embarcadero Developer Camp

【T9】テクニカルセッション

Delphi/C++BuilderでiOS/Macアプリを作ろう！

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

アジェンダ

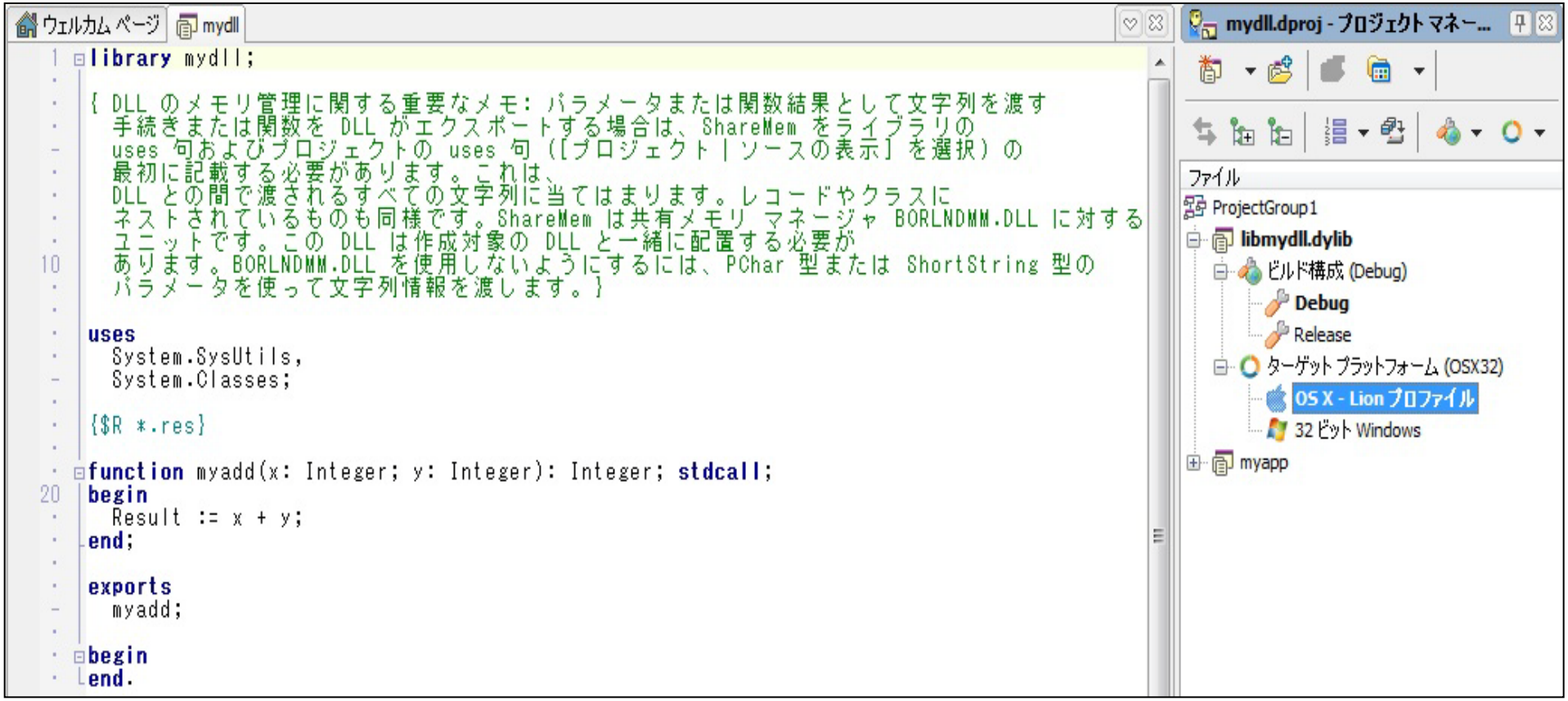
- OS Xで動的ライブラリ(.dylib) デモ
  - Delphiで作成して、Delphiアプリから呼び出し
  - C++Builderで作成して、C++Builderアプリから呼び出し
- SQLite3 を利用する デモ
  - C++BuilderでOS X内蔵のSQLite3を利用する
- OpenCL を利用する デモ
  - C++BuilderでOS X内蔵のOpenCLを利用する
- OS XでSOAPクライアントを作成する デモ
  - Win32/Win64のSOAPサーバーからのTClientDataSetを受信して、FireMonkeyのStringGridに表示
- Update 4 で新たに追加されたFireMonkey向けモバイルコネクタ デモ
  - iOS(4.x, 5.0)から、FreePascal向けプロキシでDataSnapを呼び出す
    - 注意点あり

embarcadero
Developer Camp

OS Xで動的ライブラリ(.dylib)

Delphiで作成

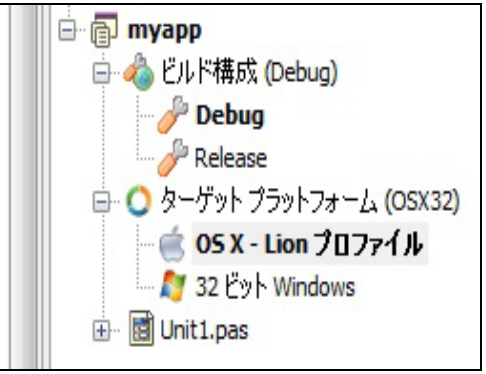
- lib[プロジェクト名].dylib が生成される
  - 共有メモリマネージャ(ShareMemユニット,BORLNDMM.DLL)は無い



