Developer Camp

Delphi/C++Builder**テクニカルセッション**

「マルチデバイス対応アプリ開発の勘所」

エンバカデロ・テクノロジーズ
エヴァンジェリスト 高橋智宏

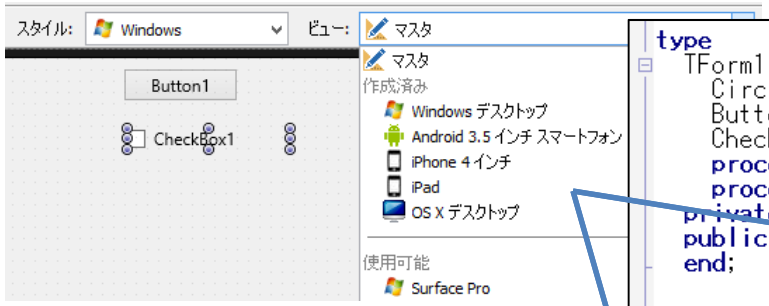
www.embarcadero.com/jp

アジェンダ

- FireUIマルチデバイスデザイナ
  - iPhone 6(Plus), Android Wear向けビュー, Yosemite(10.10)スタイル, Lollipopの新ART
- メッセージボックス系はモバイルで要注意
  - DelphiとC++言語で無名メソッドを利用(Non-Blocking & Blocking)
- Android APIにアクセス
  - Java2OP.exeの使い方(C++/Delphi), C++言語でGPSのリスナーを記述!?
- SQLite + FireDAC + Visual LiveBinding
  - プロジェクトマネージャと配置マネージャの関係
  - マルチプラットフォーム(Win/Mac/iOS/Android)で試す
- HTTP + XML
  - Indy(TIdHTTP), TXMLDocument(Omni XML)
  - XMLデータバインディング(C++/Delphi)が便利
- TRESTxxxxコンポーネント + JSON
  - Indy, TXMLDocument(Omni XML)
  - XMLデータバインディング(C++/Delphi)が便利
- SOAPサーバー & クライアント
  - Delphi(C++)サーバーからクラスインスタンスを返すには?
  - DOMベンダーを「Omni XML」にするには?
- DataSnap RESTサーバー & クライアント
  - データセット1個または複数個を一度に送(受)信するには?

FireUIの基礎

- 「スタイル&マスタ」と「ビュー」
  - マスタに追加したGUIコンポーネントを個別のビューで設計時にカスタマイズしておく
    - カスタマイズした内容は専用の.fmxファイルに保存される
    - 特定のOS/フォームファクタでは Visible := False など
  - .pas, .cppにフォームファクタおよびOS指定したリンク命令が追加される



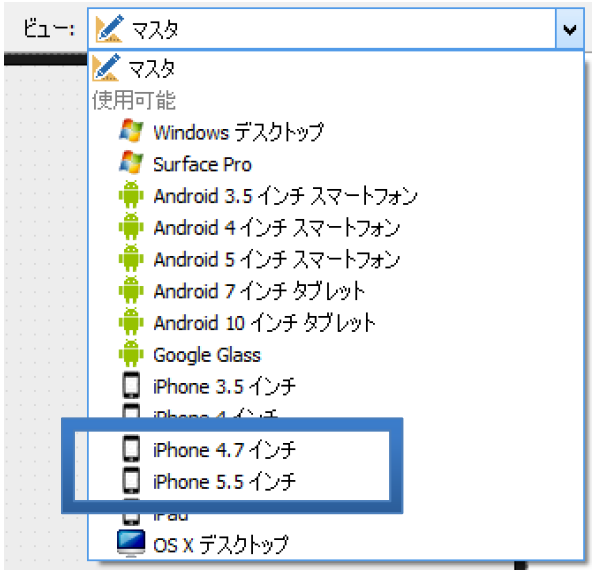

```
type
  TForm1 = class(TForm)
    Circle1: TCircle;
    Button1: TButton;
    CheckBox1: TCheckBox;
    procedure Button1Click(S
    procedure test(Sender: T
  private
  public
  end;

var
  Form1: TForm1;

implementation

{$R *.fmx}
{$R *.Windows.fmx MSWINDOWS}
{$R *.SmXhdpiPh.fmx ANDROID}
{$R *.iPhone4in.fmx IOS}
{$R *.iPad.fmx IOS}
{$R *.Macintosh.fmx _MACOS}

procedure TForm1.Button1Click(Sender: TObject);
begin
  // ...
end;
```

```
#include <fmx.h>
#pragma hdrstop

#include "Unit2.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.fmx"
#pragma resource ("*.Windows.fmx", _PLAT_MSWINDOWS)
#pragma resource ("*.SmXhdpiPh.fmx", _PLAT_ANDROID)
#pragma resource ("*.iPhone4in.fmx", _PLAT_IOS)
#pragma resource ("*.iPad.fmx", _PLAT_IOS)
#pragma resource ("*.Macintosh.fmx", _PLAT_MACOS)
```

個別のビューの作成

- iPhone 6, iPhone 6 Plus向けビュー
  - iPad系は1種類で足りる
  - .fmxのリンク例

```
implementation

{$R *.fmx}
{$R *.iPhone47in.fmx IOS}
{$R *.iPhone55in.fmx IOS}
```



- Android Wear向けカスタムビュー
  - Samsung Gear Live(1.63インチ, 320 x 320, 278ppi)
  - パッケージ(.pas,.bpl)の作成およびIDEへの登録
  - C:¥Users¥[User]¥AppData¥Roaming¥Embarcadero¥BDS¥15.0¥MobileDevices.xmlへ追記し、IDEを再起動
  - アプリプロジェクトに.pasを追加して組み込む


```
unit GearLive;
interface
implementation
uses System.Devices, System.Types, System.SysUtils;

initialization
  TDeviceinfo.AddDevice(TDeviceinfo.TDeviceClass.Watch,      // Watch
                        'GearLive',
                        TSize.Create(320, 320),              // 物理サイズ
                        TSize.Create(213, 213),              // 論理サイズ(hdpi)
                        TOSVersion.TPlatform.pfAndroid,      // Android
                        278);                                // dpi=278(hdpi)
end.
```

最新のOS

- OS X Yosemite(10.10)に対応するには?
  - Yosemite向けスタイル(Yosemite.fsf または YosemiteDark.fsf)をTStyleBookで利用

