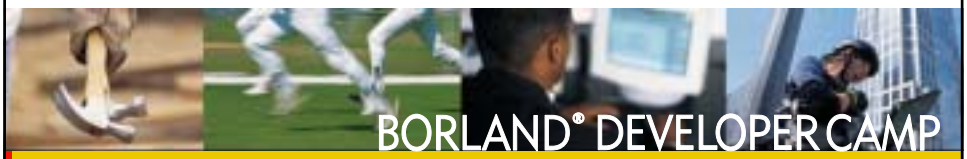
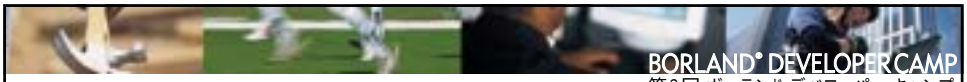


「gSOAP, Pthreads-Win32, OpenSSL を使った SOAP C/Sアプリ開発」

ボーランド株式会社
Developer Tools Group
高橋智宏

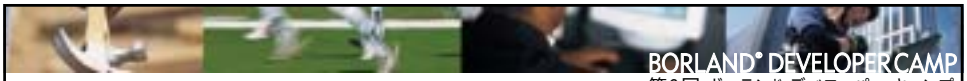
Borland



講師紹介

- 高橋智宏
- 1973年生まれ、京都大学 法学部卒
- 学生の時購入したTurboC++2ndからの熱狂的なボーランドファン
- 参加しているメーリングリストやコミュニティ
  - JBuilder ML,C++Builder ML,Delphi ML,C# ML,CORBA ML 等…
  - ミクシィ
    - http://mixi.jp/show_friend.pl?id=208738
- 「Java読書会」を運営
  - http://www.javareading.com/bof/

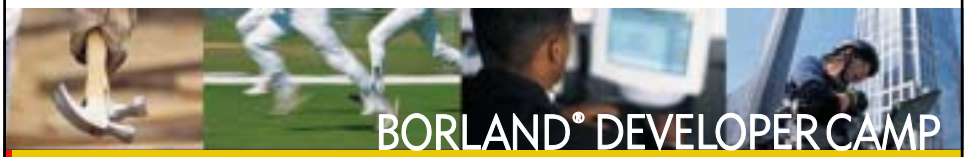
Borland



アジェンダ

- SOAPのおさらい
  - プロトコル, WSDL, スタブ/スケルトン
- IDL2WSDLコマンド
  - IDLからWSDLを生成してみよう
- gSOAP + Webサーバ
  - とりあえずCGI形式で実装
  - SOAPクライアントの作成
- gSOAP + Pthreads-Win32
  - CGIからマルチスレッドで動作する単体サーバに書き換える
- gSOAP + Pthreads-Win32 + OpenSSL
  - さらに SOAP over SSL を導入する
  - Delphiクライアントから SSLで接続する

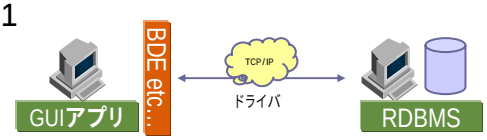
Borland



SOAPのおさらい

C/Sの形態

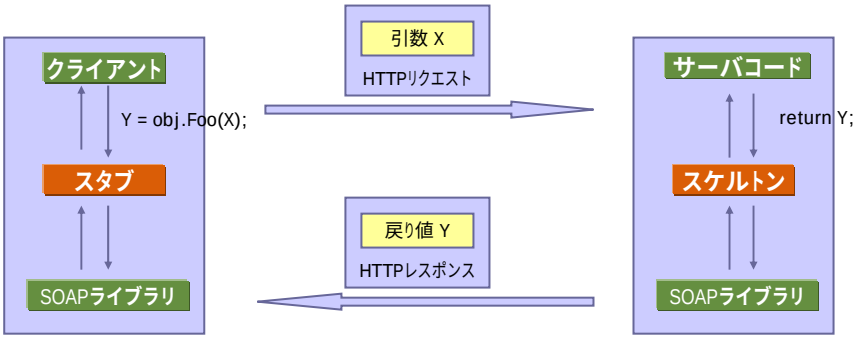
- その1



- その2 - 今回作成します

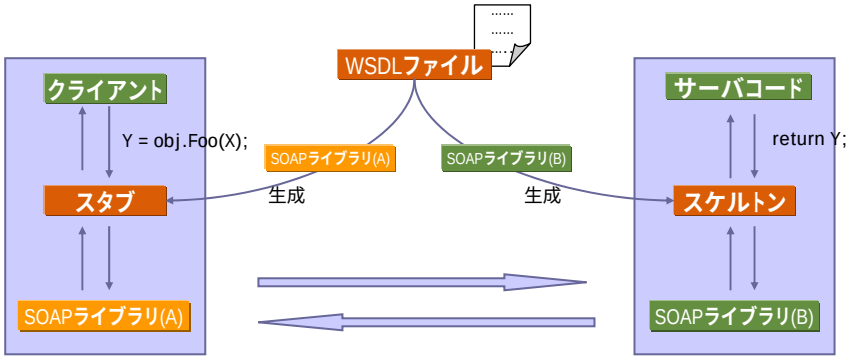
一般的なSOAPのプロトコル

- XML形式のデータをHTTP上で送受信する



WSDL

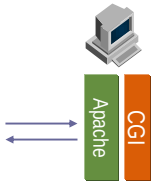
- Webサービス記述言語 (Web Services Description Language)
- 提供するWebサービスのインターフェース (メソッド等) を規定する
- WSDLファイルから、クライアント/サーバ各用のスタブ/スケルトンコードを生成する