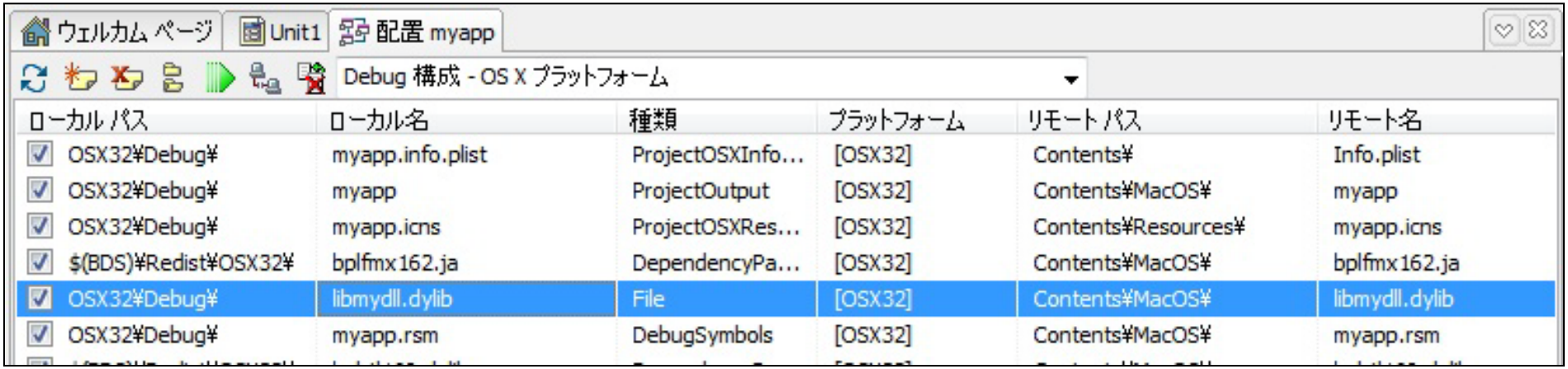
Delphiアプリから呼び出し

- external ‘xxxx.dylib’;

```
implementation

{$R *.fmx}

function myadd(x: Integer; y: Integer): Integer; stdcall; external 'libmydll.dylib';

procedure TForm1.Button1Click(Sender: TObject);
begin
  ShowMessage(IntToStr(myadd(1, 2)));
end;
```



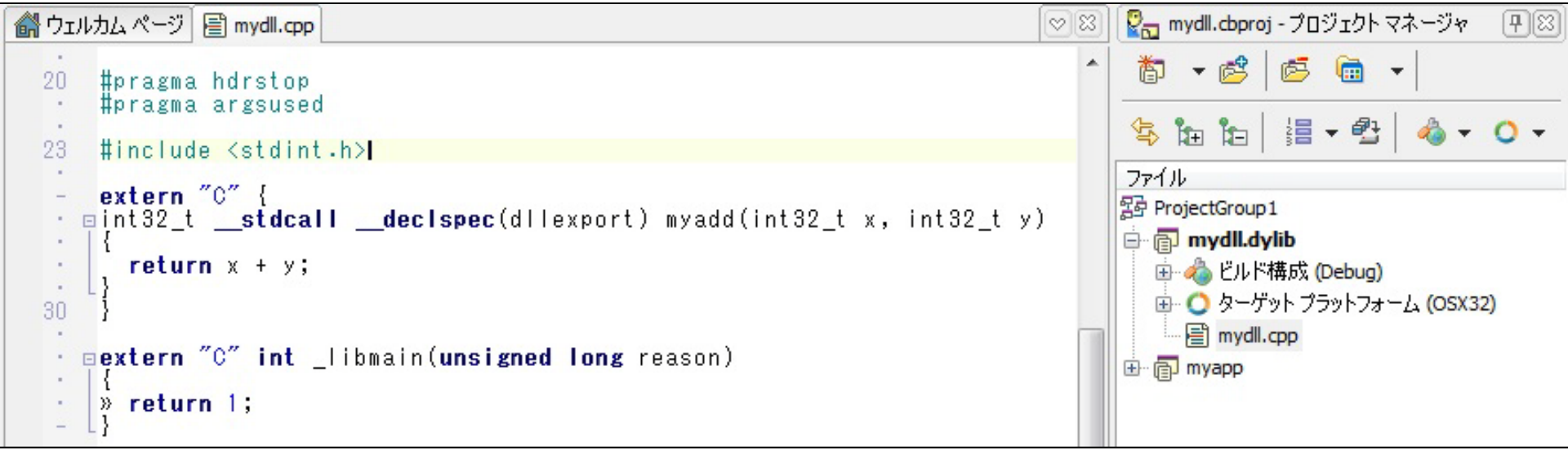
- 配置ウィザードで.dylibファイルを追加



ローカル パス	ローカル名	種類	プラットフォーム	リモート パス	リモート名
OSX32¥Debug¥	myapp.info.plist	ProjectOSXInfo...	[OSX32]	Contents¥	Info.plist
OSX32¥Debug¥	myapp	ProjectOutput	[OSX32]	Contents¥MacOS¥	myapp
OSX32¥Debug¥	myapp.icns	ProjectOSXRes...	[OSX32]	Contents¥Resources¥	myapp.icns
$(BDS)¥Redist¥OSX32¥	bplfmx162.ja	DependencyPa...	[OSX32]	Contents¥MacOS¥	bplfmx162.ja
OSX32¥Debug¥	libmydll.dylib	File	[OSX32]	Contents¥MacOS¥	libmydll.dylib
OSX32¥Debug¥	myapp.rsm	DebugSymbols	[OSX32]	Contents¥MacOS¥	myapp.rsm

C++Builderで作成

- [プロジェクト名].dylib が生成される
  - 共有メモリマネージャ(MEMMGR.LIB,BORLNDMM.DLL)は無い
- int32_t, char16_t などの型(stdint.h)を使いましょう
  - MacOS Xでは、wchar_t は 4バイト(32bit) です!!

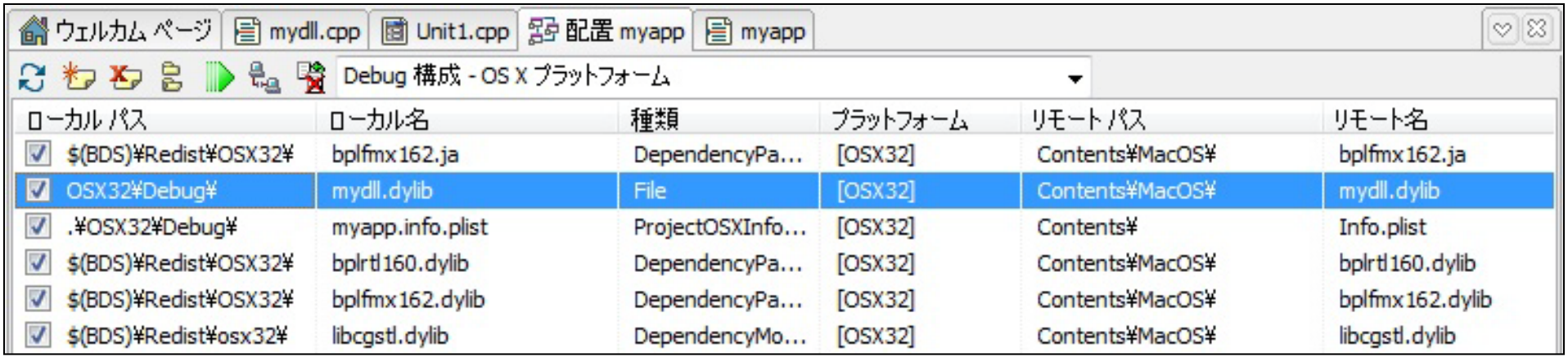


C++Builderアプリから呼び出し

- #pragma link ‘xxxx.dylib’

```
#pragma link "mydll.dylib"
extern "C" {
int32_t __stdcall __declspec(dllimport) myadd(int32_t x, int32_t y);
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  ShowMessage(IntToStr(myadd(1, 2)));
}
```

