
Developer Camp

【A2】C++テクニカルセッション

# 「C++開発者のための最新プログラミングエッセンス」

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

- DLL/dylibを試す
- C++Builderのパッケージを試す
- モバイル向けコンパイラでNative[U]Intを試す
- vector & shared_ptr を試す
- Delphiの無名スレッド機能をC++で試す
- FireMonkeyのプラットフォームサービスを試す
- C++ CORBAを試す


# DLL/dylibを試す

# 関数のexport & import

- Windowsは .dll, Mac OS Xは .dylib (※Androidは .so)
  - モジュール内のCの関数を外部に公開(C++のクラスも可)
  - 事前にリンクまたは動的にロードして呼び出す

```
#include <fmx.h>

#pragma hdrstop
#pragma argsused

// bcc32: アンダースコアを付けてシンボル名を生成する OFF (-u-)
extern "C" {
__declspec(dllexport) int32_t __cdecl MyAdd(int32_t x, int32_t y)
{
  return x + y;
}
}

extern "C" int _libmain(unsigned long reason)
{
  return 1;
}
```

# 関数のexport & import(続き)

```
extern "C" {
typedef int32_t __cdecl (*MyAdd)(int32_t x, int32_t y);
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
#ifdef _Windows
  UnicodeString libname = "Project1.dll";
  AnsiString funcname = "MyAdd";
#else
  UnicodeString libname = "Project1.dylib";
  UnicodeString funcname = "MyAdd";
#endif
  HMODULE hm = LoadLibrary(libname.w_str());
  if(hm) {
    MyAdd f = (MyAdd)GetProcAddress(hm, funcname.c_str());
    if(f) {
      int32_t a = f(1, 2);
      ShowMessage(IntToStr(a));
    }
    FreeLibrary(hm);
  }
}
```

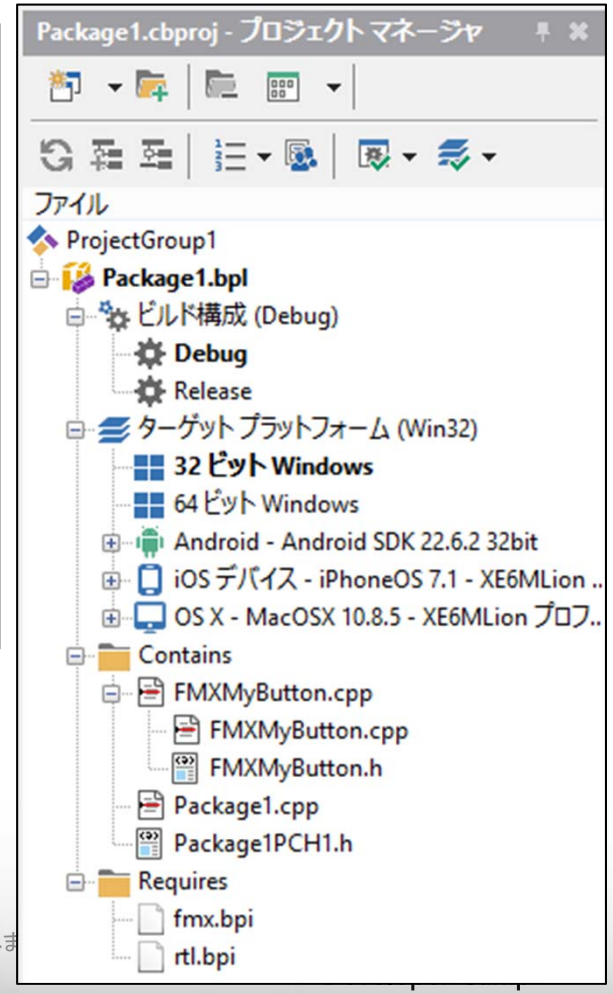

# C++Builderのパッケージを試す

# コンポーネント等をexport & import

- .bplプロジェクトで、C++言語によるカスタムコンポーネントや関数を公開
- C++Builder XE6からWin64向けパッケージに対応!!

