# [VS2013 编译运行 OTL 示例代码](http://blog.csdn.net/sunyeyi/article/details/44457543)

# 摘要

本文详细记录了，使用VS2013，VC++ Windows Console 程序运行 OTL 一个示例代码 Example 185 （<http://otl.sourceforge.net/otl4_ex185.htm>）的过程，希望能帮到首次接触OTL的朋友。

# 1. OTL简介

OTL 是 Oracle, Odbc and DB2-CLI Template Library 的缩写，是一个C++操控关系数据库的模板库，它目前几乎支持所有的当前各种主流数据库，例如Oracle, MS SQL Server, Sybase, Informix, MySQL, DB2, Interbase / Firebird, PostgreSQL, SQLite, SAP/DB, TimesTen, MS ACCESS等等。OTL中直接操作Oracle主要是通过Oracle提供的OCI接口进行，进行操作DB2数据库则是通过CLI接口来进行，至于MS的数据库和其它一些数据库，则OTL只提供了ODBC来操作的方式。当然Oracle和DB2也可以由OTL间接使用ODBC的方式来进行操纵。OTL最新版本为4.0，项目主页：[http://otl.sourceforge.net/](http://otl.sourceforge.net/" \t "_blank)

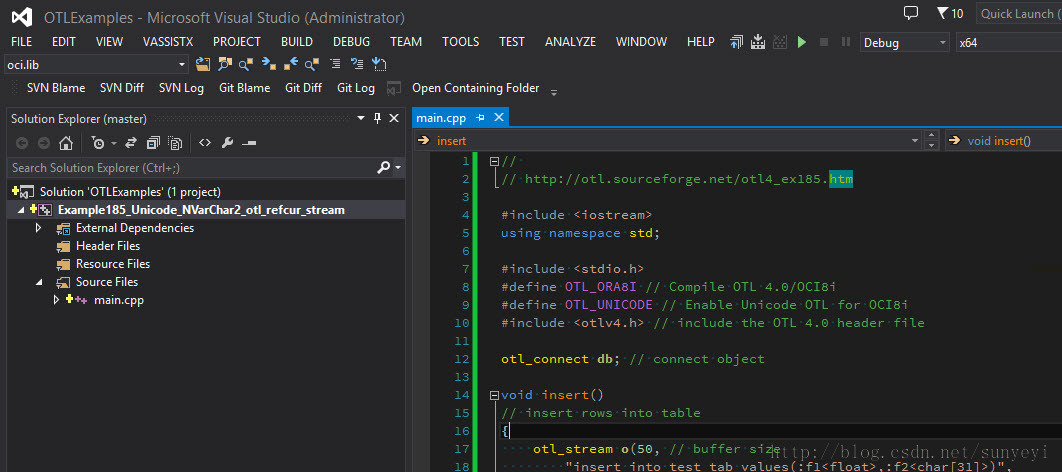
优点：  
      a. 跨平台  
      b. 运行效率高，与C语言直接调用API相当  
      c. 开发效率高，起码比ADO.net使用起来更简单，更简洁  
      d. 部署容易，不需要ADO组件，不需要.net framework 等  
缺点：  
      我不知道，所以我不说:)

# 2. OTL下载

从主页：http://otl.sourceforge.net/可以进入下载页面，使用OTL开发，只需要下载其一个头文件就可以了，下载下来的头文件名为：otlv4.h。在下载页面也同时提供了帮助文档和例子程序的下载。

# 3. 编译调试运行示例 Example 185

3.1 打开VS2013，新建一个VC++空白的Win32 Console Application工程，Solution name：OTL Example，Project Name： Example185\_Unicode\_NVarChar2\_otl\_refcur\_stream，添加cpp文件，main.cpp， 将示例代码（[http://otl.sourceforge.net/otl4\_ex185.htm](http://otl.sourceforge.net/otl4_ex185.htm" \t "_blank)）拷贝到main.cpp中，就向这样

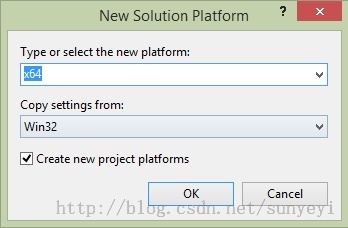


别着急编译，一大堆错误呢，接着往下看。

3.2 配置x64 Solution Platform

系统环境是 Windows 8.1 Pro，64-bitOperating System, x64-based processor。OracleClient 64-bit, Release 11.2.0.3.0。