- 配置ウィザードで.dylibファイルを追加



ローカル パス	ローカル名	種類	プラットフォーム	リモート パス	リモート名
$(BDS)¥Redist¥OSX32¥	bplfmx162.ja	DependencyPa...	[OSX32]	Contents¥MacOS¥	bplfmx162.ja
OSX32¥Debug¥	mydll.dylib	File	[OSX32]	Contents¥MacOS¥	mydll.dylib
.¥OSX32¥Debug¥	myapp.info.plist	ProjectOSXInfo...	[OSX32]	Contents¥	Info.plist
$(BDS)¥Redist¥OSX32¥	bplrtl160.dylib	DependencyPa...	[OSX32]	Contents¥MacOS¥	bplrtl160.dylib
$(BDS)¥Redist¥OSX32¥	bplfmx162.dylib	DependencyPa...	[OSX32]	Contents¥MacOS¥	bplfmx162.dylib
$(BDS)¥Redist¥osx32¥	libcgstl.dylib	DependencyMo...	[OSX32]	Contents¥MacOS¥	libcgstl.dylib

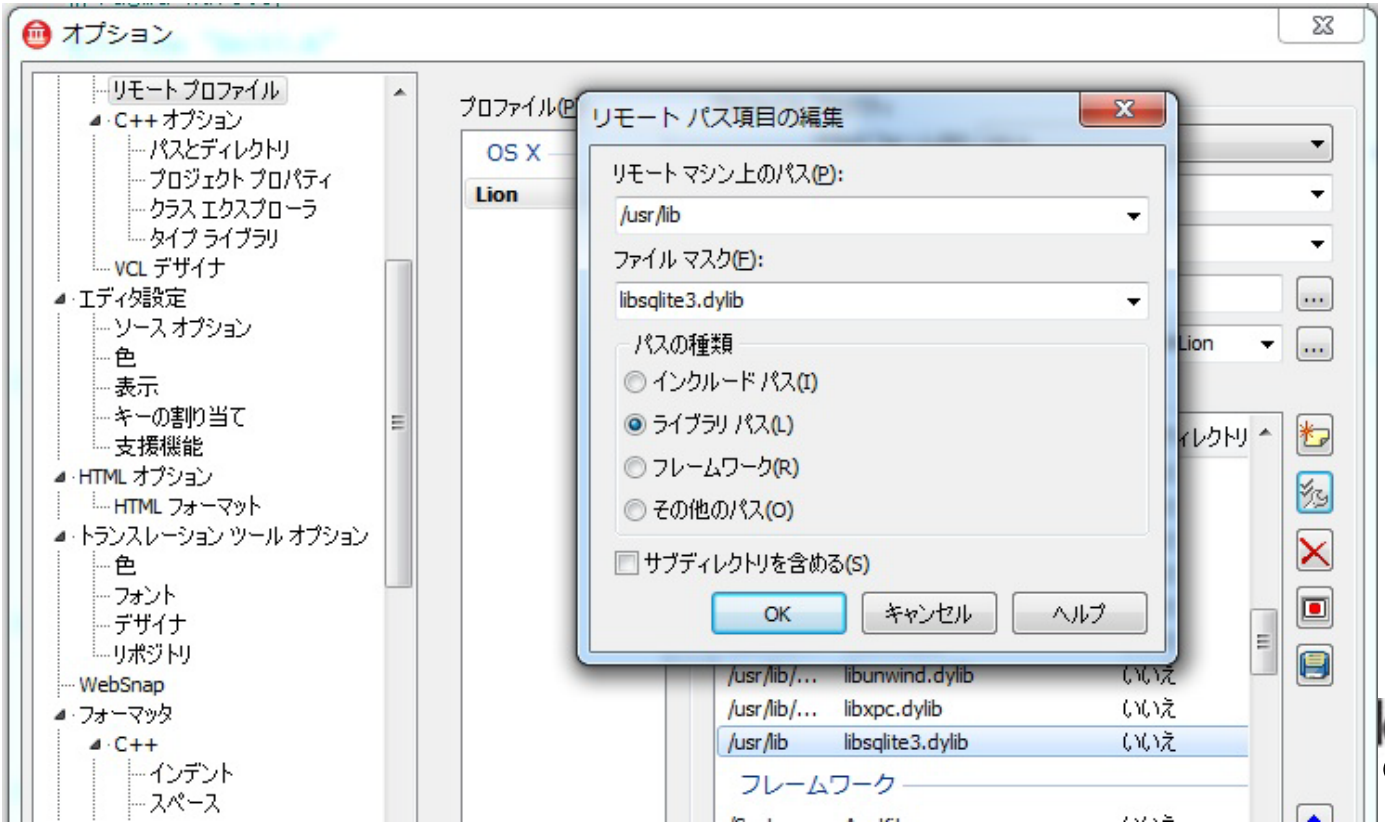
embarcadero
Developer Camp

2

SQLite3 を利用する

/usr/include/sqlite3.h , /usr/lib/libsqlite3.dylib

- [ツール]-[オプション]-[環境オプション]-[リモートプロファイル]-[リモートパス]
  - 必要に応じて、.h と .dylib をPAServerからインポート
    - [ローカルファイルキャッシュの更新]を忘れずに!