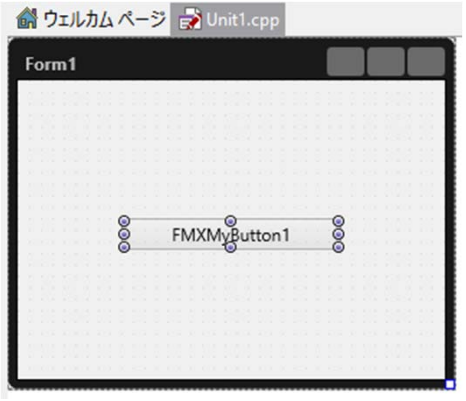
```
class PACKAGE TFMXMyButton : public TButton
{
private:
  int FMyProp;
protected:
public:
  __fastcall TFMXMyButton(TComponent* Owner);
  __fastcall virtual ~TFMXMyButton();
__published:
  __property int MyProp = {read=FMyProp,write=FMyProp};
};
extern PACKAGE bool MyFlag;
extern PACKAGE int __fastcall MyAdd(int x, int y);
```

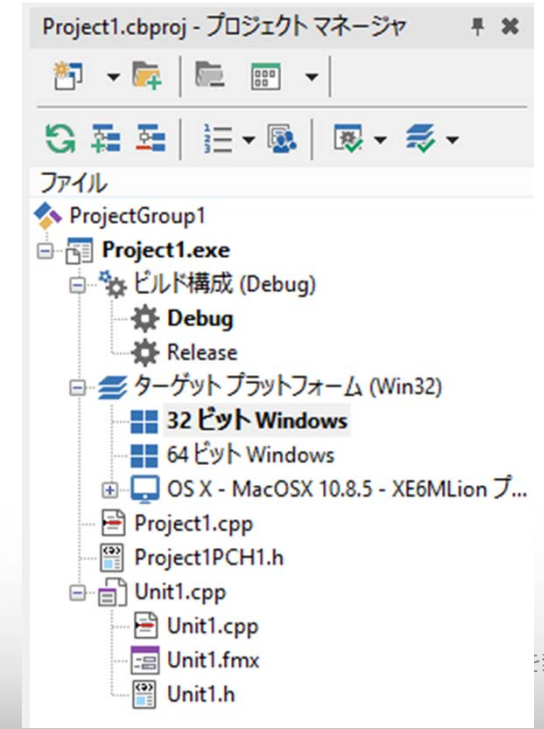
```
// アプリのコード
void __fastcall TForm1::FMXMyButton1Click(TObject *Sender)
{
  if (MyFlag) {
    ShowMessage(IntToStr(MyAdd(1,2)));
  }
}
```



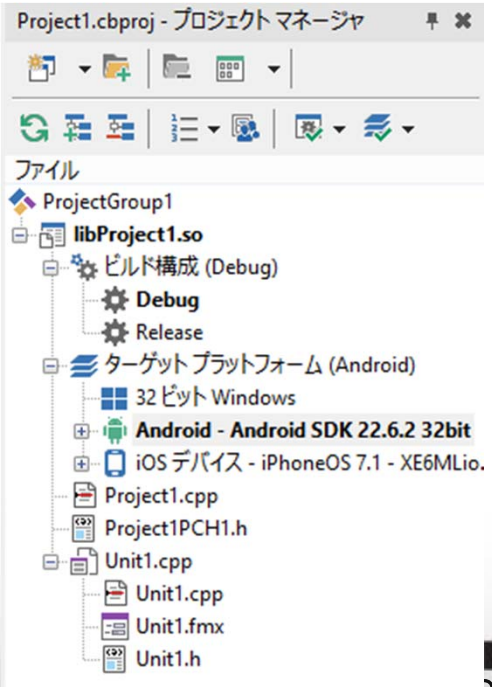
# パッケージを試す

Win32, Win64, OS X




iOS, Android


# モバイル向けコンパイラで Native[U]Intを試す

# Delphi由来のNativeInt型

- 例: TButtonクラスのTagプロパティ
  - Win32/OS X
    - typedef int NativeInt;
  - Win64
    - typedef __int64 NativeInt;
  - iOS/Android
    - NativeIntクラス型

