## 目录

[目录 1](#_Toc490821864)

[一、工具准备 2](#_Toc490821865)

[二、编译C++接口 2](#_Toc490821866)

[三、实现浏览器 5](#_Toc490821867)

[浏览器定制 5](#_Toc490821868)

[浏览器默认最大化，标题从配置文件读取 5](#_Toc490821869)

[设置浏览器中文环境，开启npapi插件功能，并注册身份证阅读器插件（见第四节） 7](#_Toc490821870)

[文件下载 8](#_Toc490821871)

[首先浏览器处理类要继承CefDownloadHandler 类 8](#_Toc490821872)

[并提供回调接口（继承任何处理类都要提供类似的回调接口） 8](#_Toc490821873)

[文件下载具体重载函数 8](#_Toc490821874)

[文件下载重载函数实现 9](#_Toc490821875)

[F5刷新功能 10](#_Toc490821876)

[继承按键处理类 10](#_Toc490821877)

[提供回调接口 10](#_Toc490821878)

[重载声明 11](#_Toc490821879)

[重载实现 11](#_Toc490821880)

[右键菜单添加刷新按钮 12](#_Toc490821881)

[继承菜单处理类 12](#_Toc490821882)

[提供回调接口 12](#_Toc490821883)

[重载声明 12](#_Toc490821884)

[重载实现 13](#_Toc490821885)

[四、身份证阅读器插件开发 14](#_Toc490821886)

[工程建立 14](#_Toc490821887)

[首先下载npapi插件开发包，解压进入 14](#_Toc490821888)

[新建win32 dll工程，工程名以NP开头，将上图的文件引入工程。 14](#_Toc490821889)

[新建stdafx.h头文件，加入 15](#_Toc490821890)

[设置预处理 16](#_Toc490821891)

[设置头文件包含 16](#_Toc490821892)

[功能实现 17](#_Toc490821893)

[加载身份证识别模块 17](#_Toc490821894)

[定义功能标识 17](#_Toc490821895)

[实现具体功能 18](#_Toc490821896)

[定义插件模块，其中MIMEType，是在html中调用的关键字 19](#_Toc490821897)

[生成插件 20](#_Toc490821898)

[插件部署 20](#_Toc490821899)

[插件测试 21](#_Toc490821900)

[插件调试 23](#_Toc490821901)

[浏览器增加启动参数 23](#_Toc490821902)

[启动浏览器 24](#_Toc490821903)

[在vs中打好断点，附加到进程 24](#_Toc490821904)

[进入断点 25](#_Toc490821905)