データベースファイルの作成またはオープン

- #include <sqlite3.h>
- #pragma link ‘libsqlite3.dylib’
- sqlite3_open: 作成またはオープン
- sqlite3_close: クローズ

```
//---------------------------------------------------------------------------
#include <sqlite3.h>
#pragma link "libsqlite3.dylib"
static sqlite3* db = NULL;
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  db = NULL;
  int ret = sqlite3_open("test.db", &db);
  if( ret != SQLITE_OK )
    ShowMessage(L"エラー:sqlite3_open");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
  if( db != NULL ) {
    sqlite3_close(db);
    db = NULL;
  }
}
```

テーブルの作成 および 行のINSERT

- C++BuilderのUnicodeString型(1文字16ビット)を使用
  - 一部、UTF8String型(UTF8Encode関数)を使用
- テーブルの存在確認
  - システムテーブルにSELECT文を実行
    - sqlite3_get_table / sqlite3_free_table
- テーブルの作成
  - sqlite3_exec で CREATE TABLE文を実行
- 行のINSERT
  - パラメータ付きクエリ (例: …=:param …)
  - char16_t版の関数を利用
    - sqlite3_prepare16 / sqlite3_reset: SQL文の準備
    - sqlite3_bind_text16: 文字列パラメータのセット
    - sqlite3_step: SQL文の実行
    - sqlite3_finalize: SQL文の後始末

テーブルの作成 および 行のINSERT (続き)

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  // テーブルの有無をチェック
  UnicodeString sql = L"SELECT name FROM sqlite_master WHERE type='table'"¥
                      " AND name='CUSTOMER'";
  char** pazResult;
  int pnRow, pnCol;
  char* errmsg;
  sqlite3_get_table(db, &(UTF8Encode(sql)[1]), &pazResult, &pnRow, &pnCol, &errmsg);
  sqlite3_free_table(pazResult);
  if( pnRow > 0 )
    return;

  // テーブルを作成
  sql = L"CREATE TABLE CUSTOMER"¥
        "(ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT, ADDRESS TEXT)";
  int ret = sqlite3_exec(db, &(UTF8Encode(sql)[1]), NULL, NULL, &errmsg);

  // 行を追加
  sql = L"INSERT INTO CUSTOMER(NAME, ADDRESS) VALUES(:name, :address)";
  UnicodeString p1 = L"エンバカデロ";
  UnicodeString p2 = L"東京都";
  sqlite3_stmt* stmt;
  const void *pzTail;
  sqlite3_prepare16(db, sql.c_str(), -1, &stmt, &pzTail);
  sqlite3_reset(stmt);
  sqlite3_bind_text16(stmt, 1, p1.c_str(), -1, SQLITE_STATIC);
  sqlite3_bind_text16(stmt, 2, p2.c_str(), -1, SQLITE_STATIC);
  ret = sqlite3_step(stmt);
  sqlite3_finalize(stmt);
}
```

SELECT文で行を検索

- sqlite3_step とその戻り値で結果セットをイテレート
  - SQLITE_ROW: 行がある
  - SQLITE_DONE: 行が無くなった
- sqlite3_column_text16 でカレント行の文字列を取得
  - 戻り値の型を char16_t* 型にするのを忘れずに!!
  - C++BuilderのUnicodeString型に変換する

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  UnicodeString sql = L"SELECT ADDRESS FROM CUSTOMER WHERE NAME=:name";
  UnicodeString p1 = L"エンバカデロ";
  sqlite3_stmt* stmt;
  const void *pzTail;
  sqlite3_prepare16(db, sql.c_str(), -1, &stmt, &pzTail);
  sqlite3_reset(stmt);
  sqlite3_bind_text16(stmt, 1, p1.c_str(), -1, SQLITE_STATIC);
  int ret = sqlite3_step(stmt);
  while( ret == SQLITE_ROW ) {
    const char16_t* adr = (char16_t*)sqlite3_column_text16(stmt, 0);
    ShowMessage(UnicodeString(adr));
    ret = sqlite3_step(stmt);
  }
  sqlite3_finalize(stmt);
}
```