sysmac.h

```
#define _DECLARE_ARITH_TYPE_ALIAS(Base, Alias)          ¥
  class Alias : public System::AliasT<Base, Alias> {    ¥
  public:                                               ¥
    Alias() : AliasType()                               ¥
    {}                                                  ¥
    constexpr explicit Alias(Base i) : AliasType(i)     ¥
    {}                                                  ¥
  private:                                              ¥
    Alias& operator=(const Base &rhs);                  ¥
  };

  // When mangled as strong aliases, Native[U]Int is[are]
not in the System namespace :(
  _DECLARE_ARITH_TYPE_ALIAS(int, NativeInt);
  _DECLARE_ARITH_TYPE_ALIAS(unsigned, NativeUInt);
```

# Delphi由来のNativeUInt型

- 例: TThreadクラスのThreadIDプロパティ
  - Win32/OS X
    - typedef unsigned int NativeUInt;
  - Win64
    - typedef unsigned __int64 NativeUInt;
  - iOS/Android
    - NativeUIntクラス型

sysmac.h

```
#define _DECLARE_ARITH_TYPE_ALIAS(Base, Alias)        ¥
  class Alias : public System::AliasT<Base, Alias> {  ¥
  public:                                             ¥
    Alias() : AliasType()                             ¥
    {}                                                ¥
    constexpr explicit Alias(Base i) : AliasType(i)   ¥
    {}                                                ¥
  private:                                            ¥
    Alias& operator=(const Base &rhs);                ¥
  };

  // When mangled as strong aliases, Native[U]Int is[are]
not in the System namespace :(
  _DECLARE_ARITH_TYPE_ALIAS(int, NativeInt);
  _DECLARE_ARITH_TYPE_ALIAS(unsigned, NativeUInt);
```

# Native[U]Int型の使い方

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
#if defined(TARGET_OS_IPHONE) || defined(__ANDROID__)
  //int& tag = Button1->Tag.get();
  int tag = (int)Button1->Tag;
#else
  NativeInt tag = Button1->Tag;
#endif
  switch(tag)
  {
    case 123:
      ShowMessage(IntToStr(tag));
      break;
    default:
      break;
  }
}
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
#if defined(TARGET_OS_IPHONE) || defined(__ANDROID__)
  //unsigned int& tid = TThread::CurrentThread->ThreadID.get();
  unsigned int tid = (unsigned int)TThread::CurrentThread->ThreadID;
#else
  unsigned int tid = TThread::CurrentThread->ThreadID;
#endif
  ShowMessage(IntToStr((int)tid));
}
```


# vector & shared_ptrを試す

# vectorとは?

- サイズ可変な配列を表すコンテナ
  - #include <vector>
  - std::vector
- Win32/Win64/OS X/iOS/Androidで共通のコード
  - Win32/Win64/OS X : Dinkumware
  - iOS : "/usr/include/c++/4.2.1/bits/stl_vector.h"
  - Android : "android-ndk-r9d/sources/cxx-stl/gnu-libstdc++/4.8/include/bits/stl_vector.h"

```
#include <vector>
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  std::vector<int> iv;
  iv.push_back(123);
  ShowMessage(IntToStr(iv[0]));
}
```

# shared_ptrとは?

- 参照カウンタで所有権を共有できるスマートポインタ
  - カウンタが0になるとリソースを解放する