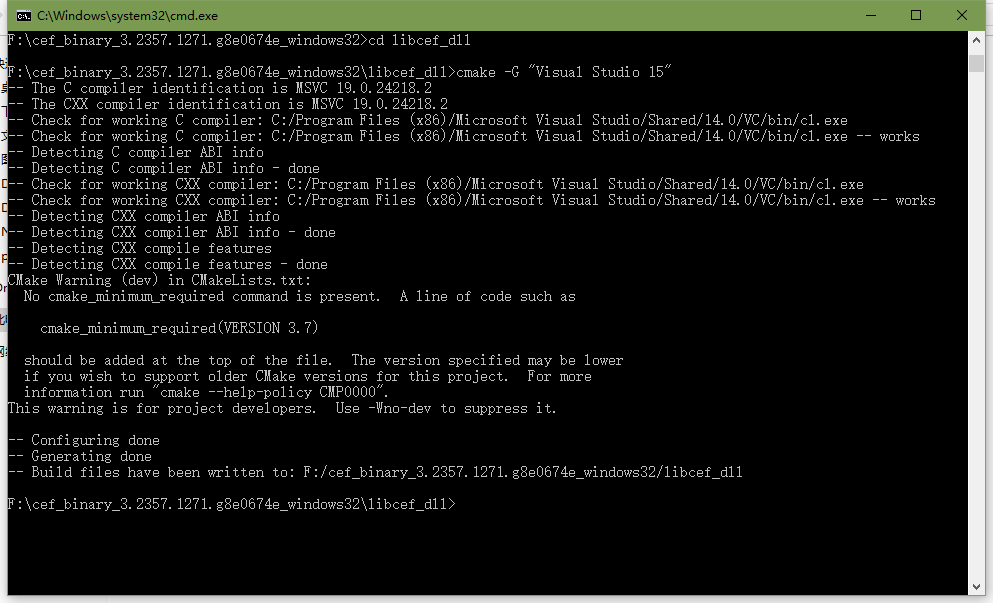
## 一、工具准备

1. vs 2017 community
2. cmake 3.7 +
3. cef\_binary\_3.2357.1271.g8e0674e\_windows32，包含已编译好的c接口动态库和c++接口源码，C++接口默认没有编译，以源码提供