embarcadero
Developer Camp

3

OpenCL を利用する

KHRONOS GROUP

OpenCL
The Open Standard for Heterogeneous Parallel Programming

OpenCL on Snow Loepard, Lion

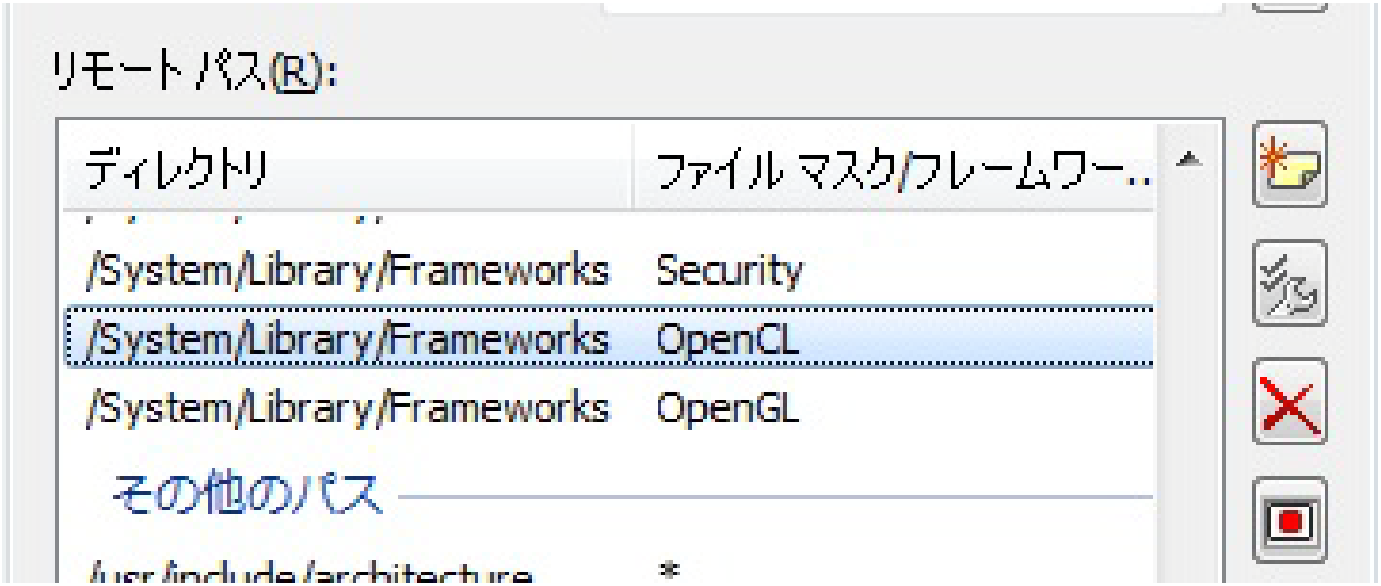
- OpenCL とは?
  - 出典: wikipedia

> **OpenCL**（オープンシーエル、Open Computing Language）は、OpenCL C言語による、マルチコアCPUやGPU、Cellプロセッサ、DSPなどによる異種混在の計算資源（ヘテロジニアス環境、Heterogeneous）を利用した並列コンピューティングのためのクロスプラットフォームなフレームワークである。用途には高性能計算サーバやパーソナルコンピュータのシステムのほか、携帯機器などでの利用も想定されており、組み込みシステム向けに必要条件を下げたOpenCL Embedded Profileが存在する。
>
> OpenCLの仕様はアップル社によって提案されたのち、標準化団体クロノス・グループの作業部会OpenCL Working Group（旧Compute Working Group）によって策定されている。仕様はロイヤリティフリーなオープン標準として公開されており、仕様に基づいたフレームワークの実装はサードパーティによって行われる。ただし、実装されたフレームワークに対して許諾される商標ライセンスに必要な仕様一致性テストには、'nominal fee'（名目上の手数料）が必要である[1]。
>
> OpenCL 1.1は2010年6月11日に発表された。2012年2月現在の最新の正式版は2011年11月15日に発表されたOpenCL 1.2である。

- C++Builder XE2 がサポートしている OS X 10.6.x, 10.7.x には、OpenCL が標準搭載

OpenCL,OpenGL - /System/Library/Frameworks

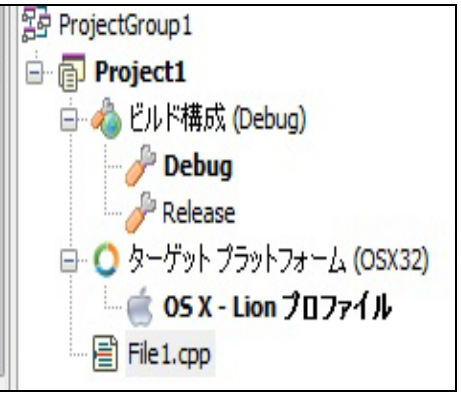
- [ツール]-[オプション]-[環境オプション]-[リモートプロファイル]-[リモートパス]
  - フレームワークとして OpenCL と OpenGL をインポート
    - OpenGL も忘れずに!!
    - 必要なヘッダとライブラリが利用できるようになります



GPUに送り込むカーネルコードを定義

- コンパイル前のカーネルコードを文字列で定義
  - C言語的なコードを書きます
    - char配列でもファイルでもOK
  - float値を2乗して、結果を入れて返す: y[i] = x[i] * x[i];

```
const char *KernelSource = "¥n" ¥
"__kernel void square(                                                  ¥n" ¥
"  __global float* input, __global float* output, const unsigned int count) {¥n" ¥
"    int i = get_global_id(0);                                          ¥n" ¥
"    if( i < count )                                                    ¥n" ¥
"      output[i] = input[i] * input[i];                                 ¥n" ¥
"}¥n";
```



計算用の初期値, GPUに接続, コードのコンパイル

- アプリ(ホスト)側で計算用の初期値(配列)を準備
  - この配列の値をGPU(デバイス)にコピーして渡します

```
#define DATA_SIZE (16)

int _tmain(int argc, _TCHAR* argv[])
{
  float data[DATA_SIZE];
  unsigned int count = DATA_SIZE;
  for(int i = 0; i < count; i++)
    data[i] = (float)(i+1);
```

- 1個のGPU(デバイス)に接続して、コードをコンパイル
  - OpenCLでは、GPUではなくCPUを利用することも可

```
#include <OpenCL/opencl.h>

int _tmain(int argc, _TCHAR* argv[])
{
  cl_device_id device_id;

  cl_int err = clGetDeviceIDs(NULL, CL_DEVICE_TYPE_GPU, 1, &device_id, NULL);
  cl_context ctx = clCreateContext(NULL, 1, &device_id, NULL, NULL, &err);
  cl_program pg = clCreateProgramWithSource(ctx, 1, (const char **)&KernelSource, NULL, &err);
  err = clBuildProgram(pg, 0, NULL, NULL, NULL, NULL);
  cl_kernel kernel = clCreateKernel(pg, "square", &err);
```

GPUで使用するパラメータの作成

- パラメータ用の配列をGPU内に作成して、そこにアプリ(ホスト)側の配列をコピーする
- 計算結果を格納する配列をGPU内に作成
- 計算用の配列のサイズも渡しておく

```
cl_mem input = clCreateBuffer(ctx, CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
                              sizeof(float)*count, data, &err);
cl_mem output = clCreateBuffer(ctx, CL_MEM_WRITE_ONLY,
                               sizeof(float)*count, NULL, &err);
err = clSetKernelArg(kernel, 0, sizeof(cl_mem), &input);
err = clSetKernelArg(kernel, 1, sizeof(cl_mem), &output);
err = clSetKernelArg(kernel, 2, sizeof(unsigned int), &count);
```

GPUで計算実行，結果を取得して確認，後始末

- GPU内の複数の計算ユニットで並列実行
  - clFinish関数で実行終了を待ち合わせ
    - clEnqueueNDRangeKernel関数は非同期で実行される
- 計算結果の配列をGPUからホストにコピー
  - clEnqueueReadBuffer関数で取得（読み込みが終わるまで待たされる）
  - 試しに、CPUの計算結果とGPUの計算結果を比較する

```
cl_command_queue cq = clCreateCommandQueue(ctx, device_id, 0, &err);

size_t global_work_size = count;
err = clEnqueueNDRangeKernel(cq, kernel, 1, NULL,
                             &global_work_size, NULL, 0, NULL, NULL);
clFinish(cq);

float results[DATA_SIZE];
err = clEnqueueReadBuffer(cq, output, CL_TRUE, 0, sizeof(float)*count, results, 0, NULL, NULL);
for(int i = 0; i < count; i++) {
  if( results[i] == data[i] * data[i] )
    printf("%f¥n", results[i]);
}
```