```
#if defined(TARGET_OS_IPHONE)  // #if defined(__APPLE__) && defined(__arm__)
#include <tr1/memory>
using namespace std::tr1;
#else
  #if defined(__ANDROID__)
    #include <memory>
    using namespace std;
  #else
    #if defined(_WIN64)
      #include <memory>
      using namespace std;
    #else
      #if defined(__WIN32__)
        #include <boost/tr1/memory.hpp>
        using namespace std::tr1;
      #else // MacOSX
        #include <boost/tr1/memory.hpp>
        using namespace std::tr1;
      #endif
    #endif
  #endif
#endif
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  shared_ptr<int> p(new int(456));
  UnicodeString str = IntToStr(*p);
  ShowMessage(str);
}
```

# shared_ptrとは? (続き)

- Win32 : Boost 1.39
- Win64 : Dinkumware
- OS X : Boost 1.39
- iOS : "/usr/include/c++/4.2.1/tr1/boost_shared_ptr.h"
- Android : "android-ndk-r9d/sources/cxx-stl/gnu-libstdc++/4.8/include/bits/shared_ptr.h"

**配列 [ ] の場合**

```
#if defined(TARGET_OS_IPHONE)  // #if defined(__APPLE__) && defined(__arm__)
#include <tr1/memory>
using namespace std::tr1;
#else
  #if defined(__ANDROID__)
    #include <memory>
    using namespace std;
  #else
    #if 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
  #endif
#endif
void __fastcall TForm1::Button3Click(TObject *Sender)
{
#if defined(TARGET_OS_IPHONE) || defined(__ANDROID__) || defined(_WIN64)
  shared_ptr<int> p(new int[10], [](int* p){ delete [] p; });
#else
  shared_ptr<int> p(new int[10], default_delete<int[]>());
#endif
}
```


# Delphiの無名スレッド機能をC++で試す

# 無名スレッドとは?

- DelphiのRTLで、ワーカースレッドの生成と起動を行う
  - static TThread* __fastcall CreateAnonymousThread(const System::Sysutils::_di_TProc ThreadProc);
  - Win32/Win64/OS X/iOS/Androidのすべてで利用可能
  - ただし、Delphiの無名メソッドと同等の構文では呼び出せない

```
class TMyProc : public TCppInterfacedObject<TProc> {
public:
  __fastcall TMyProc(UnicodeString str) {
    value = str;
    //ShowMessage(L"new");      // GUI thread
  }
  __fastcall ~TMyProc() {
    //ShowMessage(L"delete");   // Non-GUI thread
  }
  void __fastcall Invoke() {    // Non-GUI thread
    TThread::Synchronize(NULL, UpdateGUI);
  }
private:
  UnicodeString value;
  void __fastcall UpdateGUI() { // GUI thread
    ShowMessage(value);
  }
};
```

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  UnicodeString str = L"OK";

  //_di_TProc f = new TMyProc(str);
  //TThread* t = TThread::CreateAnonymousThread(f);
  TThread* t = TThread::CreateAnonymousThread(
                                          new TMyProc(str));

  //t->FreeOnTerminate = false;
  t->Start();
  //t->WaitFor();
  //delete t;
}
```

# 無名スレッド - C++11のラムダ式を渡すには?

- Clang/LLVMベースのWin64/iOS/Androidは可能

```
#if defined(TARGET_OS_IPHONE)
#include <tr1/functional>
using namespace std::tr1;
#else
  #if defined(_WIN64) || defined(__ANDROID__)
    #include <functional>
    using namespace std;
  #endif
#endif

#if defined(TARGET_OS_IPHONE) || defined(_WIN64) || defined(__ANDROID__)
class TMyProcEx : public TCppInterfacedObject<TProc> {
public:
  __fastcall TMyProcEx(function<void()> f) {
    func = f;
  }
  __fastcall ~TMyProcEx() {
  }
  void __fastcall Invoke() {
    TThread::Synchronize(NULL, UpdateGUI);
  }
private:
  function<void()> func;
  void __fastcall UpdateGUI() {
    func();
  }
};
#endif
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
#if defined(TARGET_OS_IPHONE) || defined(_WIN64) || defined(__ANDROID__)
  UnicodeString str = L"OK";
  function<void()> f = [=]() { ShowMessage(str); };
  TThread* t = TThread::CreateAnonymousThread(new TMyProcEx(f));
  t->Start();
#endif
}
```