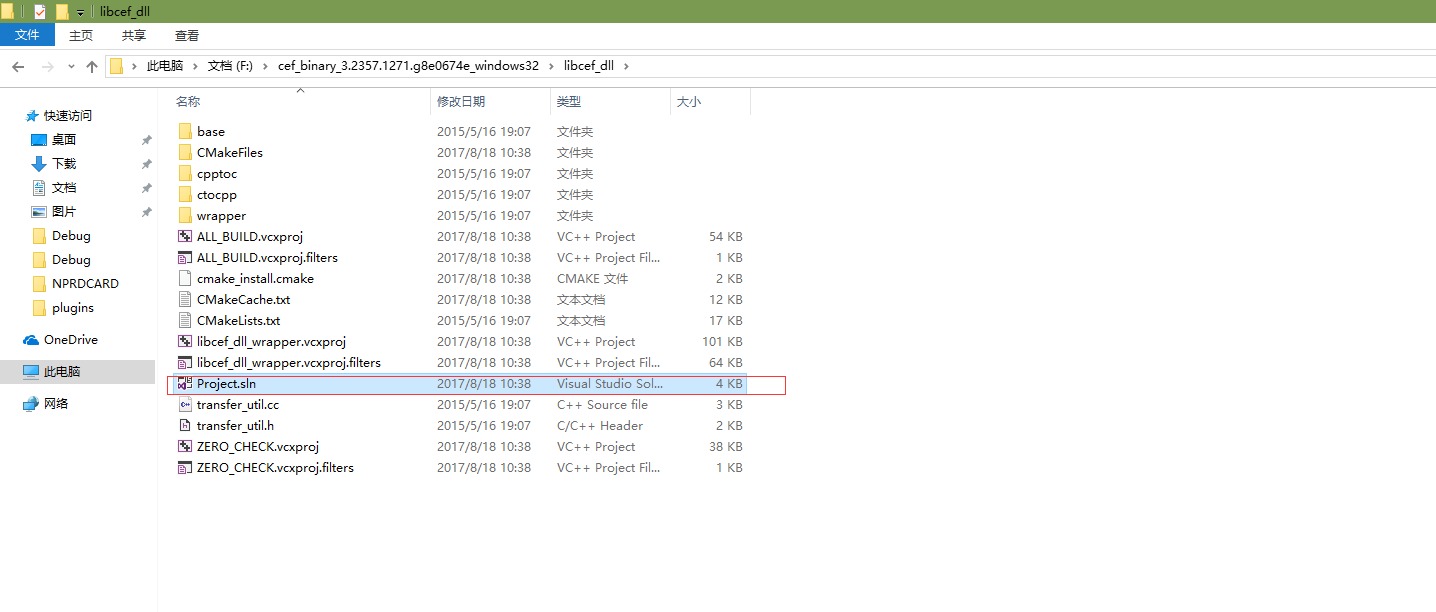
## 二、编译C++接口

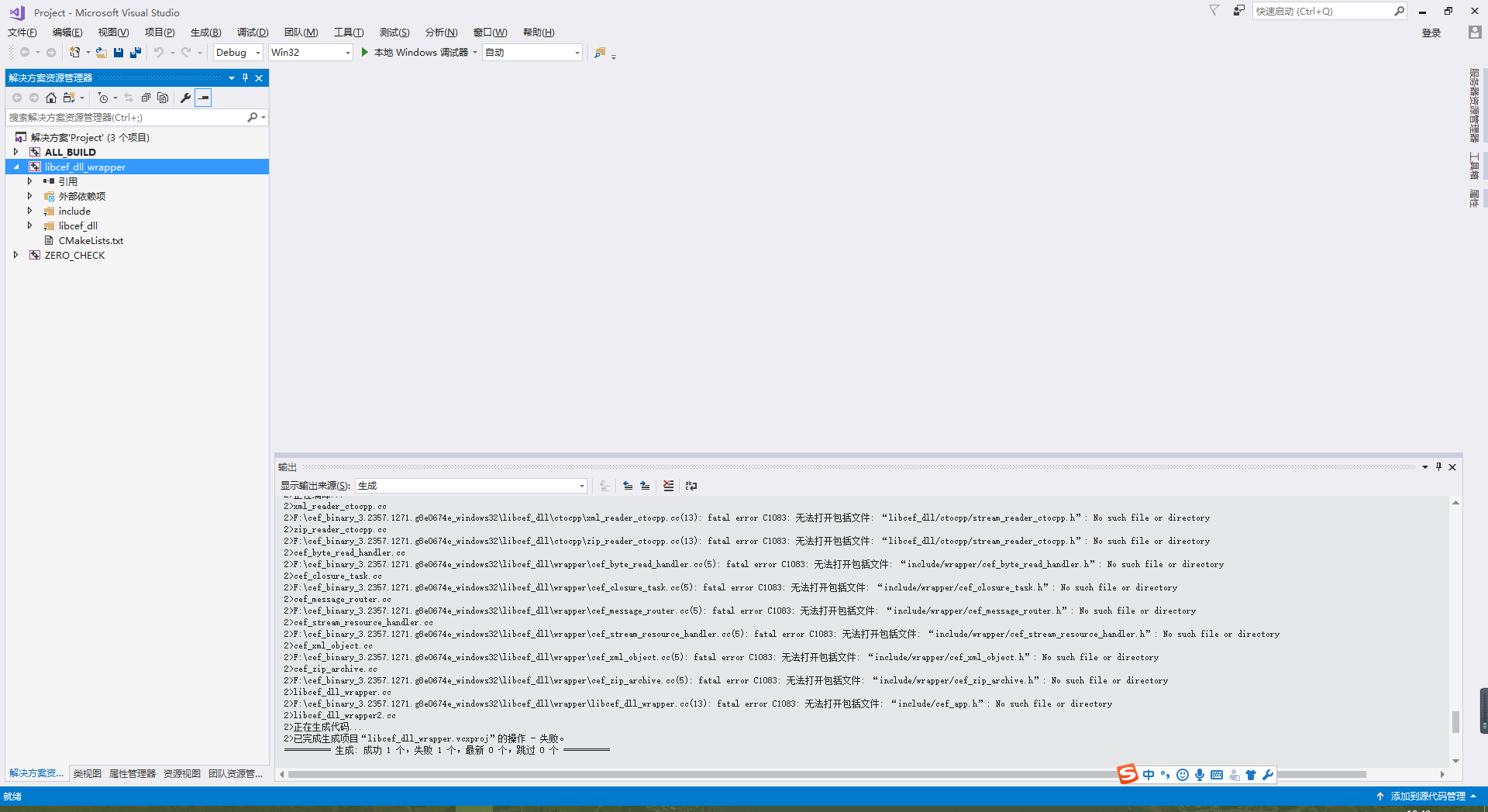
### 生成vs工程

进入cef源码根目录，进入libcef\_dll。此目录是c++接口源码目录。C接口动态库放在根目录的Release目录中。输入命令cmake -G "Visual Studio 15"，生成VS工程