所以必须，新建x64 SolutionPlatform



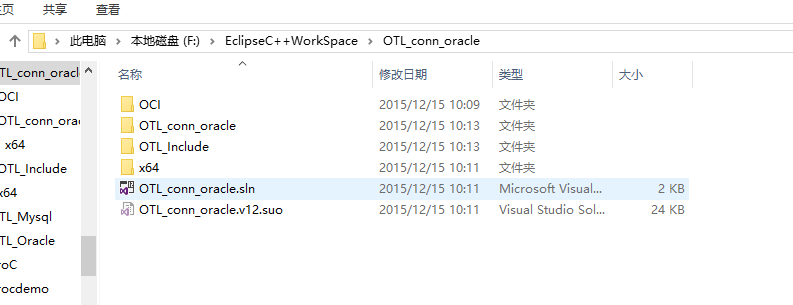
点击OK，然后将编译平台设置为x64

3.3 添加引用otlv4.h

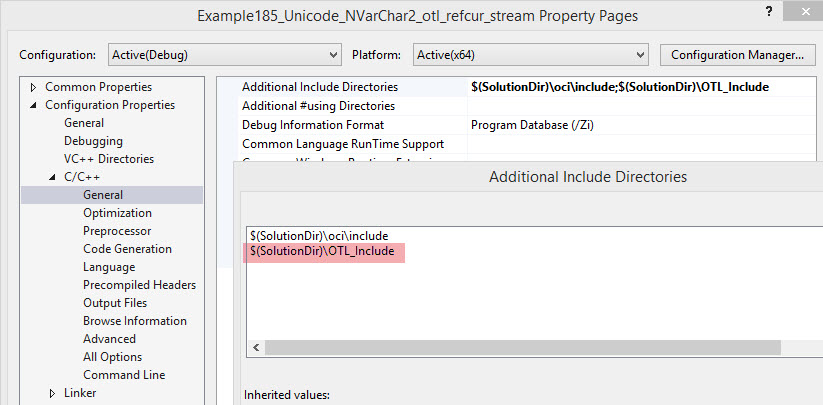
在解决方案文件所在的目录下，新建文件夹OTL\_Include，将下载下来的otlv4.h文件放入其中，然后在工程属性中，添加刚才新建的目录作为otlv4.h引用路径：