# FireMonkeyの
# プラットフォームサービスを試す

# FMX.Platformユニット(名前空間)

- 単一コードでマルチデバイス対応のFireMonkeyアプリを作成するのに重要
  - スクリーンサイズ(論理的サイズ)の取得 や 画素密度(スケール)の取得

Delphiコード

```
uses FMX.Platform;

procedure TForm1.Button1Click(Sender: TObject);
var os: TOSVersion;
begin
  if (os.Platform = pfMacOS) AND ((TPlatformServices.Current.GetPlatformService(IFMXScreenService) as
IFMXScreenService).GetScreenScale >= 2.0) then
    ShowMessage('Mac Retina Display');
end;
```

C++コード

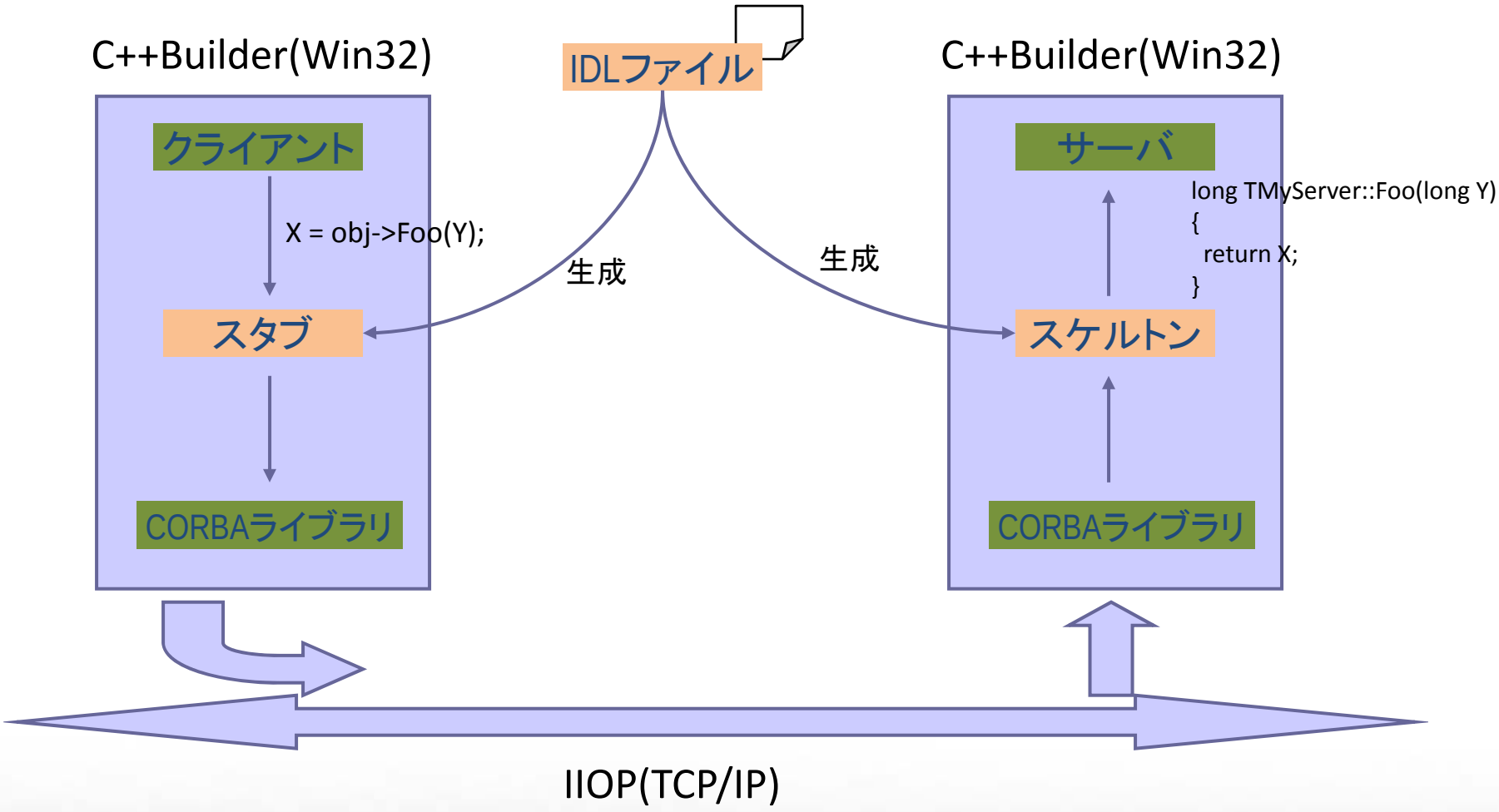
```
#include <FMX.Platform.hpp>
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TOSVersion os;
  if((os.Platform == TOSVersion::TPlatform::pfMacOS) &&
(((_di_IFMXScreenService)TPlatformServices::Current->GetPlatformService(__uuidof(IFMXScreenService)))-
>GetScreenScale() >= 2.0))
    ShowMessage("Mac Retina Display");
}
```


