておく



4) リソースを公開する

```
procedure TFmxTetheringAppForm.ButtonSendImageClick
  (Sender: TObject);
begin
  if not Assigned(FStream) then
    FStream := TMemoryStream.Create;

  ImageToSend.Bitmap.SaveToStream(FStream);

  FStream.Position := 0;
  TetheringAppProfile1.Resources.FindByName('Picture').Value
  := FStream;
end;
```

5) リソースの変更を取得する

```
procedure TMobileForm.TetheringAppProfile1PictureResourceReceived
  (const Sender: TObject; const AResource: TRemoteResource);
begin
  ImageReceivedPicture.Bitmap.LoadFromStream
  (AResource.Value.AsStream);
end;
```

REST

REST
(Representational State Transfer)

- ステートレスな接続
  - HTTP / HTTPS による接続
- URI でリソースを特定
  - http://example.com/users/masahiro/
- メソッド（動詞）で動作を規定
  - GET, POST, PUT, DELETE
- JSON, XML などのデータを利用
- 多くのベンダーが開発者向けのAPIをRESTで提供
  - Google, Amazon, Salesforce, JIRA,

「つながる」仕組みとしての REST

- メリット
  - 構造が単純
  - 多くのベンダーが REST をサポート
  - プラットフォーム非依存
- 難しい点
  - 実装の詳細はベンダー依存
    - ユーザー認証などの実装はベンダーごとに異なる
    - 一般的なデータベース操作などに対する標準が無い
    - コードのベンダー間での使い回しは（ちょっと）難しい

「つながる」仕組みとしての RESTの利用例

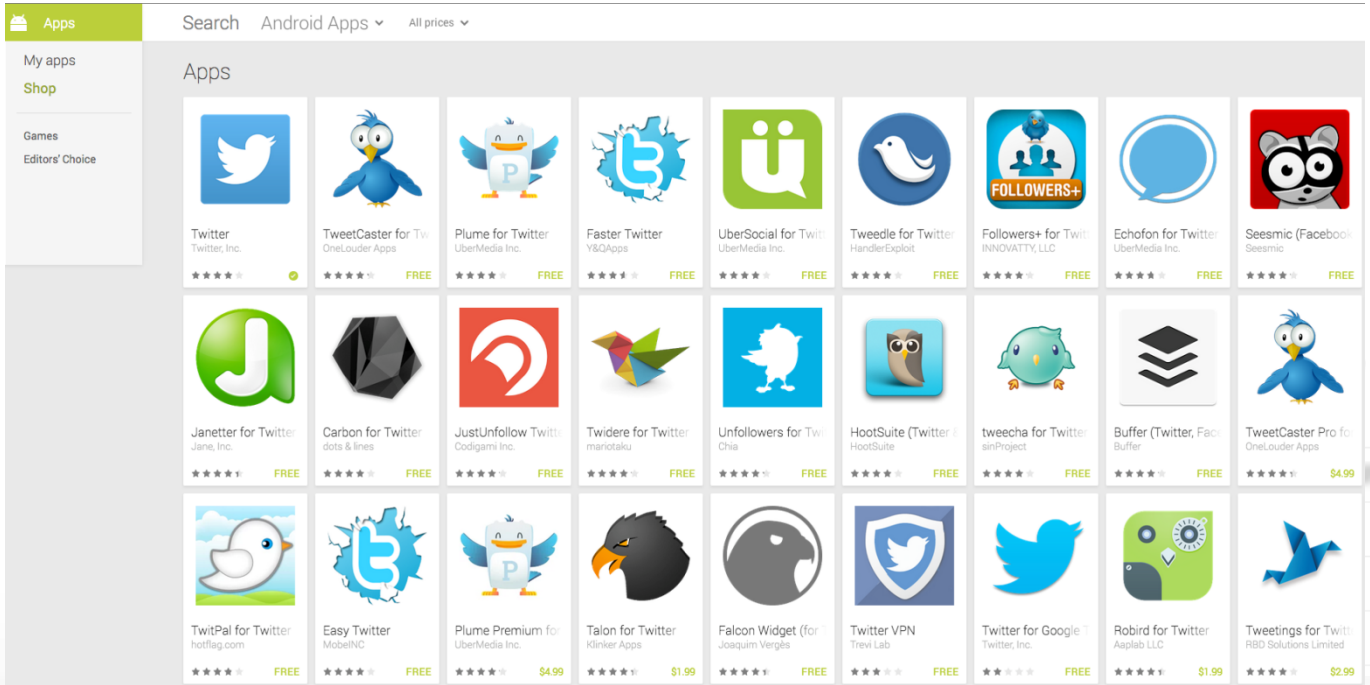
- 企業の情報システム
  - Salesforce.com
- クラウドサービス
  - Google Apps, Amazon, Azure
- ソーシャルネットワーク
  - Facebook, Twitter
- データベース製品
- 変わったところでは、
  - Jira, MediaWiki, Jenkins