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  {$IFDEF MACOS}
  if (TOSVersion.Major = 10) and (TOSVersion.Minor = 10) then
    Form1.StyleBook := YosemiteStyleBook;
  {$ENDIF}
end;
```

- 残念ながら、XE6(5)のAndroidアプリは、Lollipop 5.0(新ART) とKitKat 4.4の実験的ARTモードでは起動しません
  - 最新のXE7を使用する必要があります
- iOS 8.0, 8.1
  - 特別なスタイルは不要, TEditのiOSネイティブモードも利用可
- Windows 10 !?

iOS,Androidでは要注意

- XE7のiOS,Androidでは、UIイベントのフリーズを防ぐため、慣れ親しんだ以下の4つのメソッドは、呼び出すと直ちに呼び出し側に戻ります(Non-Blocking)
  - ShowMessage
    - ウィンドウを閉じるタイミングを知るには、MessageDlgへの移行が必要
  - MessageDlg
  - InputQuery(※複数項目の入力はiOSは未対応)
  - InputBox

RAD Studio XE7のオンラインヘルプより
- 「Delphi XE7 および C++Builder XE7 の新機能 - ダイアログ ボックスのメソッドでは閉じる際の処理を行う無名メソッドをサポート」

Delphiの場合

- 無名メソッドで閉じるイベントを捕捉

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  MessageDlg('Hello from MessageDlg',
             TMsgDlgType.mtError, [TMsgDlgBtn.mbOK, TMsgDlgBtn.mbCancel], 0,
             procedure(const AResult: TModalResult)
             begin
               if AResult = mrOk then
                  // handle ok press
               else if AResult = mrCancel then
                  // handle cancel press
             end);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  InputQuery('Input Data',
             ['First Name', 'Last Name', 'Extra Name'],
             ['John','Doe','Finch'],
             procedure (const AResult: TModalResult; const AValues: array of string)
             var
               S, Msg: String;
             begin
               Msg := '';
               for S in AValues do
                 Msg := Msg + S + ' ';
               ShowMessage(Format('%d: %s', [AResult, Msg]));
             end);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  InputBox('Caption', 'Prompt', 'Default',
           procedure(const AResult: TModalResult; const AValue: string)
           begin
             if AResult = mrOk then
                // handle ok press
             else if AResult = mrCancel then
                // handle cancel press
           end);
end;
```

C++言語の場合

- TCppInterfacedObjectの派生クラスでoverrideしたInvokeメソッドを使用

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  class TMyInputCloseDialogProc : public TCppInterfacedObject<TInputCloseDialogProc> {
  public:
    void __fastcall Invoke(const TModalResult AResult) {
      if(AResult == mrOk) {
      }
      else {
      }
    }
  };
  MessageDlg(u"Hello from MessageDlg", TMsgDlgType::mtError,
             TMsgDlgButtons() << TMsgDlgBtn::mbOK << TMsgDlgBtn::mbCancel, 0,
             new TMyInputCloseDialogProc());
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
  const UnicodeString APrompts[3] = {u"First Name", u"Last Name", u"Extra Name"};
  UnicodeString ADefaultValues[3] = {u"John",u"Doe",u"Finch"};
  class TMyInputCloseQueryProc : public TCppInterfacedObject<TInputCloseQueryProc> {
  public:
    void __fastcall Invoke(const TModalResult AResult,
                           UnicodeString const *AValues, const int AValues_High) {
      UnicodeString S, Msg = u"";
      for(int i = 0; i <= AValues_High; i++) {
        S = AValues[i];
        Msg = Msg + S + u" ";
      }
      ShowMessage(IntToStr(AResult) + u": " + Msg);
    }
  };
  InputQuery(u"Input Data", APrompts, 2, ADefaultValues, 2, new TMyInputCloseQueryProc());
}

void __fastcall TForm1::Button4Click(TObject *Sender)
{
  class TMyInputCloseBoxProc : public TCppInterfacedObject<TInputCloseBoxProc> {
  public:
    void __fastcall Invoke(const TModalResult AResult, const UnicodeString AValue) {
      if(AResult == mrOk) {
        ShowMessage(AValue);
      }
      else if(AResult == mrCancel) {
      }
    }
  };
  InputBox(u"Caption", u"Prompt", u"Default", new TMyInputCloseBoxProc());
}
```

ブロッキング的フローの例

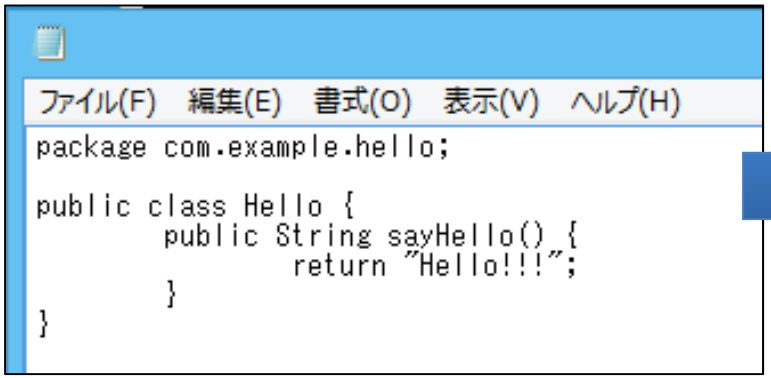
- 複数のShowMessage(と同等のMessageDlg)を順番に表示

```
uses System.SyncObjs;

procedure TForm1.Button5Click(Sender: TObject);
var
  e: TEvent;
begin
  e := TEvent.Create(nil, False, False, '');

  MessageDlg('Step#1', TMsgDlgType.mtCustom, [TMsgDlgBtn.mbOK], 0,
             procedure(const AResult: TModalResult)
             begin
               e.SetEvent;
             end);
  while True do
  begin
    if e.WaitFor(100) = TWaitResult.wrSignaled then
      Break;
    Application.ProcessMessages;
  end;

  MessageDlg('Step#2', TMsgDlgType.mtCustom, [TMsgDlg
             procedure(const AResult: TModalResult)
             begin
               e.SetEvent;
             end);
  while True do
  begin
    if e.WaitFor(100) = TWaitResult.wrSignaled then
      Break;
    Application.ProcessMessages;
  end;