# C++ CORBA

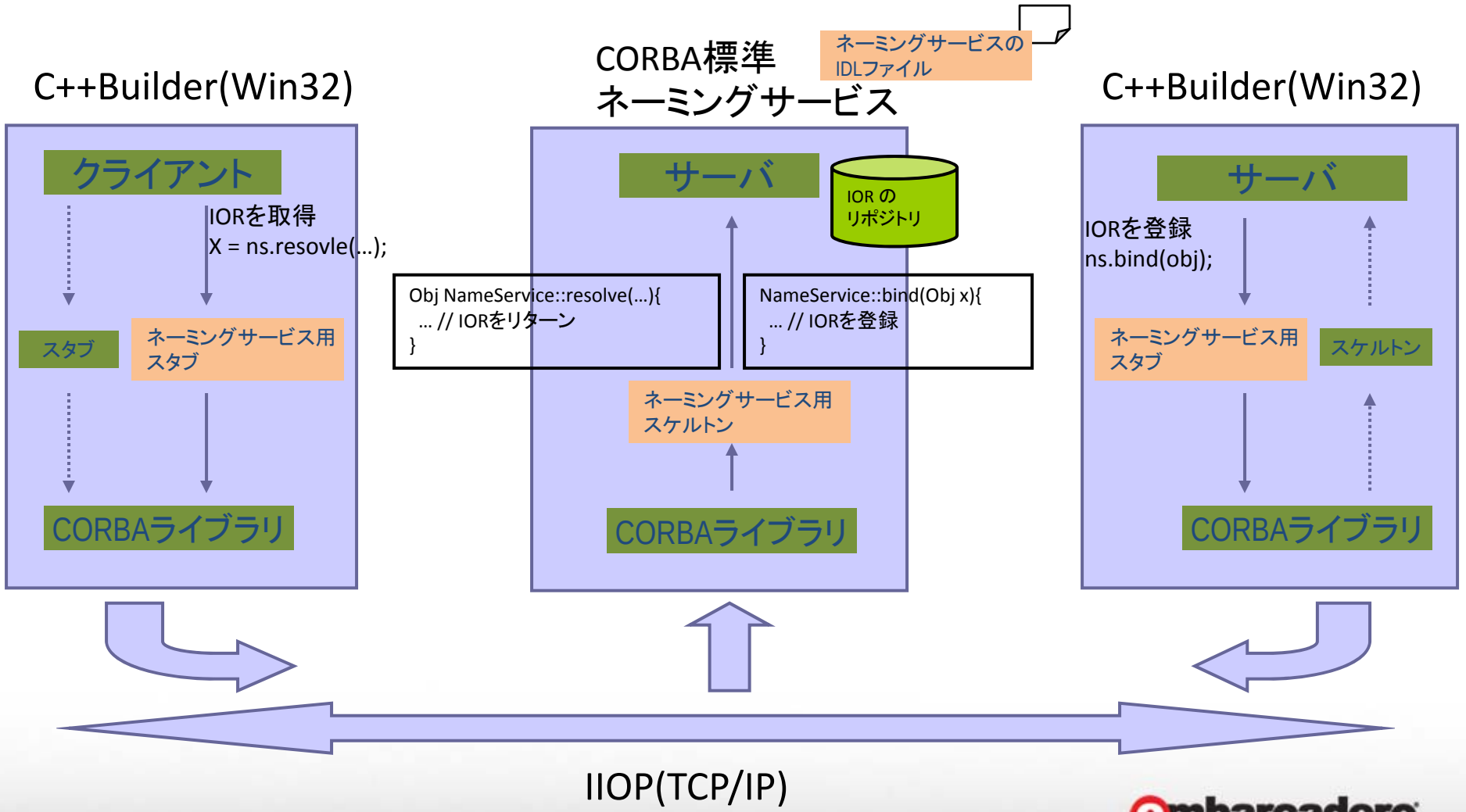
# CORBAとは?