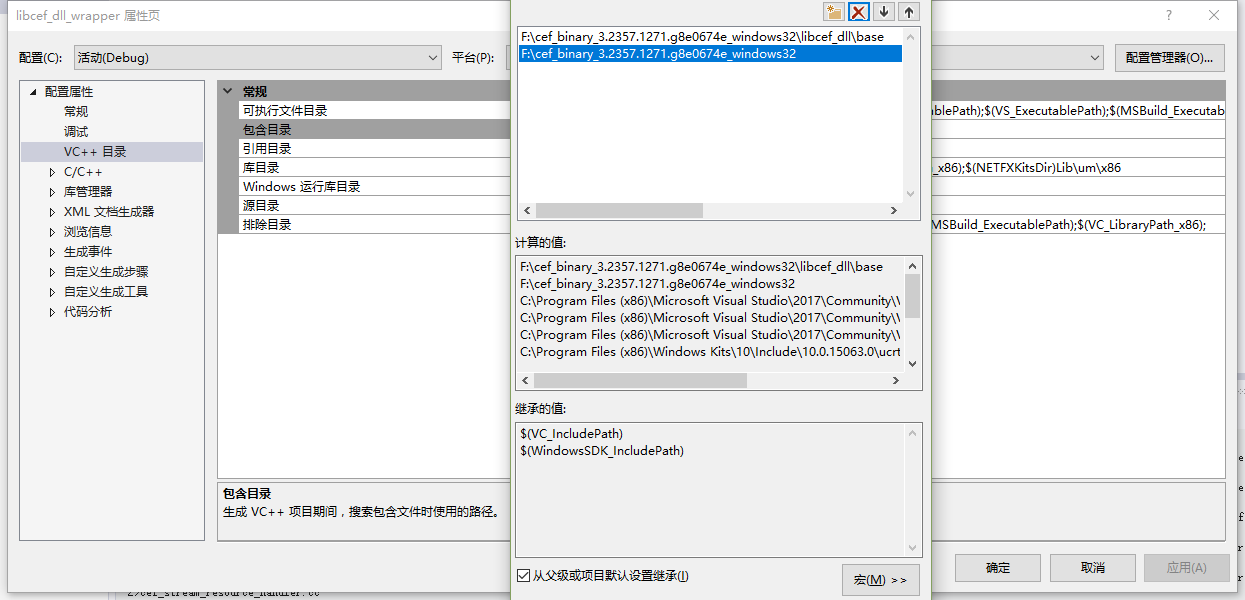


### 打开vs工程

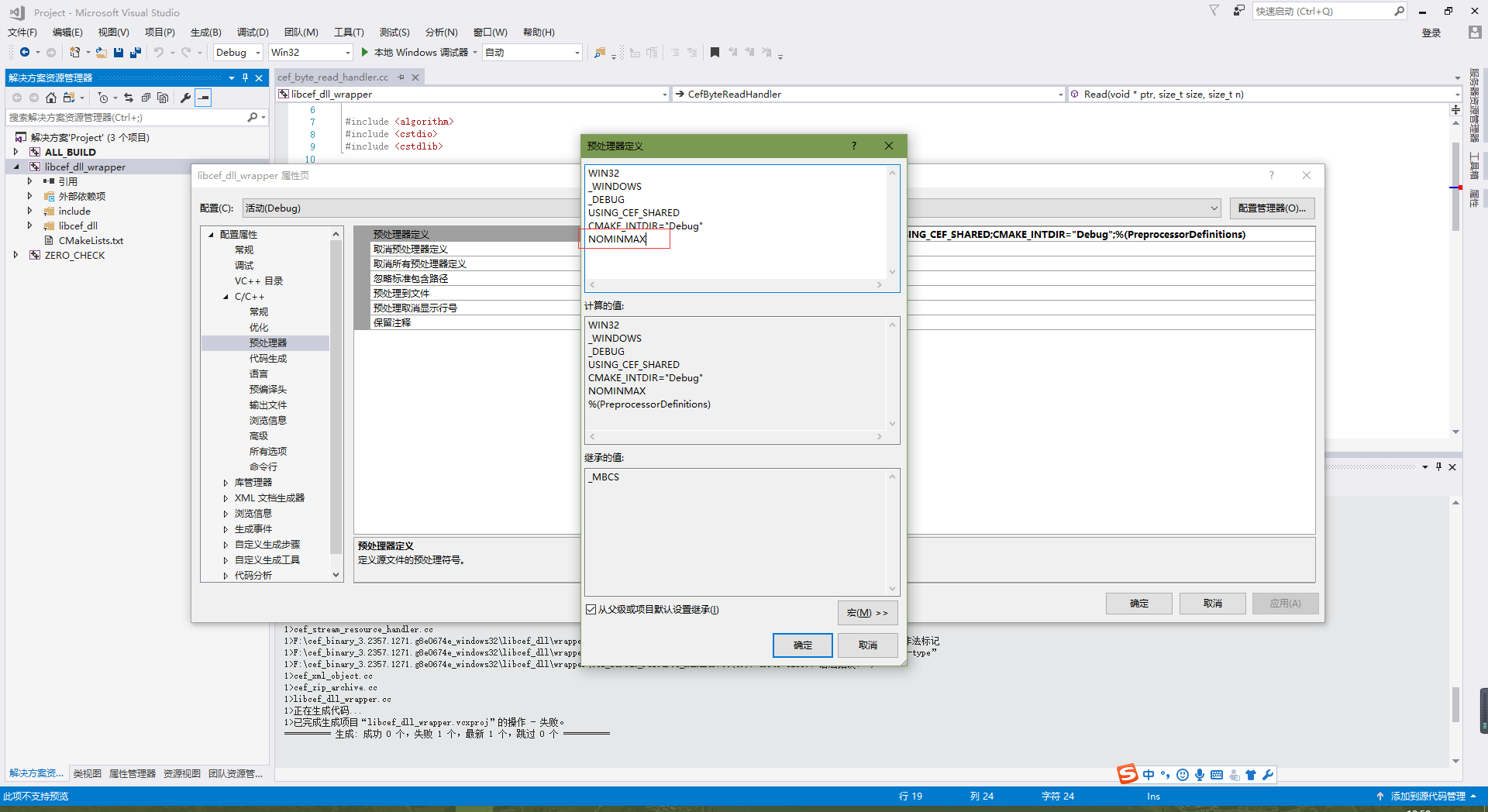