  FreeAndNil(e);
  ShowMessage('done');
end;
```

```
#include <System.SyncObjs.hpp>
void __fastcall TForm1::Button5Click(TObject *Sender)
{
  TEvent* e = new TEvent(NULL, False, False, "", False);

  class TMyInputCloseDialogProc : public TCppInterfacedObject<TInputCloseDialogProc> {
  private:
    TEvent* ev;
  public:
    __fastcall TMyInputCloseDialogProc(TEvent* e) : ev(e) {}
    void __fastcall Invoke(const TModalResult AResult) {
      ev->SetEvent();
    }
  };

  MessageDlg(u"Step#1", TMsgDlgType::mtCustom, TMsgDlgButtons() << TMsgDlgBtn::mbOK, 0,
             new TMyInputCloseDialogProc(e));
  while(true) {
    if( e->WaitFor(100) == TWaitResult::wrSignaled )
      break;
    Application->ProcessMessages();
  }

  MessageDlg(u"Step#2", TMsgDlgType::mtCustom, TMsgDlgButtons() << TMsgDlgBtn::mbOK, 0,
             new TMyInputCloseDialogProc(e));
  while(true) {
    if( e->WaitFor(100) == TWaitResult::wrSignaled )
      break;
    Application->ProcessMessages();
  }

  delete e;
}
```

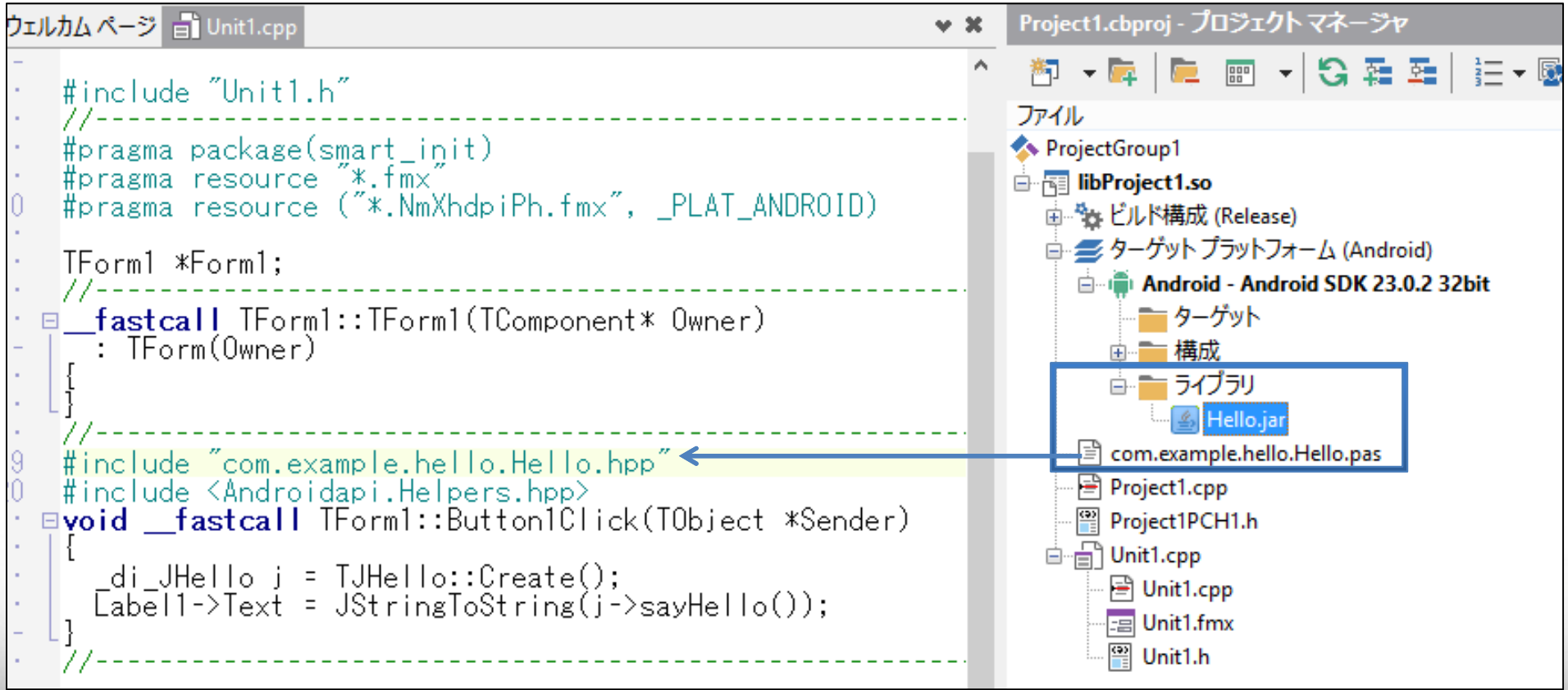
Android APIにアクセス

- 特定のJavaクラスのAPIを呼び出したい場合
  - XE7のJava2OP.exeでブリッジ(.pas)を生成し利用する
  - 例:C:¥>Java2OP.exe -jar Hello.jar -classes com.example.hello.Hello -unit com.example.hello.Hello



Android APIにアクセス(続き)

- C++からも.pasを利用可能
  - .pasをプロジェクトに追加
  - いったんビルドして、生成された.hppをインクルード
  - プロジェクトの[ライブラリ]ノードにHello.jarを追加
  - Hello.jarがマージされたclasses.dexが.apk内にパッケージングされる

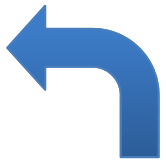


C++アプリ - AndroidでGPSにアクセス

- TLocationSensorコンポーネントで足りない場合
  - Delphi(.pas)でJava向けGPSリスナーを直接記述し、C++からも利用

エルカム ページ | Unit2