REST サービスを提供する

- RESTful な DataSnap サーバーとして実装する。
- メリット
  - TCP/IP 経由接続とほぼ同じ手順でサーバーを作成
  - クライアントもウィザードで生成
  - Delphi / C++Builder 以外のクライアントも、容易にサポート可能

REST を使用して、クラウドサービスに接続する

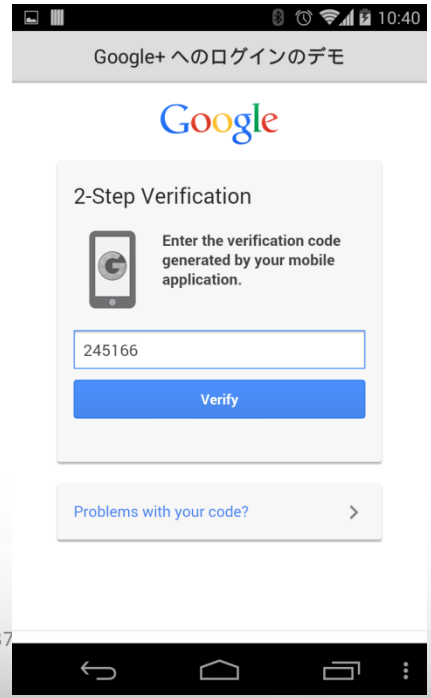
- クラウドサービスを使用する場合の手順
  - 開発者として登録
  - アプリケーションごとにApplication ID, Secretを取得

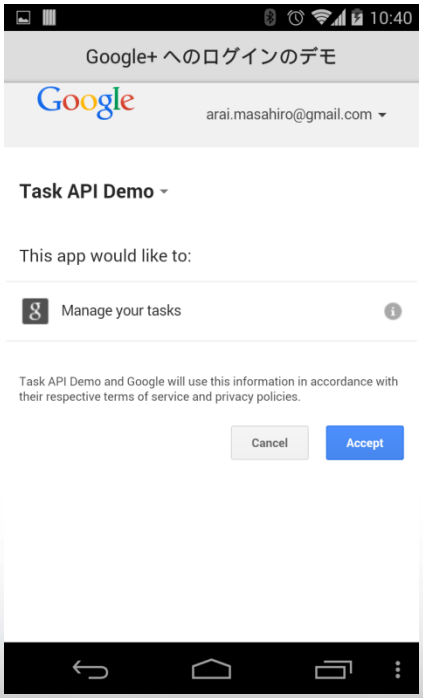


https://play.google.com/store/search?q=twitter&c=apps

REST を使用して、クラウドサービスに接続する

- Application ID, Secret はあくまでも、アプリの識別
- 個人情報にアクセスする場合
  - ユーザー名などはアプリで管理しない
  - O-Auth 2.0 などの仕組みを利用してユーザー認証





BaaS (Backend As Service) サービスの活用

BaaS の活用

- モバイルアプリで一般的に利用される機能をまとめて提供
  - データ管理
  - ユーザー管理・アクセス管理
  - プッシュ通知
- モバイル、デスクトップの両方からアクセス可能
- 自前で公開サービスを作成する必要なし
  - ユーザー環境におけるセキュリティ対策を簡略化