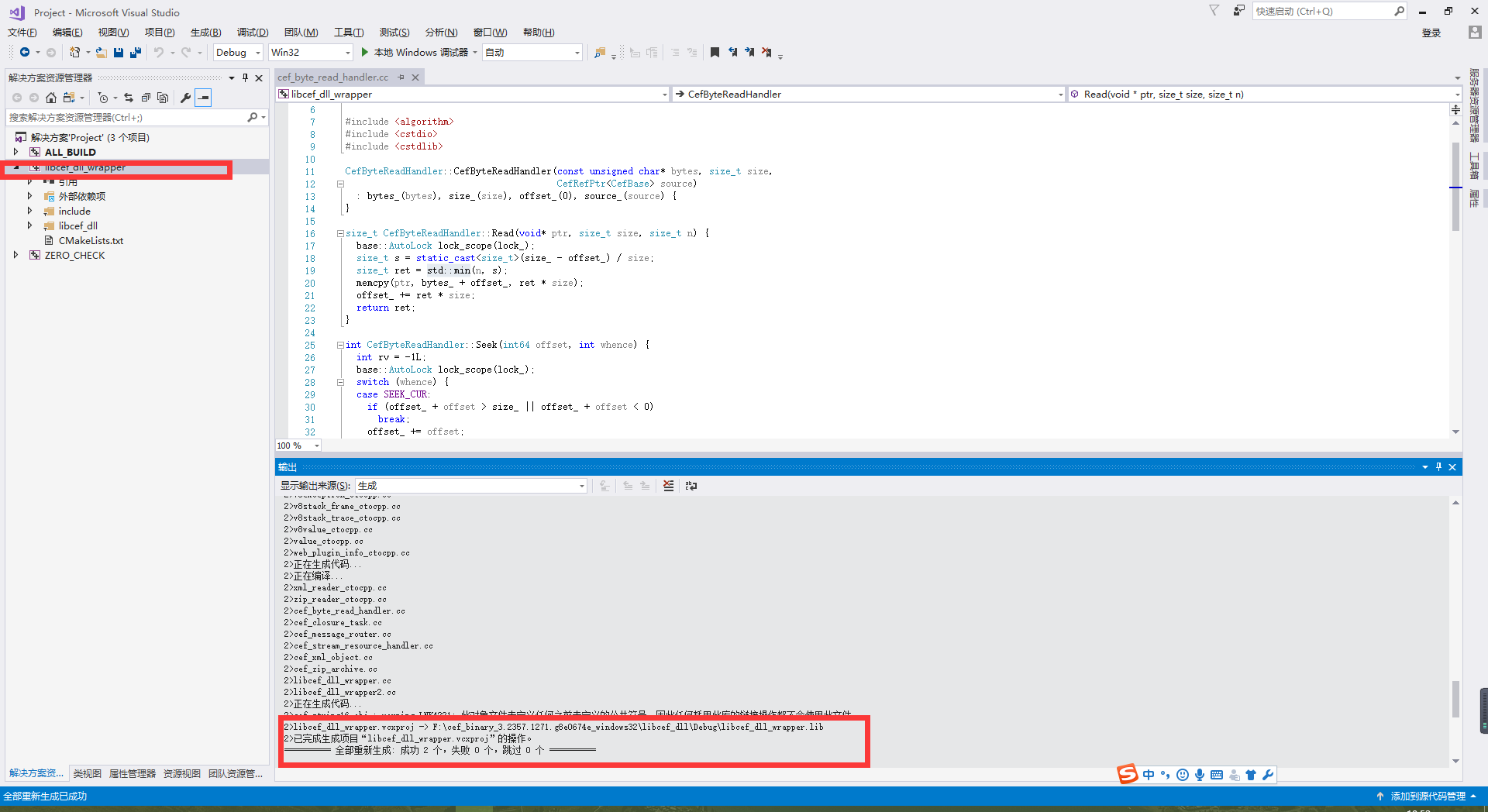
### 右击项目“libcef\_dll\_wrapper”，设置头文件包含目录