```
unit Unit2;

interface

uses
  Androidapi.JNIBridge, Androidapi.JNI.JavaTypes, Androidapi.JNI.Os, Androidapi.JNI.Location;

type
  TMyLocationChangedEvent = procedure(location: JLocation) of object;
  TMyLocationListener = class(TJavaLocal, JLocationListener)
  public
    callback: TMyLocationChangedEvent;
    constructor Create;
    procedure onLocationChanged(location: JLocation); cdecl;
    procedure onProviderDisabled(provider: JString); cdecl;
    procedure onProviderEnabled(provider: JString); cdecl;
    procedure onStatusChanged(provider: JString; status: Integer; extr
  end;

implementation

{ TMyLocationListener }
procedure TMyLocationListener.onLocationChanged(location: JLocation);
begin
  if Assigned(callback) then
    callback(location);
end;

procedure TMyLocationListener.onProviderDisabled(provider: JString);
begin
end;

procedure TMyLocationListener.onProviderEnabled(provider: JString);
begin
end;

procedure TMyLocationListener.onStatusChanged(provider: JString; statu
begin
end;

constructor TMyLocationListener.Create;
begin
  inherited Create;
end;

end.
```

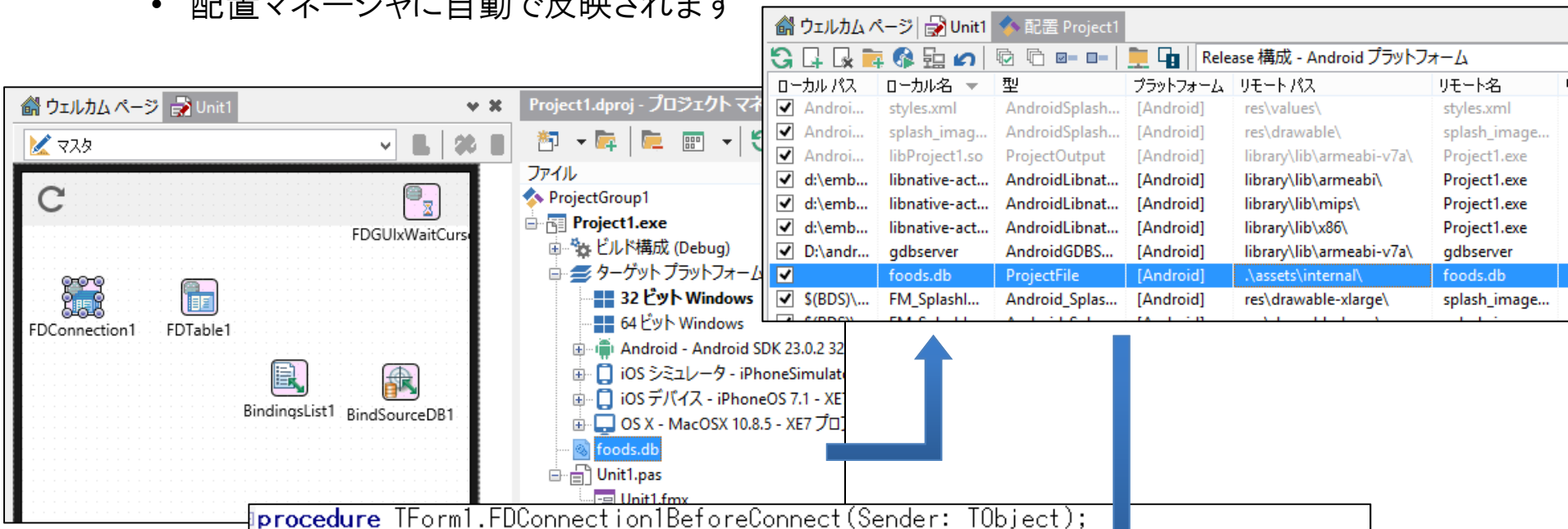


ルカム ページ | Unit1.cpp

```
//---------------------------------------------------------------------------
#include <Androidapi.JNIBridge.hpp>
#include <Androidapi.Helpers.hpp>
#include "Unit2.hpp"
static _di_JLocationManager locman = NULL;
static TMyLocationListener* gpslistener = NULL;
//---------------------------------------------------------------------------
void __fastcall TForm1::TThreadMethod()
{
  Label1->Text = FloatToStr(prev_location->getLatitude()) + L"," +
                 FloatToStr(prev_location->getLongitude());
}
void __fastcall TForm1::MyLocationChangedEvent(Androidapi::Jni::Location::_di_JLocati
{
  prev_location = location;
  TThread::Synchronize(NULL, TThreadMethod);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  if(locman == NULL) {
    locman = TJLocationManager::Wrap(
       ((_di_ILocalObject)SharedActivity()->getSystemService(
            TJContext::JavaClass->LOCATION_SERVICE))->GetObjectID());
  }
  if(gpslistener == NULL) {
    gpslistener = new TMyLocationListener();
    gpslistener->callback = MyLocationChangedEvent;
  }
  locman->requestLocationUpdates(TJLocationManager::JavaClass->GPS_PROVIDER, 0, 0,
                                 *gpslistener, TJLooper::JavaClass->getMainLooper());
}
```

組み込みDBを利用するには?

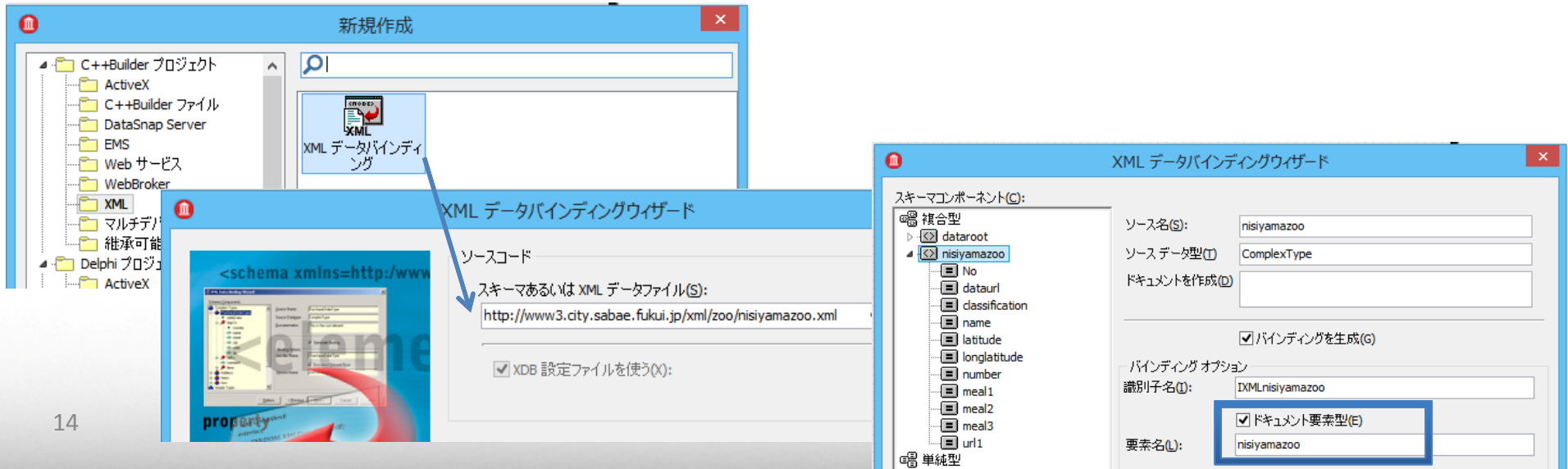
- SQLiteとマルチOS
  - FireDACでアクセスし、Visual LiveBindingで表示
    - DBファイルの物理パスは、実行時にTFDConnectionのBeforeConnectイベントで設定する
  - XE7では、DBファイルをプロジェクトマネージャに直接追加
    - 配置マネージャに自動で反映されます