Baas プロバイダーへの接続

TKinveyProvider

TParseProvider

TBackgroundUser　ユーザーアカウント

TBackgroundQuery　データの取得

TBackgroundStrage　抽象化されたデータアクセス

TBackgroundFile　ファイル保存

TBackgroundPush　プッシュ通知のリクエスト

TPushEvents　プッシュ通知イベント

サンプルデータ

_id	ID	Speaker	Title	metadata (_kmd)	access control (_acl)
"5350593ac3ff615f53000c40"	"C1"	"細川 淳"	"マルチデバイスの荒海にこぎ出す新人エンジニアのためのソフトウェア開発の心得"	{"lmt":"2014-04-17T22:44:34.234Z","ect":"2014-04-17T22:44:10.760Z"}	{"creator":"kid_Pe4
"5350595d9c1129c035000c74"	"A1"	"藤田 和宏"	"Delphiによる基幹系システム連携の具体例 ～ SAPシステムとの枯れた連携手法"	{"lmt":"2014-04-17T22:45:06.884Z","ect":"2014-04-17T22:44:45.457Z"}	{"creator":"kid_Pe4
"535059769c1129c035000c84"	"C2"	"富永 英明"	"業務アプリ開発で活かせるDelphiプログラミングテクニック"	{"lmt":"2014-04-17T22:45:31.244Z","ect":"2014-04-17T22:45:10.796Z"}	{"creator":"kid_Pe4
"53505994239cd57c21000b9c"	"A2"	"高橋 智宏"	"C++開発者のための最新プログラミングエッセンス"	{"lmt":"2014-04-17T22:46:05.321Z","ect":"2014-04-17T22:45:40.202Z"}	{"creator":"kid_Pe4
"535059afc3ff615f53000c69"	"G3"	"スティーブ・ヘイニー・新井 正広"	"さらに進化を遂げるエンバカデロのマルチデバイス開発環境」"	{"lmt":"2014-04-17T23:55:52.600Z","ect":"2014-04-17T22:46:07.543Z"}	{"creator":"kid_Pe4
"535059d4c3ff615f53000c76"	"C4"	"新井 正広"	"多様化するスマートデバイスをビジネスアプリに活用するためのアーキテクチャと開発のヒント"	{"lmt":"2014-04-17T22:47:03.487Z","ect":"2014-04-17T22:46:44.911Z"}	{"creator":"kid_Pe4
"535059ec9c1129c035000cb0"	"A4"	"伊賀 敏樹"	"Visual C++ユーザーもバッチリ！C++Builderによるマルチデバイス開発"	{"lmt":"2014-04-17T22:47:24.689Z","ect":"2014-04-17T22:47:08.502Z"}	{"creator":"kid_Pe4
"53505a039c1129c035000cbc"	"C5"	"高田 茂人／福井 翔一"	"実例！Delphiによる多層分散型基幹業務システム構築の課題と解決のポイント"	{"lmt":"2014-04-17T22:47:46.284Z","ect":"2014-04-17T22:47:31.109Z"}	{"creator":"kid_Pe4
"53505a16c3ff615f53000c92"	"A5"	"長沢 智治"	"RAD Studio で実践する継続的インテグレーション ～ アプリとデベロッパーの価値を拡張するエッセンス"	{"lmt":"2014-04-17T22:48:06.970Z","ect":"2014-04-17T22:47:50.870Z"}	{"creator":"kid_Pe4
"53505a2c9c1129c035000cd7"	"G6"	"はやし つとむ"	"共有！みんなの開発事例、開発経験、テクニック"	{"lmt":"2014-04-17T22:48:27.031Z","ect":"2014-04-17T22:48:12.955Z"}	{"creator":"kid_Pe4

TBackend*コンポーネントを利用する

- データの作成

```
  FEntity1, FEntity2: TBackendEntityValue;
//---------------------------------
Var
  SessionData: TSessionData;
begin
  SessionData := TSessionData.Create;
  try
    SessionData.ID      := EditID.Text;
    SessionData.Title   := EditTitle.Text;
    SessionData.Speaker := EditSpeaker.Text;
    BackendStorage1.Storage.CreateObject<TSessionData>
      ('Session', SessionData, FEntity1);
  finally
    SessionData.Free;
  end;
end;
```

TBackend*コンポーネントを利用する

- データの参照

```
var
  SessionData: TSessionData;
  ObjectList: TBackendObjectList<TSessionData>;
  Query: TArray<string>;
begin
  Query := TArray<string>.Create('sort=ID'); //Kinvey
  ObjectList := TBackendObjectList<TSessionData>.Create;

  BackendStorage1.Storage.QueryObjects<TSessionData>(
    'Sessions', Query, ObjectList);
  for SessionData in ObjectList do
  begin
    LabelID.Text      := SessionData.ID;
    LabelTitle.Text   := SessionData.Title;
    LabelSpeaker.Text := SessionData.Speaker;
  end;
```

TBackend*コンポーネントを利用する

- データの更新

```
  FEntity1, FEntity2: TBackendEntityValue;
//---------------------------------
Var
  SessionData: TSessionData;
begin
  SessionData := TSessionData.Create;
  try
    SessionData.ID      := EditID.Text;
    SessionData.Title   := EditTitle.Text;
    SessionData.Speaker := EditSpeaker.Text;
    BackendStorage1.Storage.UpdateObject<TSessionData>
      (FEntity1, SessionData, FEntity2);
  finally
    SessionData.Free;
  end;
end;
```

TBackend*コンポーネントを利用する

- データの削除

```
  FEntity1, FEntity2: TBackendEntityValue;
//---------------------------------------
begin
  BackendStorage1.Storage.DeleteObject(FEntity2);
end;
```

TBackend*コンポーネントを利用する

- ファイルのアップロード

```
var
  AFile: TBackendEntityValue;
  Stream: TFileStream;
begin
  Stream := TFileStream.Create('template.xml', fmOpenRead);
  try
    BackendFiles1.Files.UploadFile(
     'template.xml',       // BaaS サービス上でのファイル名
     Stream,               // アップロードするストリーム
     'application/xml',    // Content-Type
     AFile);               // [out] アップロードしたファイルの情報
  finally
    Stream.Free;
  end;
end;
```