- 作成したリソース群を解放

```
  clReleaseMemObject(input);
  clReleaseMemObject(output);
  clReleaseCommandQueue(cq);
  clReleaseKernel(kernel);
  clReleaseProgram(pg);
  clReleaseContext(ctx);

  return 0;
}
```

embarcadero
Developer Camp

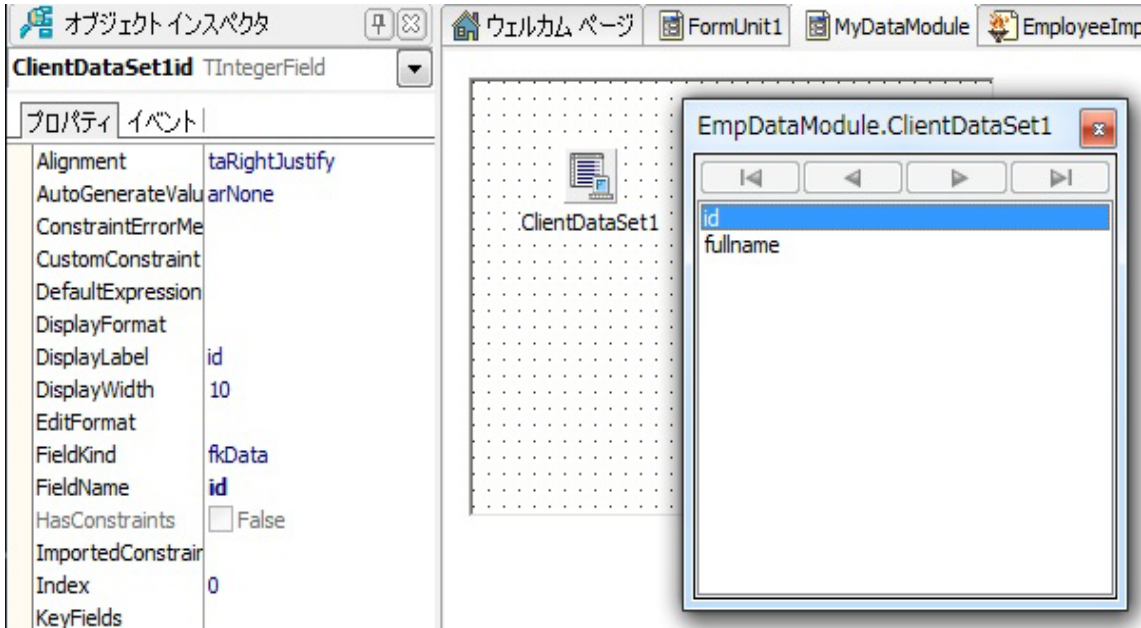
OS XでSOAPクライアントを作成する

作成するサンプルシステムの目標・設計

- SOAPサーバー
  - Win32(またはWin64)のDelphi(またはC++Builder)で作成
  - スタンドアロン(.exe)サーバー
  - ポート番号 8080
  - TClientDataSetを返すメソッドをクライアントに公開
    - ただし、TClientDataSetそのものではなく、XML化した文字列(string)を採用
- SOAPクライアント
  - MacOS X向けFireMonkeyアプリケーション
    - Delphi または C++Builder で作成
  - WSDLからSOAPクライアント用プロキシを生成
    - Windows版およびMacOS X版で共通!!
  - サーバーから取得したTClientDataSetをTStringGridに表示

SOAPサーバー – データモジュールを用意

- [ファイル]-[新規作成]-[その他]-[Delphiプロジェクト]-[Webサービス]-[SOAPサーバーアプリケーション]
  - サンプルのSOPAサーバーインターフェースを作成
    - サービス名は Employee
- [ファイル]-[新規作成]-[その他]-[Delphiプロジェクト]-[Delphiファイル]-[データモジュール]
  - TClientDataSet を配置
  - [項目の設定]でフィールドを追加
    - TIntegerField
      - FieldName は id
    - TWideStringField
      - FieldName は fullname