```
procedure TForm1.FDConnection1BeforeConnect(Sender: TObject);
begin
{$IF defined(IOS) or defined(ANDROID)}
  FDConnection1.Params.Database := TPath.Combine(TPath.GetDocumentsPath, dbname);
{$ELSE}
  FDConnection1.Params.Database := ExtractFilePath(ParamStr(0)) + dbname;
{$ENDIF}
end;
```

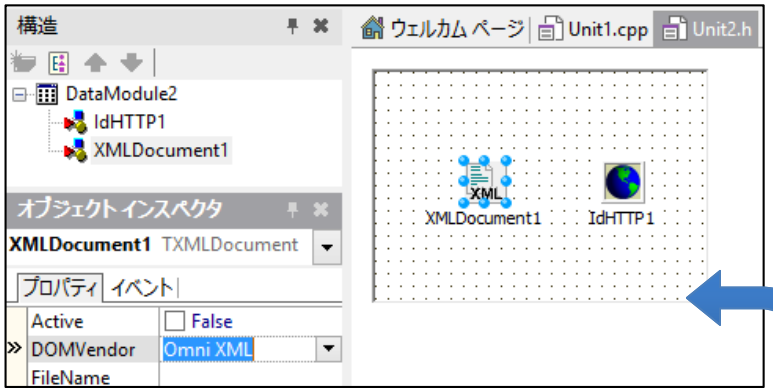
XMLを受信してDOMにアクセス

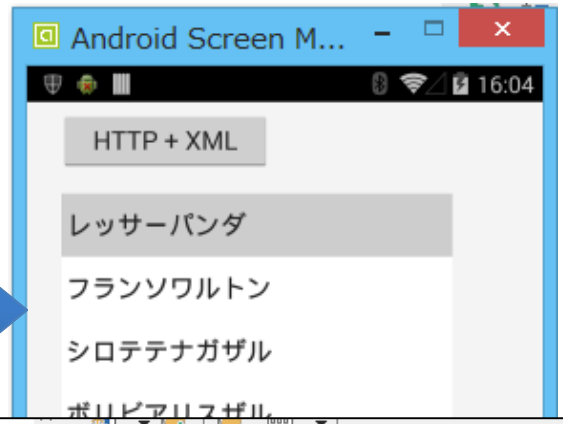
- HTTPクライアントとXML DOMパーサとマルチOS
  - Indy(TIdHTTP)でWebにアクセスし、TXMLDocumentでDOMにアクセス
    - XE7には、マルチOS対応の新しい「Omni XML」を搭載
  - DOMツリーを直接操作するより、より安全・便利な「XMLデータバインディングウィザードが生成するラッパー」がオススメです
    - DelphiおよびC++言語に対応
  - 行政によるオープンデータ化の推進
    - 福井県鯖江市 西山動物園の動物（XML）
      - http://www3.city.sabae.fukui.jp/xml/zoo/nisiyamazoo.xml



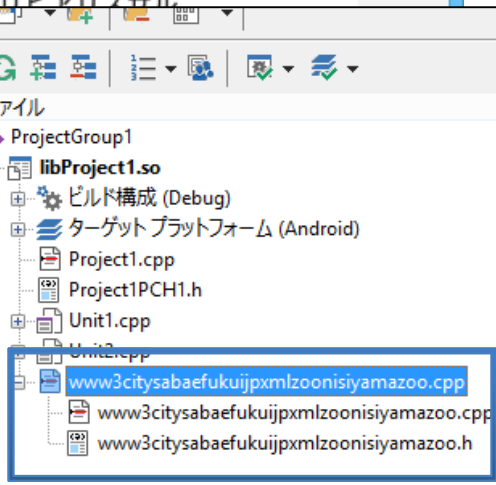
TXMLDocumentの使い方

- DOMVendorプロパティで Omni XML を指定
  - Delphi: uses Xml.omnixmldom が自動追加される
  - C++: #include <Xml.omnixmldom.hpp> が自動追加される