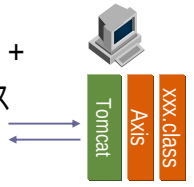
Borland

一般的なSOAPサーバの形態

- Apache + CGI

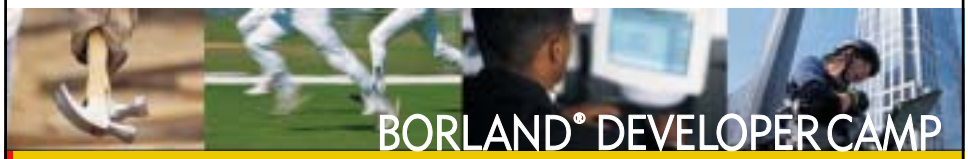


- Tomcat + Apache Axis(サーブレット) + 実装クラス



- IIS + ASP.NET + 実装クラス

Borland



IDL2WSDLコマンド

WSDLの生成方法

- WSDLファイルの中身はXML形式。WSDLを直接手書きすることは無理
- 一般的にWSDLを生成するには、先に特定の実装言語でメソッド等を記述して、それをWSDLに変換する必要がある。時には配布＆実行まで必要な場合も…

Demo.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://tempuri.org" xmlns:impl="http://tempuri.org" xmlns:intf="http://tempuri.org"
xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns1="http://CORBA.omg.org"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://tempuri.org">
   <import namespace="http://CORBA.omg.org"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="Employee">
    <sequence>
     <element name="id" type="xsd:int"/>
     <element name="data" nillable="true" type="soapenc:string"/>
    </sequence>
   </complexType>
   <complexType name="NoSuchEmployee">
    <complexContent>
     <extension base="tns1:UserException">
      <sequence>
       <element name="reason" nillable="true" type="soapenc:string"/>
      </sequence>
     </extension>
    </complexContent>
   </complexType>
  </schema>
  <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://CORBA.omg.org">
   <import namespace="http://tempuri.org"/>
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType abstract="true" name="UserException">
    <sequence/>
   </complexType>
  </schema>
 </wsdl:types>
  <wsdl:message name="findByPrimaryKeyRequest">
    <wsdl:part name="in0" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="findByPrimaryKeyResponse">
    <wsdl:part name="findByPrimaryKeyReturn" type="impl:Employee"/>
  </wsdl:message>
  <wsdl:message name="NoSuchEmployee">
    <wsdl:part name="fault" type="impl:NoSuchEmployee"/>
  </wsdl:message>
  <wsdl:portType name="EmployeeManager">
    <wsdl:operation name="findByPrimaryKey" parameterOrder="in0">
```

Borland

そこで IDLファイルが使えないものかと...

- IDL – Interface Definition Language
  - インターフェース(メソッド等)を定義するプラットフォーム非依存の言語
  - 前回の「第2回デベロッパーキャンプ」の資料を参考にしてください
- 先にIDLでインターフェースを記述しておいて、後でWSDLファイルに変換できないものか？

Demo.idl

社員番号から社員データを検索するインターフェース