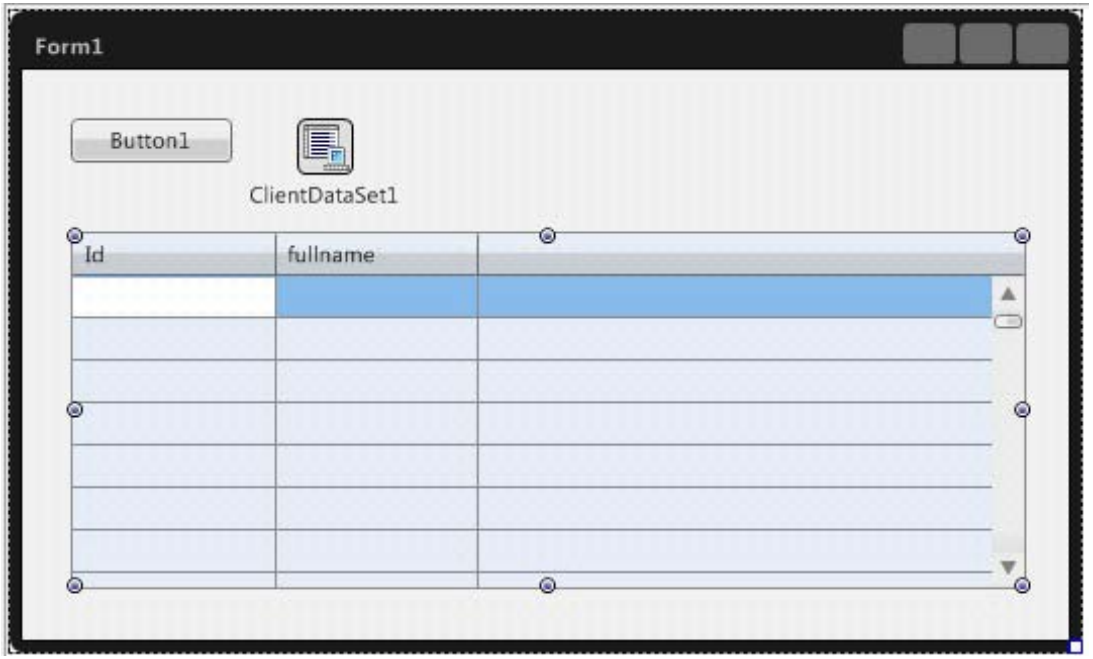
クライアントに公開するメソッド

- function getEmployeeDataSetXML: string; stdcall;
  - interface と 実装class を編集
    - [サービス名]Intf.pas
    - [サービス名]Impl.pas
- TClientDataSetのXMLDataプロパティでXML表現を取得

```
type
  TEmployee = class(TInvokableClass, IEmployee)
  public
    function getEmployeeDataSetXML(): string; stdcall;
    function echoMyEmployee(const id: Integer): TMyEmployee; stdcall;
  end;

implementation

uses MyDataModule;

function TEmployee.getEmployeeDataSetXML: string;
var
  dm: TEmpDataModule;
begin
  Result := '';
  dm := TEmpDataModule.Create(nil);
  try
    dm.ClientDataSet1.CreateDataSet;
    dm.ClientDataSet1.Insert;
    dm.ClientDataSet1id.AsInteger := 1;
    dm.ClientDataSet1fullname.AsString := '山田太郎';
    dm.ClientDataSet1.Post;
    dm.ClientDataSet1.Insert;
    dm.ClientDataSet1id.AsInteger := 2;
    dm.ClientDataSet1fullname.AsString := '山田花子';
    dm.ClientDataSet1.Post;
    Result := dm.ClientDataSet1.XMLData;
  finally
    dm.Free;
  end;
end;
```

SOAPクライアント – WSDLをインポート

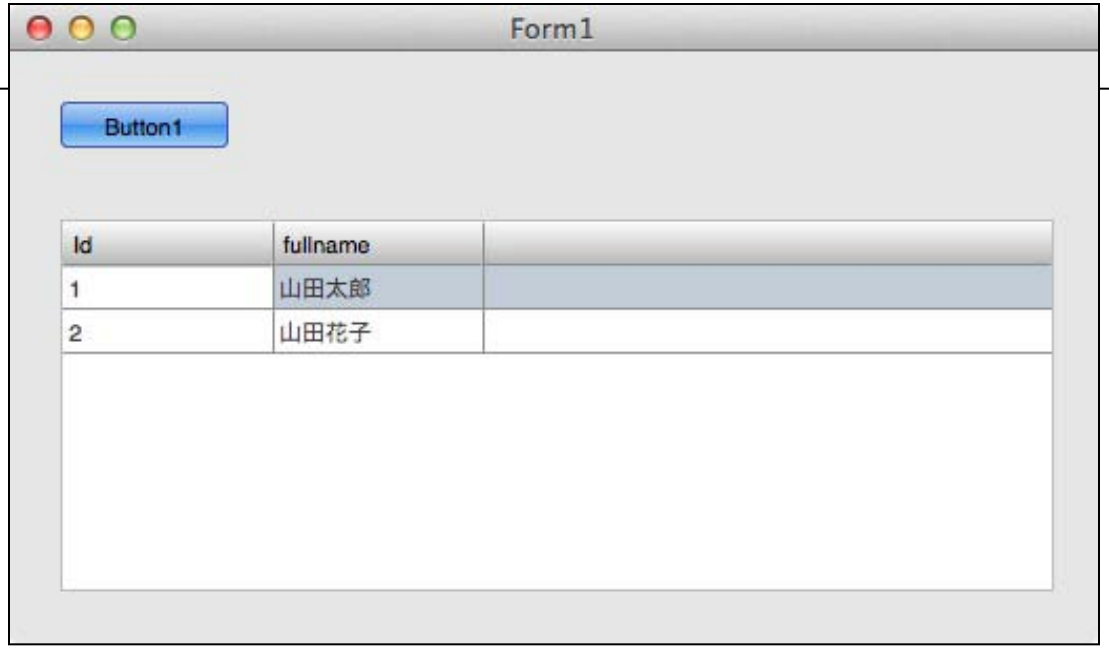
- [ファイル]-[新規作成]-[その他]-[Delphiプロジェクト]-[Webサービス]-[WSDLインポータ]
  - FireMonkeyフォームからクライアントプロキシユニットを参照
- FireMonkeyフォーム上に、以下を配置
  - TButton
  - TClientDataSet
  - TStringGrid
    - [項目エディタ]で項目(TStringColumn)の追加
      - Id
      - fullname



SOAPクライアント – TClientDataSetを復元

- TClientDataSetのXMLDataプロパティにXML表現をセットするだけ

```
procedure TForm1.Button1Click(Sender: TObject);
var
  xml: string;
begin
  xml := proxy.getEmployeeDataSetXML();
  ClientDataSet1.XMLData := xml;
  StringGrid1.RowCount := ClientDataSet1.RecordCount;
  while not ClientDataSet1.Eof do
  begin
    StringGrid1.Cells[0, ClientDataSet1.RecNo-1] :=
                          ClientDataSet1.FieldByName('id').AsString;
    StringGrid1.Cells[1, ClientDataSet1.RecNo-1] :=
                          ClientDataSet1.FieldByName('fullname').AsString;

    ClientDataSet1.Next;
  end;
end;
```



embarcadero
Developer Camp

Update 4 で新たに追加された iOS/FireMonkey向け モバイルコネクタ

Update 4 をインストールすると…

- iOS FireMonkey(FreePascal)向けのモバイルコネクタがリポジトリフォルダに追加される
  - …¥RAD Studio¥9.0¥ObjRepos¥ja[en]¥dsrest¥connectors¥
    - freepascal_ios42 フォルダ: iOS 4.2以降
    - freepascal_ios50 フォルダ: iOS 5.0向け
- DataSnap RESTサーバーのプロジェクト
  - DelphiのWebModule用ユニットのuses(例: WebModuleUnit1.pas)
    - Datasnap.DSProxyFreePascal_iOS
  - C++BuilderのWebModule用ユニットの.h(例: WebModuleUnit1.h)
    - #include <Datasnap.DSProxyFreepascal_iOS.hpp>
  - プロジェクトのproxyフォルダにモバイルコネクタ(必要があればコピー)
    - サーバーを起動してWebブラウザまたは専用ツールを起動
      - http://server[:port]/proxy/freepascal_ios42.zip
      - http://server[:port]/proxy/freepascal_ios50.zip
      - DSProxy.pasが自動生成される