```
#include "www3citysabaefukuijpxmlzoonisiyamazoo.h"
#include <memory>
using namespace std;
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  unique_ptr<TDataModule2> dm(new TDataModule2(NULL));
  unique_ptr<TMemoryStream> bsData(new TMemoryStream());
  dm->IdHTTP1->Get("http://www3.city.sabae.fukui.jp/xml/zoo/nisiyamazoo.xml", bsData.get());
  if( dm->IdHTTP1->ResponseCode == 200)
  {
    dm->XMLDocument1->LoadFromStream(bsData.get(), TXMLEncodingType::xetUTF_8);
    dm->XMLDocument1->Active = True;
    _di_IXMLdataroot root = Getdataroot(dm->XMLDocument1);
    ListBox1->Clear();
    for(int i = 0; i < root->Count; i++)
    {
      _di_IXMLnisiyamazoo animal = root->nisiyamazoo[i];
      ListBox1->Items->Add(animal->name);
    }
  }
}
```



JSON > XML ?

- JSONのほうがXMLよりも構造がシンプル
- 一般的にエンコードは UTF-8
- MIMEタイプ: application/json
- Delphi/C++Builderは、TJSONxxxx系クラスで処理
- XML を JSON に変換するには?
  - 例: XML-JSON相互変換ツール - Bluehawk's lab.
    - http://bluehawk.infinitybird.com/dev/xmljson.html

XML

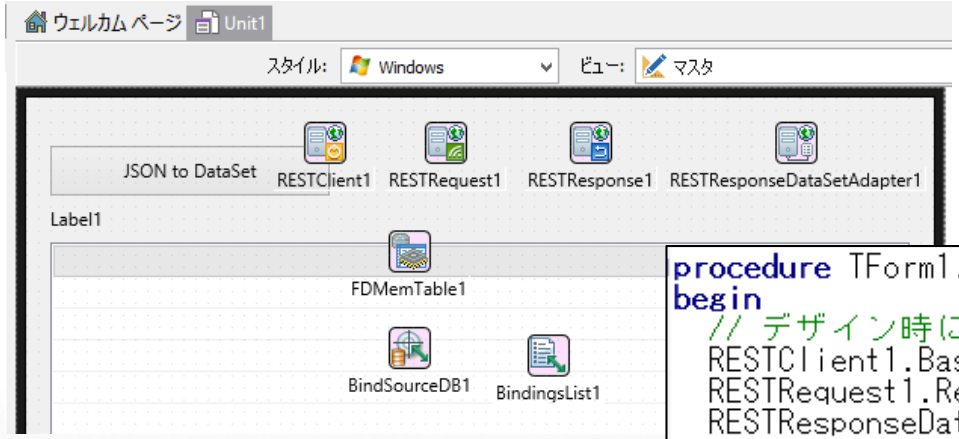
```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="nisiyamazoo.xsd" generated="2012-09-02T12:03:57">
<nisiyamazoo>
<No>1</No>
<classification>乳類</classification>
<name>レッサーパンダ</name>
<latitude>35.950946</latitude>
<longlatitude>136.180778</longlatitude>
<number>9頭</number>
<meal1>9：00ごろ 竹の葉</meal1>
<meal2>11：00ごろ リンゴ・バナナ・にんじんなど</meal2>
<meal3>16：30ごろ くだものおかゆ(ペースト状にすりつぶしたもの)</meal3>
<url1>http://www.city.sabae.fukui.jp/users/zoo/animals/animal_01.html</url1>
</nisiyamazoo>
<nisiyamazoo>
<No>2</No>
<classification>乳類</classification>
```

JSON

```
{
  "dataroot": {
    "-generated": "2012-09-02T12:03:57",
    "-xsi:noNamespaceSchemaLocation": "nisiyamazoo.xsd",
    "-xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
    "-xmlns:od": "urn:schemas-microsoft-com:officedata",
    "nisiyamazoo": [
      {
        "No": "1",
        "classification": "乳類",
        "name": "レッサーパンダ",
        "latitude": "35.950946",
        "longlatitude": "136.180778",
        "number": "9頭",
        "meal1": "9：00ごろ 竹の葉",
        "meal2": "11：00ごろ リンゴ・バナナ・にんじんなど",
        "meal3": "16：30ごろ くだものおかゆ(ペースト状にすりつぶしたもの)",
        "url1": "http://www.city.sabae.fukui.jp/users/zoo/animals/animal_01.html"
      },
      {
        "No": "2",
        "classification": "乳類",
        "name": "フランソワルトン",
```

JSONを受信して処理する

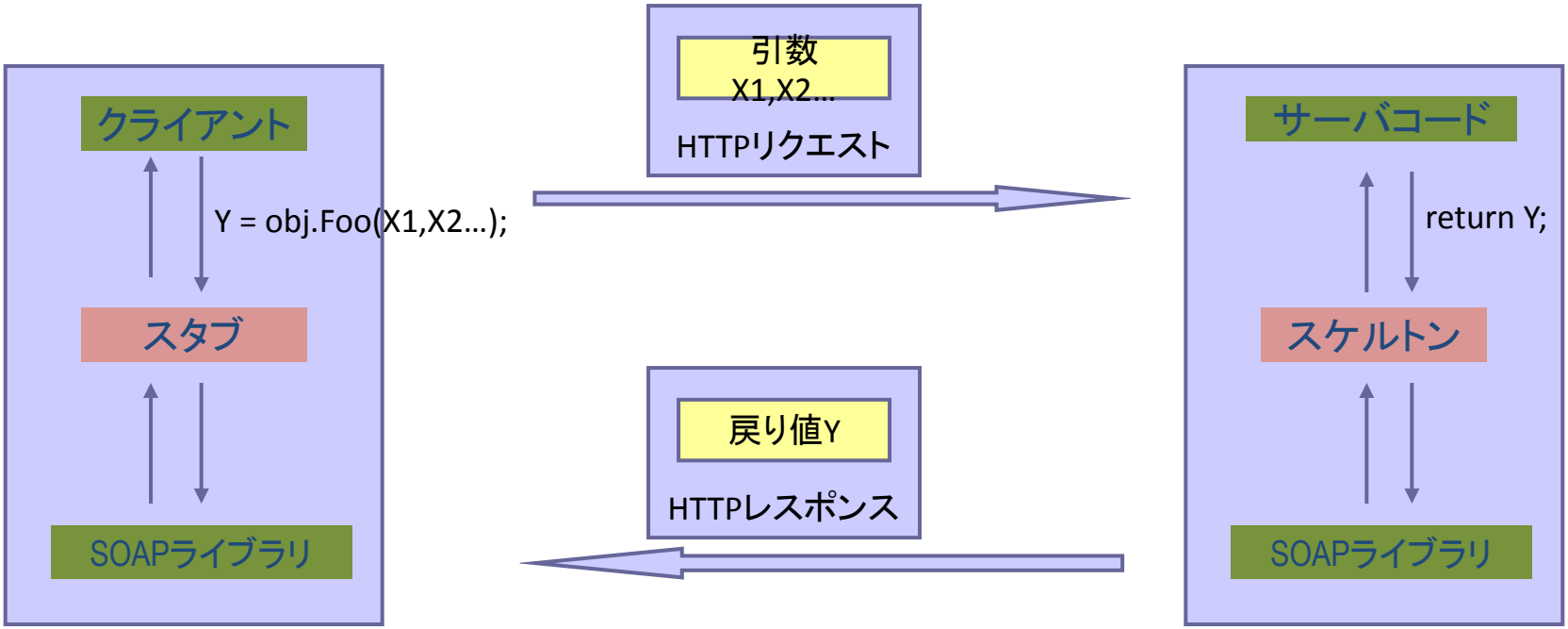
- TRESTxxxx系コンポーネントがとっても便利
  - TRESTClient
  - TRESTRequest
  - TRESTResponse
  - TRESTResponseDataSetAdapter
- 問題は、JSONのデータ構造を示すスキーマが存在しないこと
  - 安全で便利な JSON <—> クラス マッピングが難しい
  - JSONをTDataSet型(例:TFDMemTable)にマッピングすると便利
    - TRESTResponseDataSetAdapterのResponseJSONプロパティを利用