```
module Demo {
  struct Employee {
    long id;
    wstring data;
  };
  exception NoSuchEmployee {
    wstring reason;
  };
  interface EmployeeManager {
    Employee findByPrimaryKey(in long id) raises(NoSuchEmployee);
  };
};
```

引数: 社員番号

戻り値: 社員データ

IDLからWSDLを生成するには

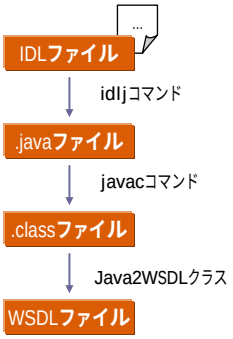
- Borland VisiBroker 7.0 には IDLからWSDLファイル等を生成するコマンドが用意されている

が...

- 今回は、無償のもので代用します
- 用意するもの
  - JDK (Java Development Kit)
  - Apache Axis (特にその中の Java2WSDL コマンド)
    - Web Services Description Language for Java Toolkit (WSDL4J)
  - JavaMail API (※Apache Axisに必要)
  - JavaBeans Activation Framework (JAF) (※Apache Axisに必要)

IDLからWSDLを生成するには（続き）

- 変換処理の流れ



IDLからWSDLを生成する手順

- JDKをダウンロードしてインストール
  - http://java.sun.com/j2se/1.4.2/download.html
  - idljコマンド，javacコマンド，javaコマンドを使用します
- Apache Axis 1.4 をダウンロードして展開
  - http://ftp.kddilabs.jp/infosystems/apache/ws/axis/1_4/axis-bin-1_4.zip
  - libフォルダ内のすべての.jarファイルを使用します
- JavaMail API 1.4 をダウンロードして展開
  - http://java.sun.com/products/javamail/downloads/index.html
  - 付属の mail.jar のみを使用します
- JavaBeans Activation Framework 1.0.2 をダウンロードして展開
  - http://java.sun.com/products/archive/javabeans/jaf102.html
  - 付属の activation.jar のみを使用します

IDLからWSDLを生成する手順（続き）

- JDKのbinフォルダにPATHが通っていることを確認
- IDLファイル（先に紹介した Demo.idl ）を用意する
- idljコマンドで、Demo.idlファイルから、対応するJavaソースコードを生成
  - c:¥...>idlj Demo.idl
  - findByPrimaryKey(int id) というメソッドを含んだ“Demo.EmployeeManagerOperations インターフェース” が生成される
- Demo¥EmployeeManagerOperations.java をコンパイルする
  - c:¥...>javac Demo¥EmployeeManagerOperations.java
  - Demo¥EmployeeManagerOperations.class 等が生成される

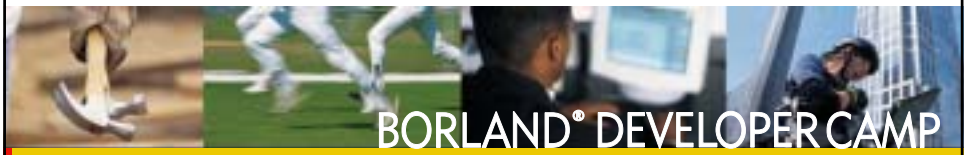
BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

IDLからWSDLを生成する手順（続き）

- Apache Axis の Java2WSDLクラス を利用するために、環境変数CLASSPATHを設定する
  - カレントディレクトリ + Axisのlibフォルダにある全jarファイル + JavaMail API + JAF
  - c:¥...>set CLASSPATH=.;C:¥axis-1_4¥lib¥axis.jar;C:¥axis-1_4¥lib¥axis-ant.jar;C:¥axis-1_4¥lib¥commons-discovery-0.2.jar;C:¥axis-1_4¥lib¥commons-logging-1.0.4.jar;C:¥axis-1_4¥lib¥jaxrpc.jar;C:¥axis-1_4¥lib¥log4j-1.2.8.jar;C:¥axis-1_4¥lib¥saaj.jar;C:¥axis-1_4¥lib¥wsdl4j-1.5.1.jar;C:¥axis-1_4¥lib¥mail.jar;C:¥axis-1_4¥lib¥activation.jar
- Java2WSDLクラスを使って、Javaのクラス（今回はインターフェース）からWSDLファイルを生成する
  - c:¥...>java org.apache.axis.wsdl.Java2WSDL -o Demo.wsdl -n http://tempuri.org -l https://localhost:18081 -s EmployeeManager -b EmployeeManagerSoapBinding -P EmployeeManager Demo.EmployeeManagerOperations

IDLからWSDLを生成する手順（続き）

- Java2WSDLクラスに必要なオプション
  - -o ー 生成されるWSDLファイル名（例: Demo.wsdl）
  - -n ー WSDL内で使用するXMLのnamespace（例: http://tempuri.org）
  - -l ー WebサービスにアクセスするためのURL（例: https://localhost:18081）
  - 引数ー WSDLに変換されるJavaクラス（例: Demo.EmployeeManagerOperations）



gSOAP（CGI形式） + Webサーバ

gSOAPって？

- WSDLも手に入って、いよいよWebサービスを作成してみます
- gSOAP
  - 5年以上の歴史を持つのC/C++向け汎用SOAPライブラリ
  - オープンソース、ライセンスはMPL1.1がベース
  - Windows,各種UNIX,MacOSX、各種コンパイラに対応している
  - 構築に必要なファイルは、stdsoap2.cpp,stdsoap2.h のたった2個!!
  - URL: http://www.cs.fsu.edu/~engelen/soap.html
- WSDLファイルを読み込んで、C/C++用のスタブ/スケルトンを生成するツールが付属
- CGI形式、単独実行形式のサーバ、SSL（OpenSSL）をサポート

gSOAP と 無償で使えるC++開発環境 を準備

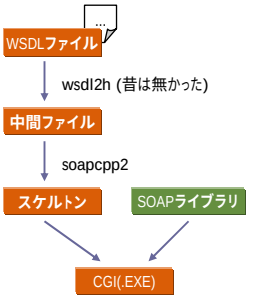
- SourceForgeから最新の gSOAP 2.7.9 をダウンロードして展開
  - http://jaist.dl.sourceforge.net/sourceforge/gsoap2/gsoap_win32_2.7.9.zip
  - 日本語を含む文字列を char*（デフォルト）ではなく std::wstring として処理するため、設定ファイル <gSOAP_Dir>¥typemap.dat を編集する

コメントアウト →
```
# xsd__string = | char* | char*
xsd__string = | std::wstring | std::wstring
SOAP_ENC__string = | std::wstring | std::wstring
```
} 2行追加

- Turbo C++ Explorer版をダウンロードしてインストール
  - http://www.borland.com/downloads/download_turbo.html

CGIを作成する

- コンソールアプリケーションを新規作成する
  - VCL は使わない
  - マルチスレッド を使う (後の改良に備えて)
- gSOAPの「wsdl2hコマンド」を使って、WSDLファイル (Demo.wsdl) から、C言語のヘッダファイル風な中間ファイル (Demo.h) を生成する
  - c:¥...>wsdl2h -t <gSOAP_Dir>¥typemap.dat -o Demo.h Demo.wsdl
- gSOAPの「soapcpp2コマンド」を使って、中間ファイル (Demo.h) から、スケルトン (.cpp,.h等) を生成する
  - c:¥...>soapcpp2 -S -L -I <gSOAP_Dir>¥import Demo.h
    - soapC.cpp
    - soapServer.cpp
    - soapH.h
    - EmployeeManagerSoapBinding.nsmap
- プロジェクトに以下のファイルを追加
  - soapC.cpp ― スケルトン
  - soapServer.cpp ― スケルトン
  - stdsoap2.cpp ― gSOAP付属のSOAPライブラリ



Borland

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

CGIを作成する (続き)

- main関数と、Webサービスのサーバメソッドを実装

```
#include "soapH.h"
#include "EmployeeManagerSoapBinding.nsmap"        } スケルトン用ヘッダ

int main(int argc, char* argv[])
{
  struct soap* soap = soap_new();
  soap_serve(soap);                                } CGI処理用 (gSOAPのライブラリを使用)
  return 0;
}