注意：

解决方案文件所在的目录为：

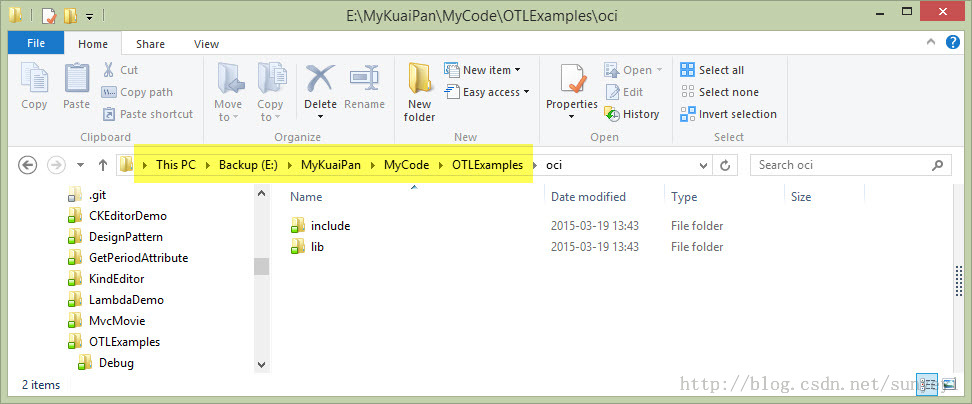


 $(SolutionDir)\OTL\_Include



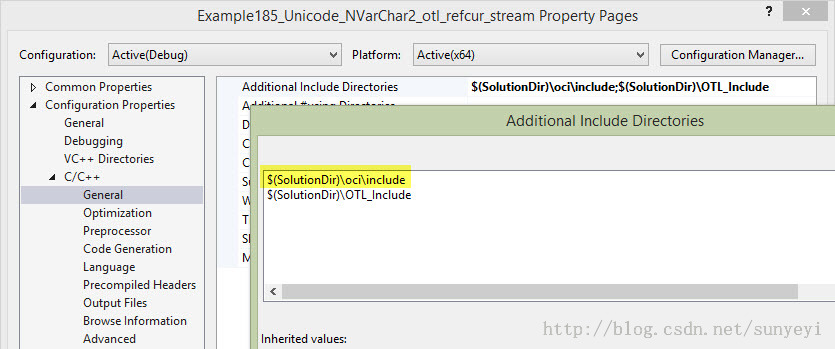
3.4 添加引用oci.h和oci.lib

从Oracle的安装路径中，找到oci.h和oci.lib所在的文件夹，将文件夹oci其拷贝到解决方案所在文件夹下。

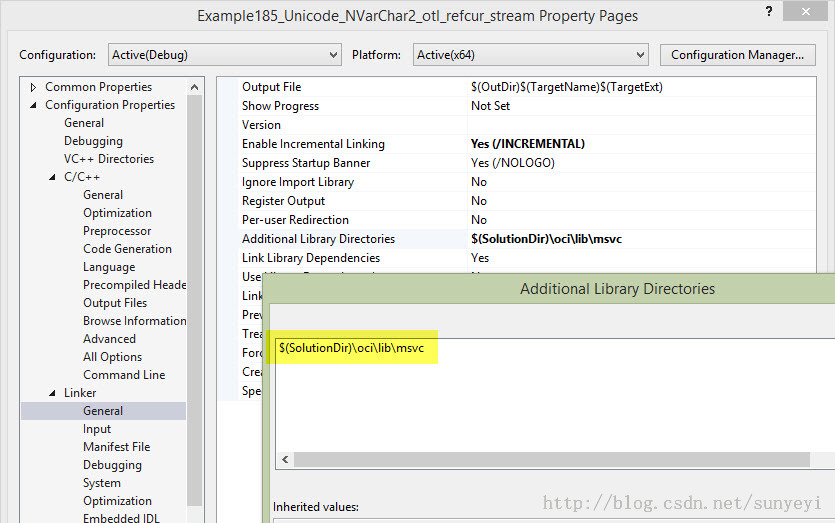


然后依次修改下面3个工程属性：

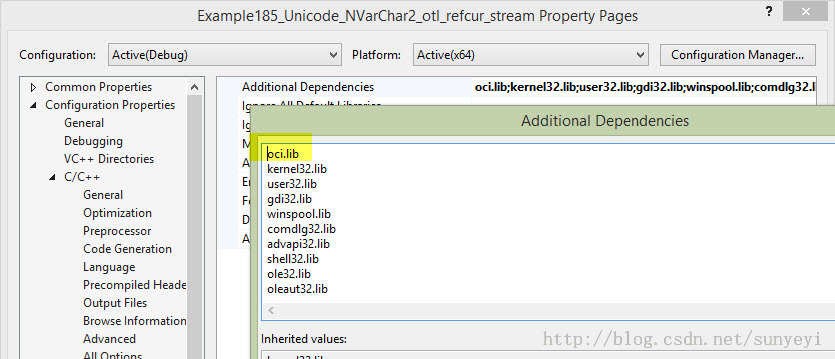
1)  oci.h引用路径：$(SolutionDir)\oci\include



2) Oci.lib引用路径：$(SolutionDir)\oci\lib\msvc



3) 指定oci.lib依赖



# 4 其它编译错误

当遇到以下编译错误时：

1>main.cpp(26): error C4996: 'sprintf': This function orvariable may be unsafe. Consider using sprintf\_s instead. To disabledeprecation, use \_CRT\_SECURE\_NO\_WARNINGS. See online help for details.

1>         C:\Program Files (x86)\Microsoft Visual Studio12.0\VC\include\stdio.h(356) : see declaration of 'sprintf'

使用sprintf\_s替换掉函数sprintf。

至此，编译应该通过了。

# 5. 修改连接字符串

从例子代码中，找到rlogon函数调用，修改其连接到测试数据库，例如，我的就修改如下：

       db.rlogon("fmuser\_syy/fm@10.86.0.7:1521/fmorcl"); // connectto Oracle

# 6. 运行效果



# 7. 示例工程下载

<http://download.csdn.net/detail/sunyeyi/8515193>

资源分分很贵的！所以附上完整源代码一份，你也可以不必去下载！

#include <iostream>

using namespace std;