```
procedure TForm1.Button1Click(Sender: TObject);
begin
  // デザイン時にプロパティ設定してもOK
  RESTClient1.BaseURL := 'http://server.hostname/';
  RESTRequest1.Resource := 'nisiyamazoo.json';
  RESTResponseDataSetAdapter1.RootElement := 'dataroot.nisiyamazoo';

  // REST(HTTP)でアクセス
  RESTRequest1.Execute;
end;
```

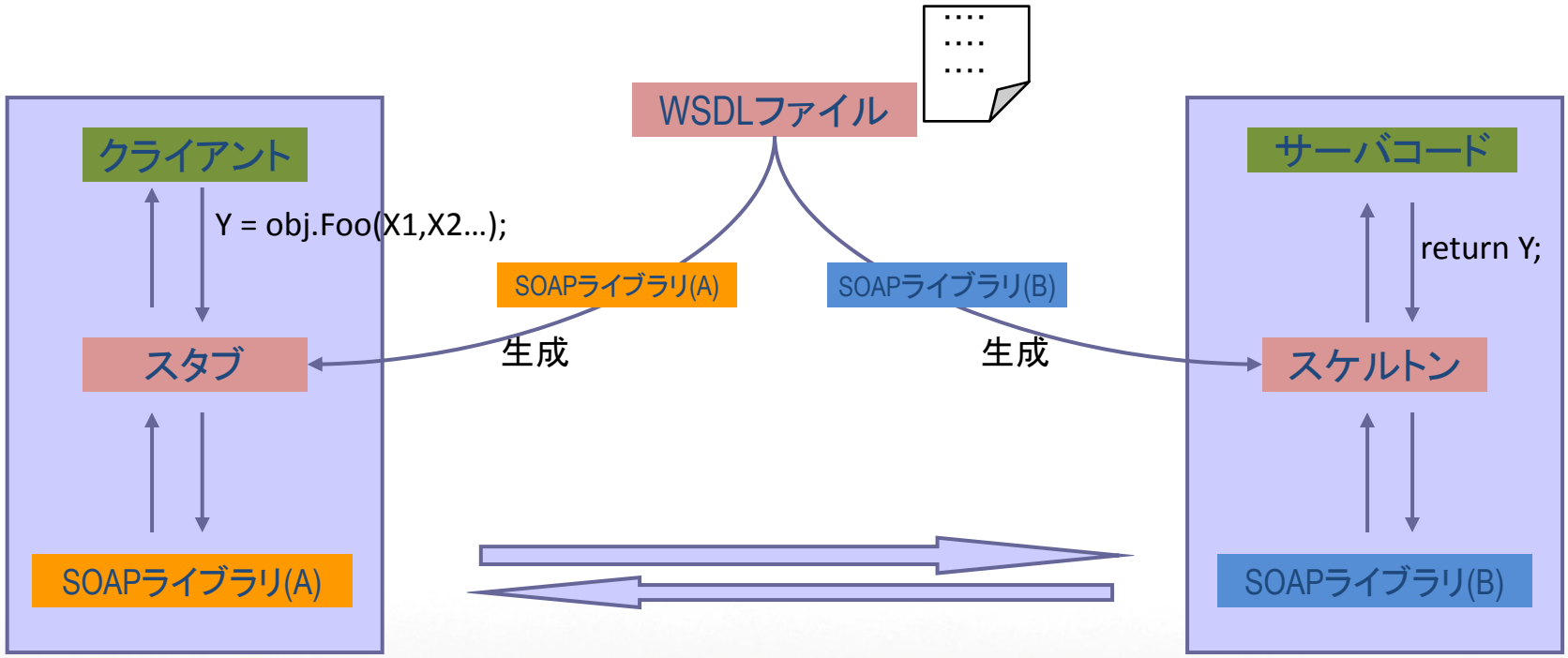
SOAP > HTTP+XML

- XML形式のデータをHTTP上で送受信する
- 利用者は、メソッド呼び出しの形式で通信できる



WSDL

- Webサービス記述言語（Web Services Description Language）
- 提供するWebサービスのインターフェース（メソッド等）を規定する
- WSDLファイルから、クライアント/サーバ各用のスタブ/スケルトンコードを生成する



SOAPサーバー

- XE7では、Win32/Win64のサーバーのみ
  - スタンドアローン(exe), GGI(exe), ISAPI(dll), Apache DSO(dll)
- WSDLはサーバーが生成してくれる
- サーバー用インターフェースを宣言し、実装クラスを定義する
- クラス型を引数・戻り値に使用する場合はTRemotableの派生クラスを定義

```
TMyEmployee = class(TRemotable)
private
  FLastName: UnicodeString;
  FFirstName: UnicodeString;
  FSalary: Double;
published
  property LastName: UnicodeString read FLastName write FLastName;
  property FirstName: UnicodeString read FFirstName write FFirstName;
  property Salary: Double read FSalary write FSalary;
end;

{ 呼び出し可能なインターフェイスは IInvokable から派生していなければなりません }
IMyTest = interface(IInvokable)
['{41B268E4-E680-45B1-A761-E76EF865B17B}']
  { 呼び出し可能なインターフェイスのメソッドではデフォルトを使用できません }
  { 呼び出し規約。stdcall をお勧めします }
  function echoEnum(const Value: TEnumTest): TEnumTest; stdcall;
  function echoDoubleArray(const Value: TDoubleArray): TDoubleArray; stdcall;
  function echoMyEmployee(const Value: TMyEmployee): TMyEmployee; stdcall;
  function echoDouble(const Value: Double): Double; stdcall;
end;
```

```
{ TMyTest の呼び出し可能な実装ファイル }

istry, System.Types, Soap.XSBuiltIns, MyTestIntf;

type
  { TMyTest }
  TMyTest = class(TInvokableClass, IMyTest)
  public
    function echoEnum(const Value: TEnumTest): TEnumTest; stdcall;
    function echoDoubleArray(const Value: TDoubleArray): TDoubleArray; stdcall;
    function echoMyEmployee(const Value: TMyEmployee): TMyEmployee; stdcall;
    function echoDouble(const Value: Double): Double; stdcall;
    constructor Create; override;
    destructor Destroy; override;
  end;
```

SOAPクライアント

- VCL(Win32/Win64)およびFireMonkey(Windows/OS X/iOS/Android)
  - スタブは「WSDLインポータ」を使用して、.pasまたは.cpp/.hを自動生成
  - DOMパーサは MSXML(Windows) または Omni XML(クロスプラットフォーム)
    - Xml.xmldomユニットのDefaultDOMVendorグローバル変数を指定