モバイルコネクタをクライアントで利用する

- モバイルコネクタ用の.pasファイル 15個 をプロジェクトに追加

```
uses DSProxy, DSRESTConnection, DSRESTTypes;

procedure TForm1.Button1Click(Sender: TObject);
var
  conn: TDSRESTConnection; // 接続情報だけ
  proxy: TServerMethods1;  // プロキシ本体
  p: WideString;
  ret: string;
begin
  conn := TDSRESTConnection.Create(nil);
  conn.Host := '172.16.89.128';
  conn.Port := 8080;
  conn.ConnectionTimeout := 60; // 60秒
  try
    proxy := TServerMethods1.Create(conn);
    try
      p := Edit1.Text;
      ret := proxy.ReverseString(p);
      Edit1.Text := ret;
    except
      on E1: DBXException do
        ShowMessage(E1.Message);
      on E2: Exception do
        ShowMessage(E2.Message);
    end;
  finally
    FreeAndNil(proxy);
  end;
  FreeAndNil(conn);
end;
```

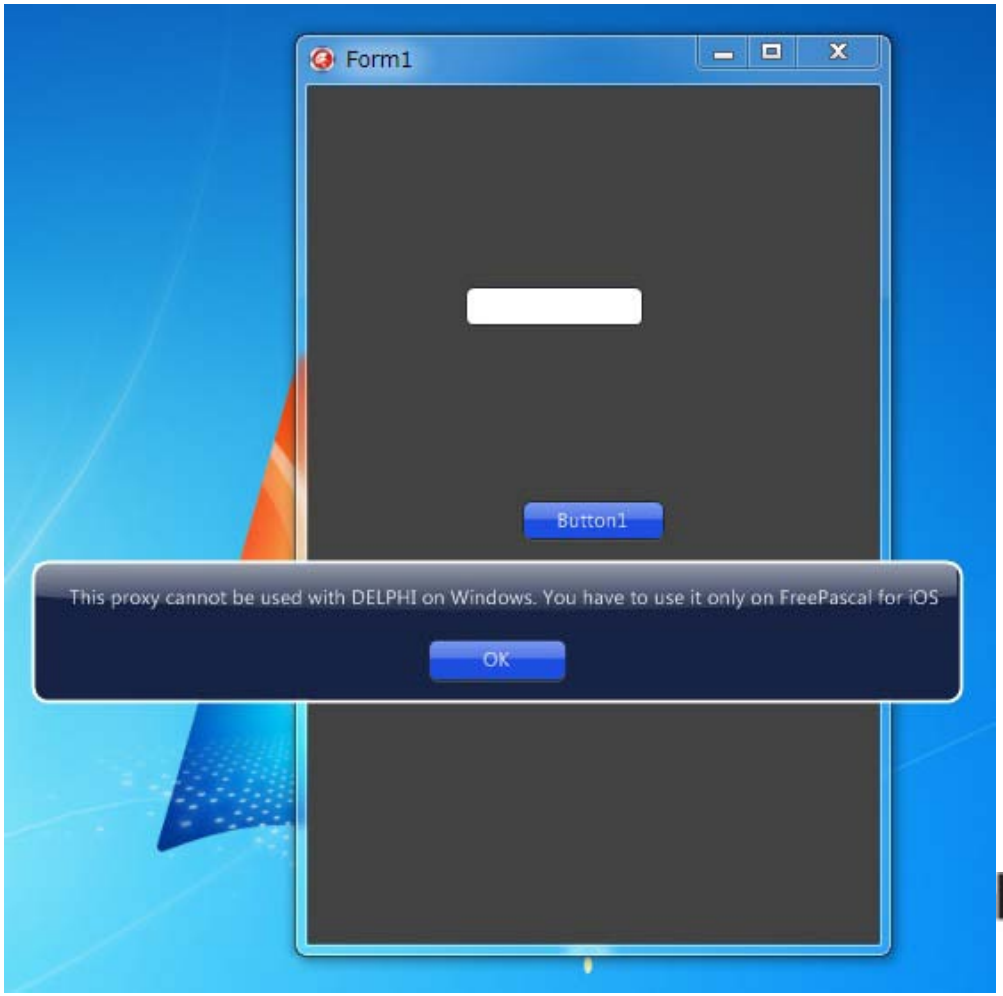
接続先を指定
デフォルトのプロトコルはhttp

DataSnapサーバーへの接続
サーバーメソッドの呼び出し

DataSnap特有の例外
DBXException

Windows上で実行すると…

- モバイルコネクタでサーバーメソッドの呼び出しをテストする
  - Windows上では、メソッドの呼び出し時に例外が発生する仕様
    - XcodeとiPhoneシミュレータでのデバッグが必須



日本語などのUnicode文字列が化けて送信される

- Update 4 に付属する DSRESTConnection.pas の不具合
  - QC#103021
    - http://qc.embarcadero.com/wc/qcmain.aspx?d=103021
- uses句 と EncodeUrlNS2関数 に加筆修正を施してください
  - 予め、リポジトリ内のファイル自体を書き換えておくと便利

```
uses
  DBXDefaultFormatter, DBXJsonTools, DBXValue, FPCStrings
{$IFDEF FPC_ON_IOS} , CFURL, CFString, CFBase{$ENDIF};

{$IFDEF FPC_ON_IOS}

function EncodeUrlNS2(Value: NSString): NSString;
var
  utf8: CFStringRef;
begin
  utf8 := CFStringCreateWithCString(nil, Value.UTF8String, kCFStringEncodingUTF8);
  Result := NSString(CFURLCreateStringByAddingPercentEscapes(nil,
    utf8, nil, CFSTR(PChar('!*();:@&=+$,/?%#[]''')),
    kCFStringEncodingUTF8));
  CFRelease(utf8);
  Result.autorelease;
end;
```

iPhoneシミュレータ上で実行

- 予め、ターミナルからiPhoneシミュレータのプロセスをUTF-8で起動しておく(日本語などの文字化け対策)
  - QC#101418
    - http://qc.embarcadero.com/wc/qcmain.aspx?d=101418

```
$ export LANG=ja_JP.UTF-8
$ cd /Developer/Platforms/iPhoneSimulator.platform/Developer/Applications/iPhone¥
    Simulator.app/Contents/MacOS

$ ./iPhone¥ Simulator
```

- Xcodeでビルドして起動!!

embarcadero
Developer Camp

Q & A