SOAP_FMAC5 int SOAP_FMAC6 ns1__findByPrimaryKey(struct soap* soap,
                                                int _in0,
                                                struct ns1__findByPrimaryKeyResponse &_param_1)
{
  ns1__Employee* employee = soap_new_ns1__Employee(soap, -1);
  try {
    employee->id   = _in0;
    employee->data = L"たかはし";
  }
  catch(...) {
    soap_delete_ns1__Employee(soap, employee);
    return soap_sender_fault(soap, "Error!!", NULL);
  }
  _param_1._findByPrimaryKeyReturn = employee;
  return SOAP_OK;
}
```

引数: 社員番号

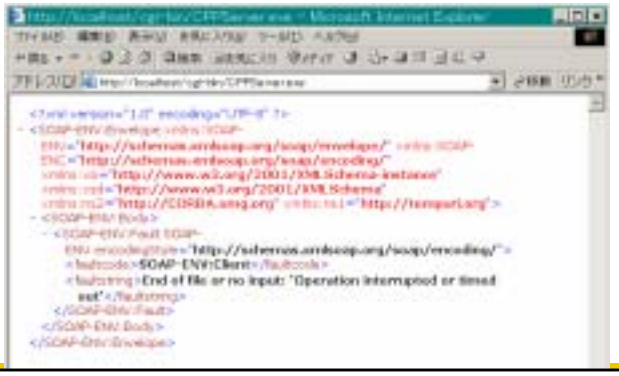
戻り値保持用の構造体

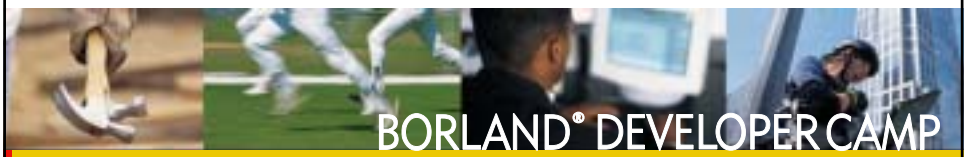
戻り値: 社員データ構造体は専用の関数を使って new する delete はスケルトンが自動的に行う

Borland

CGIを配布する

- Apache2.0.x をインストールし、CGIが利用できるよう設定変更し、起動する
- ビルドしたCGI (CPPServer.exe) を <Apache_Dir>¥cgi-bin 等に配布する
  - 他には、Cランタイム (cc3270mt.dll) が必要
- IE等のWebブラウザで、とりあえずWebサービスにアクセスしてみる
  - もちろん、エラーを示すSOAPレスポンスが返るが、起動自体は成功！

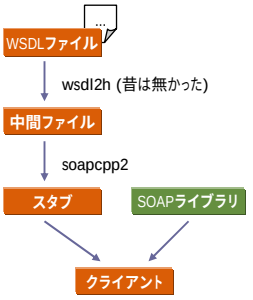




SOAPクライアント (gSOAP) の作成

スタブおよびクライアントの生成

- コンソールアプリケーションを新規作成する
  - VCL は使わない
- gSOAPの「wsdl2hコマンド」を使って、WSDLファイル (Demo.wsdl) から、C言語のヘッダファイル風な中間ファイル (Demo.h) を生成する
  - c:¥...>wsdl2h -t <gSOAP_Dir>¥typemap.dat -o Demo.h Demo.wsdl
- gSOAPの「soapcpp2コマンド」を使って、中間ファイル (Demo.h) から、スタブ (.cpp, .h等) を生成する
  - c:¥...>soapcpp2 -C -L -I <gSOAP_Dir>¥import Demo.h
    - soapC.cpp
    - soapClient.cpp
    - soapEmployeeManagerSoapBindingProxy.h
    - EmployeeManagerSoapBinding.nsmap
- プロジェクトに以下のファイルを追加
  - soapC.cpp ― スタブ
  - soapClient.cpp ― スタブ
  - stdsoap2.cpp ― gSOAP付属のSOAPライブラリ



Borland

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

クライアントコードを実装する

- main関数内に、Webサービスへのアクセスコードを実装

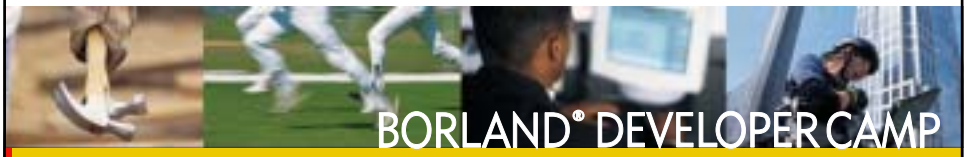
```
...
#include "soapEmployeeManagerSoapBindingProxy.h"   } スタブ用ヘッダ
#include "EmployeeManagerSoapBinding.nsmap"

int main(int argc, char* argv[])
{                                                  Webサービスアクセス用オブジェクト
  EmployeeManagerSoapBinding* service = new EmployeeManagerSoapBinding();
  service->endpoint = "http://localhost/cgi-bin/CPPServer.exe";

  int id = 99;                  引数: 社員番号
  ns1__findByPrimaryKeyResponse resp;   戻り値保持用の構造体

  int soap_ret = service->ns1__findByPrimaryKey(id, resp);   Webサービスのメソッドを呼び出す
  if( soap_ret == SOAP_OK )
  {
    cout << resp->_findByPrimaryKeyReturn->id << endl;   戻り値: 社員データの社員番号(99)を表示
  }