```
program Project2;

uses
  Vcl.Forms,
  Xml.xmldom,
  Xml.omnixmldom,
  Unit2 in 'Unit2.pas' {Form2},
  IMyTest1 in 'IMyTest1.pas';

{$R *.res}

begin
  DefaultDOMVendor := 'Omni XML';
  Application.Initialize;
  Application.MainFormOnTaskbar := True;
```

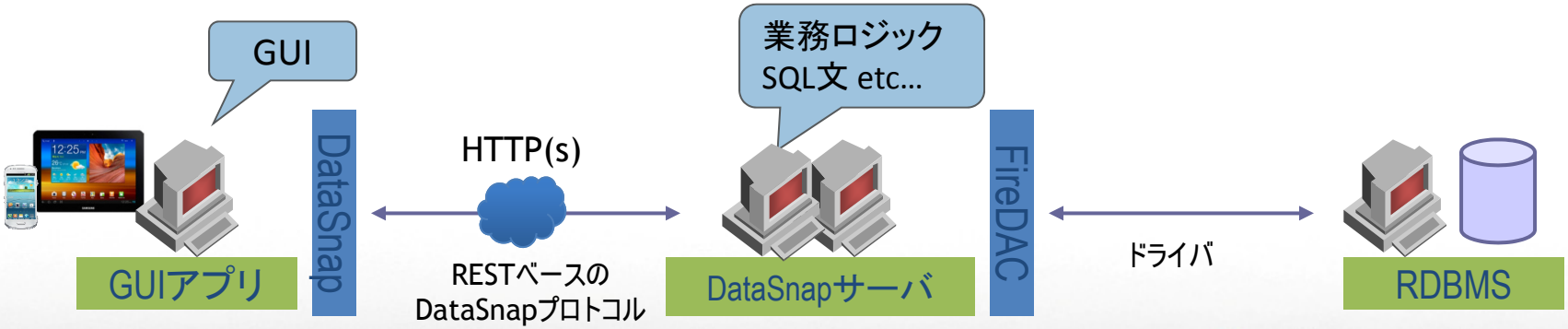
```
#include <Xml.xmldom.hpp>
#include <Xml.omnixmldom.hpp>

//---------------------------------------------
USEFORM("Unit3.cpp", Form3);
//---------------------------------------------
extern "C" int FMXmain()
{
  try
  {
     DefaultDOMVendor = u"Omni XML";
     Application->Initialize();
     Application->CreateForm(__classid(TForm3),
```

```
uses IMyTest1;

procedure TForm2.Button1Click(Sender: TObject);
var
  ws: IMyTest;
  emp: TMyEmployee;
begin
  ws := GetIMyTest;
  emp := ws.echoMyEmployee(nil);

  ListBox1.Items.Add(emp.LastName);
  ListBox1.Items.Add(emp.FirstName);
  ListBox1.Items.Add(FloatToStr(emp.Salary));

  FreeAndNil(emp);
end;
```

```
#if defined(TARGET_OS_IPHONE) || defined(__ANDROID__) || defined(_WIN64)
  #include <memory>
  using namespace std;
#else // WIN32, MacOSX
  #include <memory>
  #include <boost/tr1/memory.hpp>
  using namespace std;
  using namespace std::tr1;
#endif

void __fastcall TForm3::Button1Click(TObject *Sender)
{
  _di_IMyTest ws = GetIMyTest();
  unique_ptr<TMyEmployee> emp(ws->echoMyEmployee(NULL));
  //shared_ptr<TMyEmployee> emp(ws->echoMyEmployee(NULL));

  ListBox1->Items->Add(emp->LastName);
  ListBox1->Items->Add(emp->FirstName);
  ListBox1->Items->Add(FloatToStr(emp->Salary));
```

DataSnap > HTTP+JSON

- JSON形式のデータをHTTP上で送受信する
- 利用者は、メソッド呼び出しの形式で通信できる
  - RPC形式のサーバーメソッドの実装
    - 要は、RDBMSのストアドプロシージャと同じ扱い!!
    - パラメータ & 戻り値
      - in, var, out, return
    - インスタンスのライフサイクル
      - Server, Session, Invocation
    - Callback, Filter, Event
  - 組み込み型・TJSONxxxx型・TDataSet型 と JSON の間は自動変換



TDataSetの配列を送受信する

- なぜTDataSetの配列が必要なのか?
  - 複数テーブルの処理を単一トランザクション(1回のサーバーメソッド呼び出し)で行うため
- 「array of TDataSet」はパラメータに使用できない仕様
  - 基本的にTJSONArrayを使用し、データセットをJSON文字列にしたものを複数個入れる
  - もしくは、Data.FireDACJSONReflectユニットのTFDJSONDataSets型を利用する
    - FireDACのTFDDataSetの派生クラスのインスタンスを複数保持する

```
type
{$METHODINFO ON}
  TServerMethods1 = class(TDataModule)
    FDQueryDepartmentEmployees: TFDQuery;
    FDQueryDepartment: TFDQuery;
    FDQueryDepartmentNames: TFDQuery;
    FDGUIxWaitCursor1: TFDGUIxWaitCursor;
    FDPhysIBDriverLink1: TFDPhysIBDriverLink;
    FDConnectionEMPLOYEE: TFDConnection;
    FDStanStorageJSONLink1: TFDStanStorageJSONLink;
  private
  public
    function EchoString(Value: string): string;
    function ReverseString(Value: string): string;
    // Strongly typed methods
    function GetDepartmentNames: TFDJSONDataSets;
    function GetDepartmentEmployees(const AID: string): TFDJSONDataSets;
    procedure ApplyChangesDepartmentEmployees(const ADeltaList: TFDJSONDeltas);
    // Equivalent TJSONObject methods (C++ compatible)
    function GetDepartmentNamesJSON: TJSONObject;
    function GetDepartmentEmployeesJSON(const AID: string): TJSONObject;
    procedure ApplyChangesDepartmentEmployeesJSON(const AJSONObject: TJSONObject);
  end;
{$METHODINFO OFF}
```

Developer Camp

www.embarcadero.com/jp