#include <stdio.h>

#include <stdlib.h>

#define OTL\_ORA8I // Compile OTL 4.0/OCI8i

#define OTL\_UNICODE // Enable Unicode OTL for OCI8i

#include "otlv4.h" // include the OTL 4.0 header file

otl\_connect db; // connect object

void insert()

// insert rows into table

{

otl\_stream o(50, // buffer size

"insert into test\_tab values(:f1<float>,:f2<char[31]>)",

// SQL statement

db // connect object

);

char tmp[32];

unsigned short tmp2[32]; // Null terminated Unicode character array.

for (int i = 1; i <= 100; ++i){

sprintf\_s(tmp, "Name%d", i);

unsigned short\* c2 = tmp2;

char\* c1 = tmp;

// Unicode's first 128 characters are ASCII (0..127), so

// all is needed for converting ASCII into Unicode is as follows:

while (\*c1){

\*c2 = (unsigned char)\*c1;

++c1; ++c2;

}

\*c2 = 0; // target Unicode string is null terminated,

// only the null terminator is a two-byte character,

// not one-byte

o << (float)i;

o << (unsigned char\*)tmp2;

// overloaded operator<<(const unsigned char\*) in the case of Unicode

// OTL accepts a pointer to a Unicode character array.

// operator<<(const unsigned short\*) wasn't overloaded

// in order to avoid ambiguity in C++ type casting.

}

}

void select()

{

otl\_stream i(1, // buffer size

"begin "

" open :cur<refcur,out[50]> for "

// :cur is a bind variable name, refcur -- its type,

// out -- output parameter, 50 -- the buffer size when this

// reference cursor will be attached to otl\_refcur\_stream

" select \* "

" from test\_tab "

" where f1>=:f<int,in> and f1<=:f\*2; "

"end;", // PL/SQL block returns a referenced cursor

db // connect object

);

// create select stream with referenced cursor

i.set\_commit(0); // set stream "auto-commit" to OFF.

float f1;

unsigned short f2[32];

otl\_refcur\_stream s; // reference cursor stream for reading rows.

i << 8; // assigning :f = 8

i >> s; // initializing the refrence cursor stream with the output

// reference cursor.

while (!s.eof()){ // while not end-of-data

s >> f1;

s >> (unsigned char\*)f2;

// overloaded operator>>(unsigned char\*) in the case of Unicode

// OTL accepts a pointer to a Unicode chracter array.

// operator>>(unsigned short\*) wasn't overloaded

// in order to avoid ambiguity in C++ type casting.

cout << "f1=" << f1 << ", f2=";

// Unicode's first 128 characters are ASCII, so in order

// to convert Unicode back to ASCII all is needed is

// as follows:

for (int j = 0; f2[j] != 0; ++j){

cout << (char)f2[j];

}

cout << endl;

}

s.close(); // closing the reference cursor

i << 4; // assigning :f = 4

i >> s;

while (!s.eof()){ // while not end-of-data

s >> f1;

s >> (unsigned char\*)f2;

// overloaded operator>>(unsigned char\*) in the case of Unicode

// OTL accepts a pointer to a Unicode chracter array.

// operator>>(unsigned short\*) wasn't overloaded

// in order to avoid ambiguity in C++ type casting.

cout << "f1=" << f1 << ", f2=";

// Unicode's first 128 characters are ASCII, so in order

// to convert Unicode back to ASCII all is needed is

// as follows:

for (int j = 0; f2[j] != 0; ++j){

cout << (char)f2[j];

}

cout << endl;

}

// there is no need to explicitly calls s.close() since s's destructor

// will take care of closing the stream

}

int main()

{

otl\_connect::otl\_initialize(); // initialize OCI environment

try{

db.rlogon("scott/tiger@192.168.1.40:1521/orcl"); // connect to Oracle

otl\_cursor::direct\_exec

(

db,

"drop table test\_tab",

otl\_exception::disabled // disable OTL exceptions

); // drop table

otl\_cursor::direct\_exec

(

db,

"create table test\_tab(f1 number, f2 nvarchar2(60))"

); // create table

db.set\_character\_set(SQLCS\_NCHAR);

insert(); // insert records into table

select(); // select records from table

}

catch (otl\_exception& p){ // intercept OTL exceptions

cerr << p.msg << endl; // print out error message

cerr << p.stm\_text << endl; // print out SQL that caused the error

cerr << p.var\_info << endl; // print out the variable that caused the error

}

db.logoff(); // disconnect from Oracle

system("pause");

return 0;

}