  delete service;
  return 0;
}
```

Borland

クライアントコードを実行する

- プロジェクトをビルドして実行する

- 注意するポイント
  - 通常、Webサービスへのアクセスポイントは WSDL ファイル内で定義済み
    - スタブのヘッダファイル「soapEmployeeManagerSoapBindingProxy.h」に自動的に埋め込まれる
    - Webサービスへのアクセスポイントを独自に変更することも可能
      例: service->endpoint = "http://localhost/cgi-bin/CPPServer.exe";
  - 社員データ(ns1__Employee型)内の std::wstring型メンバは、コンソールに正しく出力できないので、社員番号(int型)のみ出力しています
    - 後ほどDelphi/BCBのGUIクライアントでWideString型として出力します



POSIX Threads を使った単独サーバ

gSOAPの単独SOAPサーバとしての機能

- gSOAPはCGI形式をサポートしますが、SOAPサーバとしてApache等のWebサーバを別途必要とします
- gSOAPは、.exe単体でSOAPサーバの機能も提供しています
  - 単独サーバのためのコードはとても簡単
  - ただし、処理をマルチスレッド化する部分は直接は用意されていない
    - デフォルトでは、同時に1リクエストしか処理できない！
    - スレッドの生成、起動、破棄は自前で行う必要がある！！
- スレッドライブラリ
  - Windowsでは、Win32 API を使うのが一般的
    - CreateThreadや_beginthreadなど
  - UNIXでは、POSIX Threads（Pthreads）を使うのが一般的
    - pthread_createなど

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

Windowsで POSIX Threads（Pthreads）を使う

- Win32 API をそのまま使って、マルチスレッド化を実現しても良いが...
  - gSOAPは、汎用的なC/C++用ライブラリだし...
  - 自分でWin32 APIを直接使うよりも、既存のライブラリを使ったほうが...
- そこで、POSIX Threads を Win32 API で実装したものを利用
  - Pthreads-Win32
  - http://sourceware.org/pthreads-win32/
  - オープンソース
  - ライセンスは LGPL

Pthreads-Win32 のセットアップ

- 最新版 2.7.0 をダウンロードする
  - ftp://sources.redhat.com/pub/pthreads-win32/pthreads-w32-2-7-0-release.exe
- c:¥pthreads2.7.0 等にインストールする
- ボーランドのインラインアセンブラ (TASM32) の機能を使うので、
  c:¥pthreads2.7.0¥pthreads.2¥config.h を以下のように書き換える

  /*#undef HAVE_TASM32*/

  #define HAVE_TASM32
- c:¥pthreads2.7.0¥pthreads.2 に移動して、
  - c:¥...>make -fBmakefile
  - 必要があれば、ボーランド用のメイクファイル (Bmakefile) を編集する
- c:¥pthreads2.7.0¥pthreads.2 に DLLとインポートライブラリが生成される
  - pthreadBC2.dll
  - pthreadBC2.lib

CGIを単独SOAPサーバへ変更する

main関数部

```
...
#include <pthread.h>
...
int main(int argc, char* argv[])
{
  struct soap soap;
  soap_init(&soap);
  serversock = soap_bind(&soap, "localhost", 80, 100);
  if( serversock < 0 )
  {
    soap_print_fault(&soap, stderr);
    return -1;
  }

  for(;;) {
    int s = soap_accept(&soap);
    if( s < 0 )
      break;

    struct soap *tsoap = soap_copy(&soap);
    pthread_t tid;
    pthread_create(&tid, NULL, process_request, (void*)tsoap);
  }
  soap_end(&soap);
  soap_done(&soap);
  return 0;
}
```

localhostの80番ポートで起動する

クライアントを受け付ける

SOAP用の構造体を複製

スレッドを起動し、スレッドにSOAP用の構造体を渡す

CGIを単独SOAPサーバへ変更する（続き）

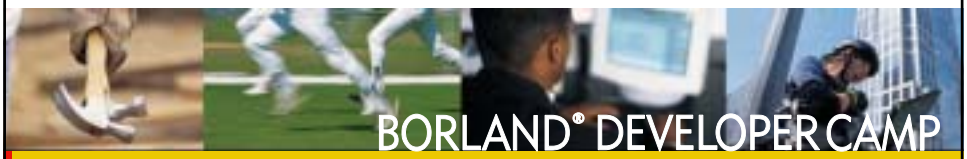
スレッド実装部

```
...
#include <pthread.h>
...
static void *process_request(void *tsoap)   ← スレッドのエントリポイント
{
  pthread_detach(pthread_self());

  struct soap* soap = (struct soap*)tsoap;
  soap_serve(soap);                          ← ns1__findByPrimaryKeyメソッドを呼び出し
  soap_destroy(soap);
  soap_end(soap);
  soap_free(soap);                           ← SOAP用の構造体を破棄

  return NULL;
}
```

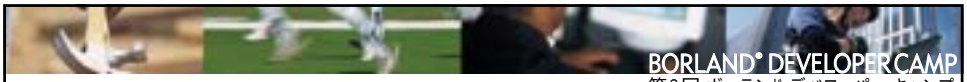
- プロジェクトのインクルードディレクトリに「c:¥pthreads2.7.0¥pthreads.2」を追加する
- プロジェクトに、インポートライブラリ（pthreadBC2.lib）を追加してビルドする

SOAPサーバ と SOAPクライアントを試す

- マルチスレッド版SOAPサーバ（CPPServer.exe）を起動する
  - Cランタイム（cc3270mt.dll）が必要
  - Pthreads-Win32のDLL（pthreadBC2.dll）も必要
- クライアントはアクセスするWebサービスのURLを変更してビルドするだけ
  - 例: service->endpoint = "http://localhost/";
- クライアントを起動する
- gSOAPサーバの注意点
  - SOAPサーバをシャットダウンする処理は省きました
    - soap_bind関数の戻り値（サーバソケット）をクローズします
  - スレッドプールの機能は実装していません



さらに、SSL (OpenSSL) を使ったSOAPサーバへ



gSOAP と SSL (OpenSSL)

- gSOAPのライブラリファイル stdsoap2.cpp, stdsoap2.h は、SSL (OpenSSL) を使ったSOAP通信をサポートしている

```
#ifdef WITH_OPENSSL
...
#endif
```

- コンパイル時に WITH_OPENSSL の定義 (define) が必要
- もちろん、OpenSSL のヘッダやライブラリ群も必要
- OpenSSL
  - オープンソース
  - http://www.openssl.org/
  - ライセンスは Apache (BSD) スタイル

Borland

OpenSSLのセットアップ

- 最新の OpenSSL をダウンロードし、展開する
  - http://www.openssl.org/source/openssl-0.9.8d.tar.gz
- makeファイルを生成するために Perl が必要なので、予めインストールしておく
- OpenSSLのインストールディレクトリに移動し、makeファイルを生成し、ビルドする
  - c:¥...>ms¥bcb4.bat
  - c:¥...>make –fbcb.mak
- ヘッダファイル(.h)は、<OpenSSL_dir>¥inc32 に生成される
- 各種コマンド(.exe)やスタティックライブラリ(.lib)は、<OpenSSL_dir>¥out32 に生成される

Borland

SOAPサーバへの変更

main関数部 前半部