- CORBA(Common Object Request Broker Architecture)の良いところ
  - あらかじめ決定しておく必要がない
    - プログラミング言語(Java, C++, .NET etc.)
    - ハードウェアプラットフォーム
    - オペレーティングシステム
    - サーバ/クライアントの分散の度合い
      - サーバプロセスをネットワークの何処に配置するか?
      - 各種サーバコンポーネントを何処のプロセス内にどれだけ配置するか?
      - サーバコンポーネントをリモートプロセス/インプロセスのどちらで実行するか?
  - CORBAの鍵となる特性は相互運用性である
    - プラットフォーム、言語、ベンダをまたいだ相互運用性
    - CORBA製品やサービスのプラグ&プレイ による代用性
- DataSnapに似てますね...

# CORBAの一般的な仕組み



# CORBAの一般的な仕組み(続き)



# C++Builderで使えるCORBA実装

- Micro Focus VisiBroker 8.5
  - C++Builder XE
  - http://supportline.microfocus.com/supportresources/VB85prodmatrix.aspx
- オープンソースのC++ CORBA実装「ACE+TAO」
  - [参考: オープンソースを利用した 3層C/Sシステムの構築方法]
  - http://edn.embarcadero.com/print/images/33673/2nd_devcamp_06.pdf
  - v6.2.1 + C++Builder XE4(bcc32)
  - v6.2.5 + C++Builder XE5/XE6(bcc32)

```
cd C:¥ACE_wrappers
echo #include "ace/config-win32.h"> ace¥config.h
set ACE_ROOT=C:¥ACE_wrappers
set TAO_ROOT=%ACE_ROOT%¥TAO
set PATH=%ACE_ROOT%¥lib;%ACE_ROOT%¥bin;%PATH%
cd %ACE_ROOT%¥ace
%ACE_ROOT%¥bin¥mwc.pl -type bmake
set RELEASE=1
make -f Makefile.bmak all
cd %TAO_ROOT%
%ACE_ROOT%¥bin¥mwc.pl TAO.mwc -type bmake
make -f Makefile.bmak all
```

# IDLを書いてみる

- IDL - Interface Definition Language(インタフェース定義言語)
  - Demo.idl（拡張子は一般的に .idl）
- module(名前空間), interface, メソッド, 引数, 戻り値 などを定義する

```
// Demo空間
module Demo {
  // 社員データ
  struct Employee {
    long id;      // 社員番号
    wstring data; // 社員情報
  };
  // アプリケーション例外を定義
  exception NoSuchEmployee {
    wstring reason;
  };
  // インターフェース
  interface EmployeeManager {
    // 社員番号で検索する
    Employee findByPrimaryKey(in long id) raises(NoSuchEmployee);
  };
};
```

マルチバイトなのでwstring(UNICODE)

in はインプット用引数の意味(※)