TBackend*コンポーネントを利用する

- ファイルのダウンロード・消去

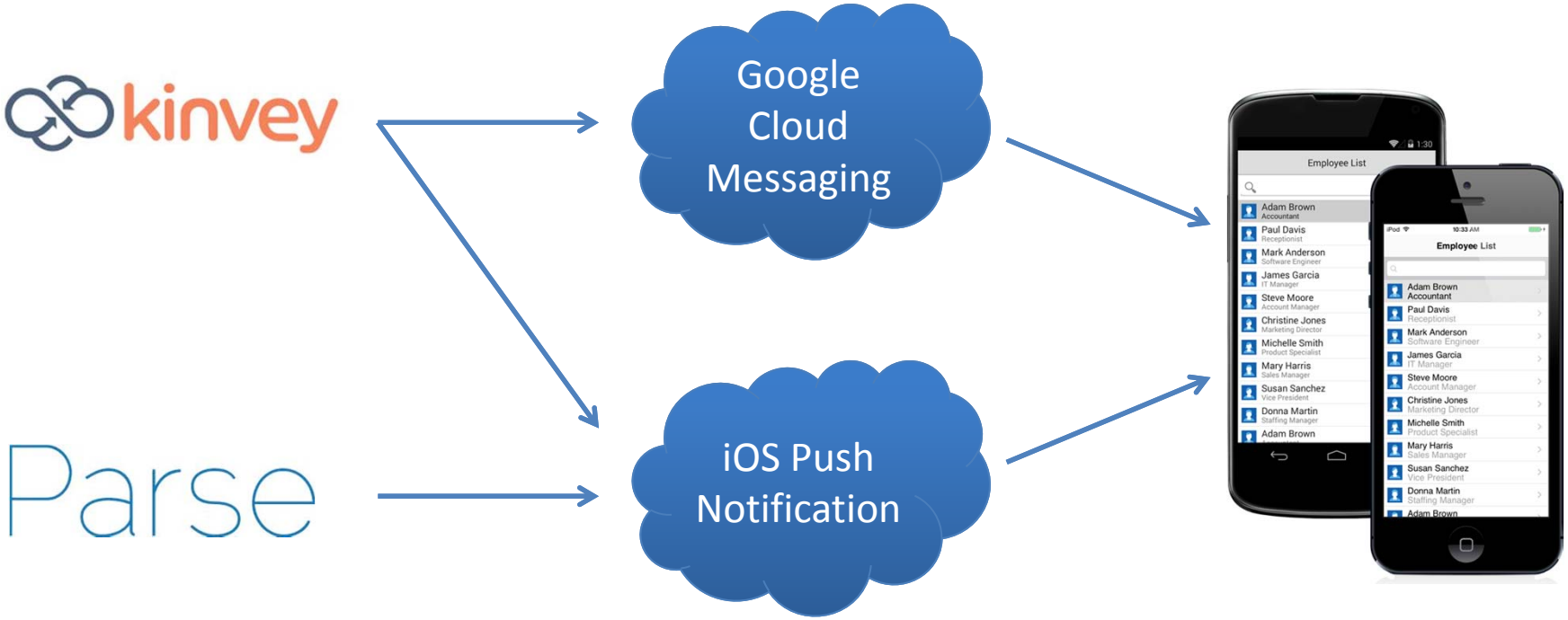
```
var
  AFile: TBackendEntityValue;
  Stream: TFileStream;
begin
  // BackendFiles1.Files.UploadFile(…);

  Stream := TMemoryStream.Create;
  try
    TDownloadURL.DownloadRawBytes(AFile.DownloadURL, Stream);
    Stream.SaveToFile('template2.xml');
  finally
    Stream.Free;
  end;

  BackendFiles1.Files.DeleteFile(AFile);
end;
```

プッシュ通知の利用

- Parse あるいは Kinvey を経由した接続
- iOS および Android 向けの通知をサポート
  - 通知センターへのメッセージ送信
  - アプリケーションの参照を促す

- 単純なメッセージの送信
- データーの変更によるメッセージの自動送信

「プッシュ通知」を利用する



※個々のサービスに対するアカウントが必要

「プッシュ通知」の受信

- データの受信そのものは単純

```
procedure TForm1.PushEvents1PushReceived(Sender: TObject;
  const AData: TPushData);
begin
  ListView1.Items.Add.Text := AData.Message;
end;
```

- しかし、アプリは休止している可能性が高い

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  if Assigned(PushEvents1.StartupNotification)
     and (PushEvents1.StartupNotification.Message <> '') then
    ListView1.Items.Add.Text
     := PushEvents1.StartupNotification.Message;
end;
```

まとめ