```
...
int CRYPTO_thread_setup();
void CRYPTO_thread_cleanup();          } OpenSSLの初期化&クリーンアップ用の自作関数
...
int main(int argc, char* argv[])
{
  soap_ssl_init();
  CRYPTO_thread_setup();               } OpenSSL関連の初期化

  struct soap soap;
  soap_init(&soap);
  if( soap_ssl_server_context(&soap,
                              SOAP_SSL_REQUIRE_SERVER_AUTHENTICATION,
                              "server.pem",  // サーバ証明書と秘密鍵
                              "password",    // パスワード
                              NULL,
                              NULL,                                  } SSLサーバのセットアップ
                              "dh512.pem",  // 鍵交換に DHE を使う（RSAではなく）
                              NULL,
                              "sslserver"))  // SSLセッションの管理に使用するID
  {
    soap_print_fault(&soap, stderr);
    return -1;
  }
  serversock = soap_bind(&soap, "localhost", 18081, 100);
  if( serversock < 0 )
  {                                                  SSLは18081番ポートを使用
    soap_print_fault(&soap, stderr);
    return -1;
  }
```

Borland

SOAPサーバへの変更（続き）

main関数部
後半部

```
...
...
for(;;)
{
  int s = soap_accept(&soap);
  if( s < 0 )
    break;

  struct soap *tsoap = soap_copy(&soap);
  if( soap_ssl_accept(tsoap) )
  {
    soap_print_fault(tsoap, stderr);
    soap_end(tsoap);
    soap_free(tsoap);
    continue;
  }

  pthread_t tid;
  pthread_create(&tid, NULL, process_request, (void*)tsoap);
}
soap_end(&soap);
soap_done(&soap);

CRYPTO_thread_cleanup();

return 0;
}
```

SSLによるクライアントの受け付け処理

スレッドの起動に関する処理などはそのまま

OpenSSL関連のクリーンアップ

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

SOAPサーバへの変更（続き）

- OpenSSLは、マルチスレッド環境下での利用を想定しているものの、「スレッドの識別」「排他制御（ロック）」に関しては、利用者側が提供することを期待している
  - 一定個数のロックオブジェクトを事前に生成する
  - 動的なロックのため、ロックオブジェクトを含んだ構造体を定義する
  - OpenSSLに、ロック処理を実装した自作関数（5個）のアドレスを登録する
- OpenSSLの書籍やgSOAP付属のサンプルでは、排他制御（ロック）に関するコードを実装する際に「マクロ」を使っています

```
#if defined(WIN32)
# define MUTEX_TYPE       HANDLE
# define MUTEX_SETUP(x)   (x) = CreateMutex(NULL, FALSE, NULL)
# define MUTEX_CLEANUP(x) CloseHandle(x)
# define MUTEX_LOCK(x)    WaitForSingleObject((x), INFINITE)
# define MUTEX_UNLOCK(x)  ReleaseMutex(x)
# define THREAD_ID        GetCurrentThreadId()
#else
# define MUTEX_TYPE       pthread_mutex_t
# define MUTEX_SETUP(x)   pthread_mutex_init(&(x), NULL)
# define MUTEX_CLEANUP(x) pthread_mutex_destroy(&(x))
# define MUTEX_LOCK(x)    pthread_mutex_lock(&(x))
# define MUTEX_UNLOCK(x)  pthread_mutex_unlock(&(x))
# define THREAD_ID        pthread_self()
#endif
```

Windowsでは、Win32 API を使う

UNIXでは、PThreads を使う

Win32用のPThreadsを手に入れたので、PThreadsに統一します

SOAPサーバへの変更（続き）

OpenSSL関連のコード
前半部

```
static pthread_mutex_t* mutex_buf = NULL;

static void locking_function(int mode, int n, const char *file, int line)
{
  if( mode & CRYPTO_LOCK )
    pthread_mutex_lock(&(mutex_buf[n]));
  else
    pthread_mutex_unlock(&(mutex_buf[n]));
}

struct CRYPTO_dynlock_value {
  pthread_mutex_t mutex;
};

static CRYPTO_dynlock_value* dyn_create_function(const char *file, int line)
{
  CRYPTO_dynlock_value* retval = (CRYPTO_dynlock_value*)malloc(sizeof(CRYPTO_dynlock_value));
  if( retval )
    pthread_mutex_init(&(retval->mutex), NULL);
  return retval;
}

static void dyn_lock_function(int mode, CRYPTO_dynlock_value* l, const char *file, int line)
{
  if( mode & CRYPTO_LOCK )
    pthread_mutex_lock(&(l->mutex));
  else
    pthread_mutex_unlock(&(l->mutex));
}

static void dyn_destroy_function(CRYPTO_dynlock_value* l, const char *file, int line)
{
  pthread_mutex_destroy(&(l->mutex));
  free(l);
}
```


Borland

BORLAND® DEVELOPER CAMP
第3回 ボーランド デベロッパー キャンプ

SOAPサーバへの変更（続き）

OpenSSL関連のコード
後半部