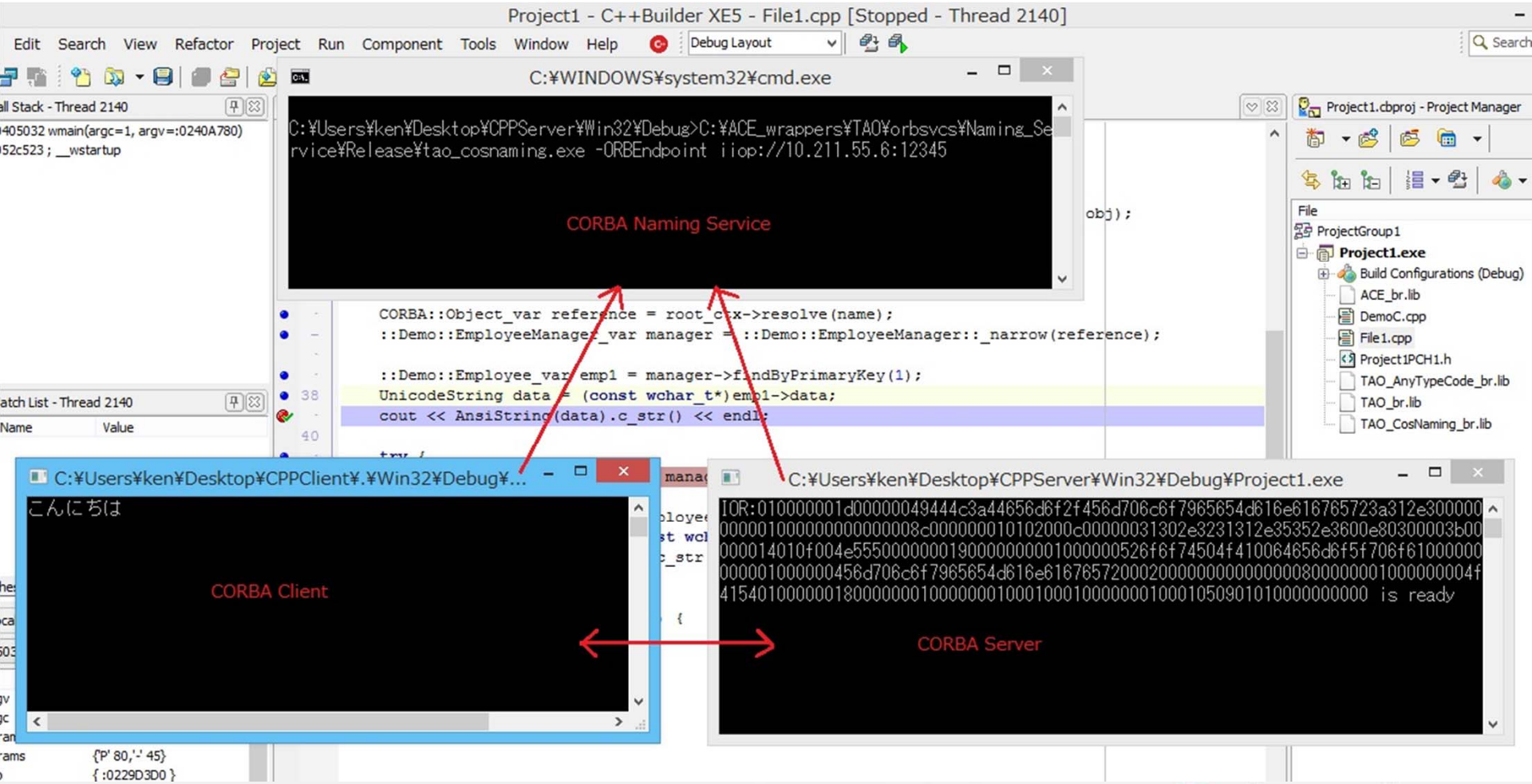
例外をスローすることがある

戻り値(※)

(※)CORBAでは一般的にNULL(nil)は扱えません

# C++ CORBAを試す

- C++Builder XE5 Update2(bcc32) + ACE/TAO 6.2.5




# Q & A

Thank you.