```
static unsigned long id_function()
{
#ifdef WIN32
  return (unsigned long)GetCurrentThreadId();
#else
  return (unsigned long)pthread_self();
#endif
}

int CRYPTO_thread_setup() {
  mutex_buf = (pthread_mutex_t*)malloc(CRYPTO_num_locks() * sizeof(pthread_mutex_t));
  if( !mutex_buf )
    return SOAP_EOM;
  for(int i = 0; i < CRYPTO_num_locks(); i++) {
    pthread_mutex_init(&(mutex_buf[i]), NULL);
  }
  CRYPTO_set_id_callback(id_function);
  CRYPTO_set_locking_callback(locking_function);
  CRYPTO_set_dynlock_create_callback(dyn_create_function);
  CRYPTO_set_dynlock_lock_callback(dyn_lock_function);
  CRYPTO_set_dynlock_destroy_callback(dyn_destroy_function);
  return SOAP_OK;
}

void CRYPTO_thread_cleanup() {
  if( !mutex_buf )
    return;
  CRYPTO_set_id_callback(NULL);
  CRYPTO_set_locking_callback(NULL);
  CRYPTO_set_dynlock_create_callback(NULL);
  CRYPTO_set_dynlock_lock_callback(NULL);
  CRYPTO_set_dynlock_destroy_callback(NULL);
  for(int i = 0; i < CRYPTO_num_locks(); i++) {
    pthread_mutex_destroy(&(mutex_buf[i]));
  }
  free(mutex_buf);
  mutex_buf = NULL;
}
```

Win32用PThreads実装では、pthead_selfの戻り値が、
整数ではなく構造体として定義されているため、
やむなくWin32 APIを使いました

Borland

Copyright (C) 2006 Borland Software Corporation 本文書の一部または全部の転載を禁止します。

サーバの証明書などを準備

- SSLに使用する証明書 (server.pem) を作成
  - opensslコマンドを使用し、ルート証明書とプライベートキーをPEM形式で格納

```
c:\...>openssl req -x509 -newkey rsa:1024 -out cacert.pem -outform PEM -config <OpenSSL>\apps\openssl.cnf
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.++++++
.....++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:password
Verifying - Enter PEM pass phrase:password
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:JP
State or Province Name (full name) [Some-State]:Tokyo
Locality Name (eg, city) []:Shinjuku
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Borland
Organizational Unit Name (eg, section) []:DevCo
Common Name (eg, YOUR name) []:localhost
Email Address []:ttakahashi@borland.com

c:\...>cat cacert.pem privkey.pem > server.pem
```


サーバの証明書などを準備 (続き)

- SSLの鍵交換アルゴリズムに使用するパラメータ (dh512.pem) を作成する

```
c:\...>openssl dhparam -2 512 -out dh512.pem
Loading 'screen' into random state - done
Generating DH parameters, 512 bit long safe prime, generator 2
This is going to take a long time
.....+.....+....+.....................................................
.....++*++*++*++*++*++*++*
```

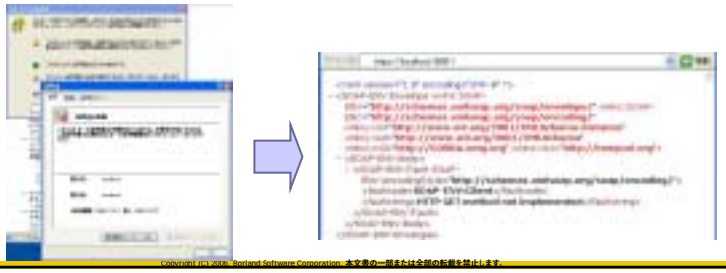
- gSOAPの soap_ssl_server_context 関数の引数に、“dh512.pem” ではなく NULL を渡せば、鍵交換アルゴリズムに DHE ではなく RSA が使用されます

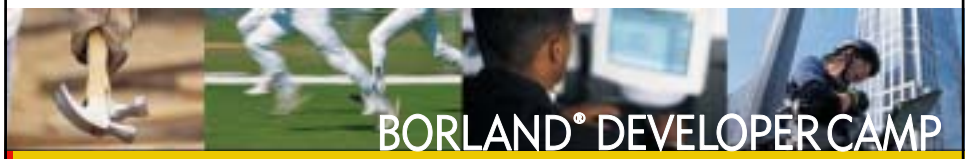
SOAPサーバを実行する

- プロジェクトに、「WITH_OPENSSL」の定義を追加する
- プロジェクトのインクルードディレクトリに「<OpenSSL>¥inc32」を追加する
- プロジェクトに、OpenSSLのライブラリを追加してビルドする
  - libeay32.lib
  - ssleay32.lib
- 便宜上、SOAPサーバの実行ファイル（.exe）と同じフォルダに次のファイルを置く
  - server.pem ― サーバ証明書＋プライベートキー
  - dh512.pem ― 鍵交換用DHEパラメータ

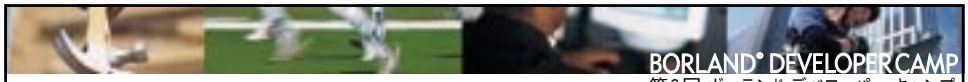
SOAPサーバを実行する（続き）

- マルチスレッド & SSL対応版のSOAPサーバ（CPPServer.exe）を起動する
  - Cランタイム（cc3270mt.dll）が必要
  - Pthreads-Win32のDLL（pthreadBC2.dll）も必要
  - OpenSSLのモジュールは、実行ファイル（.EXE）に含まれています
- IE等のWebブラウザで、とりあえずWebサービスにアクセスしてみる
  - アクセスするURLの例：**https://localhost:18081/**
  - 信頼されない証明書であることを示す警告が出ますが…
  - エラーを示すSOAPレスポンスが返るが、SOAPサーバへのアクセスは成功！





SSL（OpenSSL）を使ったSOAPクライアントへ



SOAPクライアントへの変更

main関数部

```
...
int main(int argc, char* argv[])
{
  soap_ssl_init();                          } OpenSSL関連の初期化
  CRYPTO_thread_setup();

  EmployeeManagerSoapBinding* service = new EmployeeManagerSoapBinding();
  service->endpoint = "https://localhost:18081/";
  if( soap_ssl_client_context(service->soap,
                              SOAP_SSL_REQUIRE_SERVER_AUTHENTICATION,
                              NULL,
                              NULL,                  } SSLクライアントのセットアップ
                              "cacert.pem", // 信頼できるルート証明書
                              NULL,
                              NULL))
  {
    soap_print_fault(service->soap, stderr);
    return -1;
  }
  int id = 99;
  ns1__findByPrimaryKeyResponse resp;
  int soap_ret = service->ns1__findByPrimaryKey(id, resp);
  if( soap_ret == SOAP_OK )
  {
    cout << resp._findByPrimaryKeyReturn->id << endl;
  }
  delete service;
  CRYPTO_thread_cleanup();  <-- OpenSSL関連のクリーンアップ
  return 0;
}
```

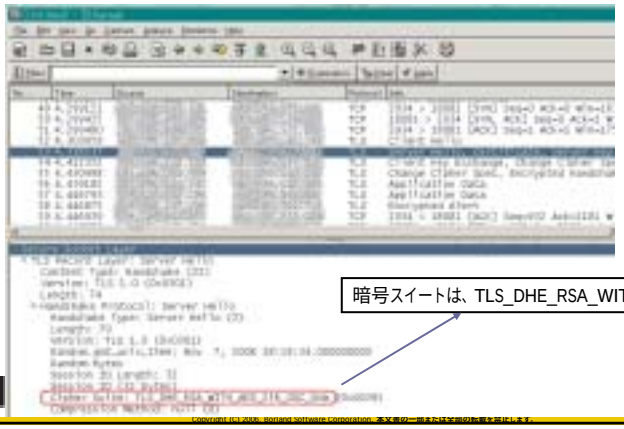
SOAPクライアントへの変更 (続き)

- OpenSSL関連のコード (初期化、ロック、クリーンアップなど) は、SOAPサーバとまったく同じです
- クライアントはアクセスするWebサービスのURLをSSL用に変更する必要があります
  - 例: service->endpoint = "https://localhost:18081/";
- プロジェクトに、「WITH_OPENSSL」の定義を追加する
- プロジェクトのインクルードディレクトリに「<OpenSSL>¥inc32」を追加する
- プロジェクトに、OpenSSLのライブラリを追加してビルドする
  - libeay32.lib
  - ssleay32.lib
- SOAPサーバを信頼するために、信頼できるルート証明書を使用します
  - SOAPサーバセットアップ時に生成した cacert.pem を使用します
  - 便宜上、SOAPクライアントの実行ファイル (.exe) と同じフォルダに次のファイルを置く

SOAPクライアントの起動

- OpenSSLのモジュールは、実行ファイル (.EXE) に含まれています

- パケットモニタ (Ethereal) で、SSL通信を覗いてみました



gSOAP、OpenSSL 利用時の注意点（その1）

- gSOAPのsoapcpp2コマンドが、スタブ/スケルトン用のファイルとして生成する soapC.cpp では、そのままではコンパイルエラーが発生する場合がある
  - Webサービスのメソッド名と同一名の構造体が混在しているため、ボーランドのC++コンパイラが混乱するようです
  - 修正例その1
    - 変更前：{ cp->ptr = (void*)new struct ns1_findByPrimaryKey;
    - 変更後：{ cp->ptr = (void*)new (struct ns1_findByPrimaryKey);
  - 修正例その2
    - 変更前：{ cp->ptr = (void*)new struct ns1_findByPrimaryKey[n];
    - 変更後：{ cp->ptr = (void*)new (struct ns1_findByPrimaryKey[n]);

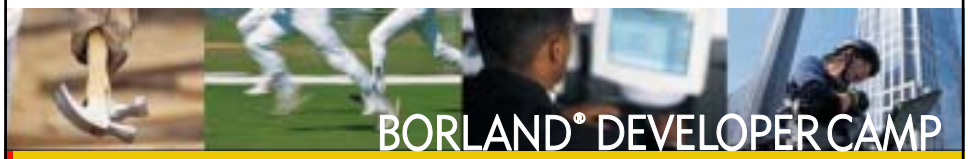
gSOAP、OpenSSL 利用時の注意点（その2）

- gSOAPに付属している stdsoap2.cpp は、最近のバージョンの OpenSSL とともに利用すると、コンパイルエラーが発生する場合がある
  - 最近のOpenSSLの一部のメソッドで、引数の型が変更されているためです
  - 修正前：

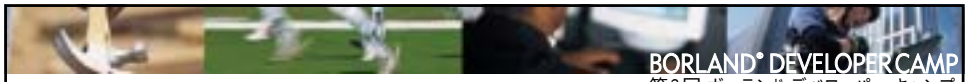
```
#if (OPENSSL_VERSION_NUMBER > 0x00907000L)
  ...
  ext_data = ASN1_item_d2i(NULL, &data, ext->value->length, ASN1_ITEM_ptr(meth->it));
  ...
  ext_data = meth->d2i(NULL, &data, ext->value->length);
  ...
#else
```

  - 修正後：

```
#if (OPENSSL_VERSION_NUMBER > 0x00907000L)
  ...
  ext_data = ASN1_item_d2i(NULL, (const unsigned char**)&data, ext->value->length,
   ASN1_ITEM_ptr(meth->it));
  ...
  ext_data = meth->d2i(NULL, (const unsigned char**)&data, ext->value->length);
  ...
#else
```



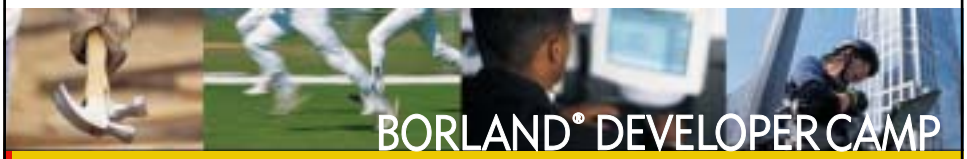
DelphiのVCL製SOAPクライアントから gSOAPサーバ（SSL）にアクセスする



デモ

- SOAPサーバの証明書が自己署名の場合には、Delphi（VCL）のSOAPコンポーネントからのアクセスは失敗する
  - QualityCentralの#10823を参照
    - http://qc.borland.com/wc/qcmain.aspx?d=10823
- IEを利用して、信頼されるルート証明機関として cacert.pem をインポートすればOK

Borland



Thank you

Borland