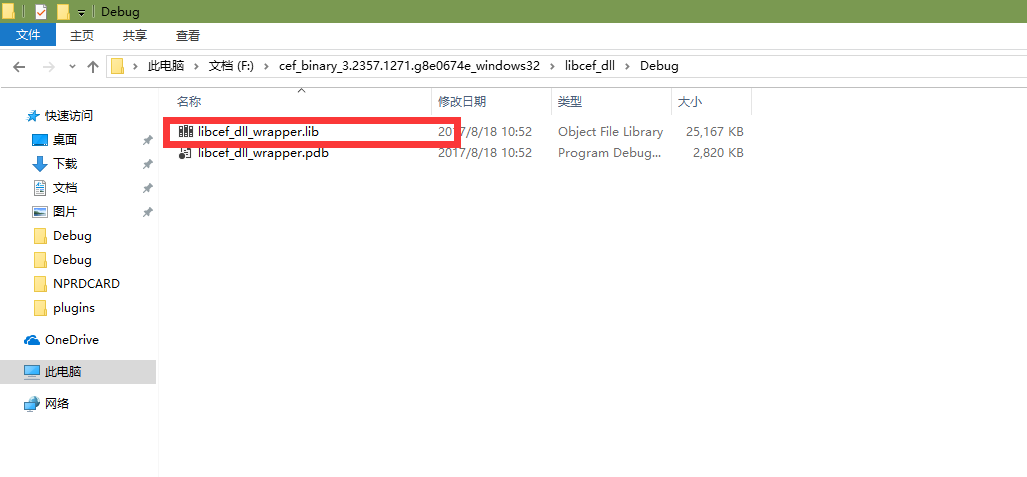
### 加入预处理命令 NOMINMAX



### 生成“libcef\_dll\_wrapper”项目



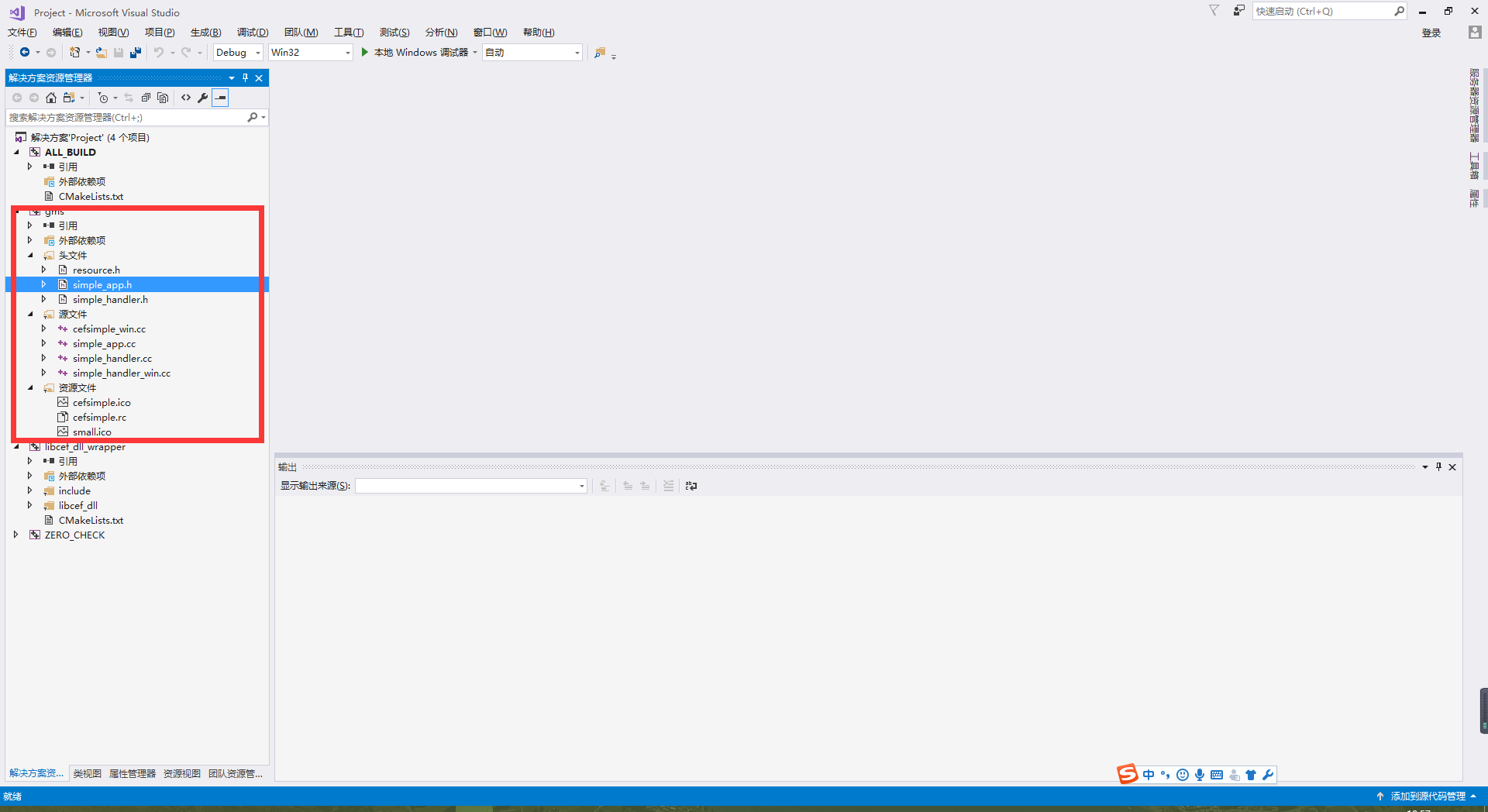
### 生成的c++接口



## 三、实现浏览器

### 浏览器定制

新建一个win32GUI的空工程，将cef目录下的cefsimple相关文件拷贝进来，cefsimple实现很简单就是打开一个网页，我们来新增一些功能



#### 浏览器默认最大化，标题从配置文件读取

void SimpleApp::OnContextInitialized() {

CEF\_REQUIRE\_UI\_THREAD();

// Information used when creating the native window.

CefWindowInfo window\_info;

#if defined(OS\_WIN)

// On Windows we need to specify certain flags that will be passed to

// CreateWindowEx().

window\_info.SetAsPopup(NULL, "cefsimple");

#endif

// 设置窗口样式

window\_info.style = WS\_OVERLAPPEDWINDOW | WS\_VISIBLE | WS\_MAXIMIZE;

UINT width = GetSystemMetrics(SM\_CXSCREEN);

UINT height = GetSystemMetrics(SM\_CYSCREEN);

window\_info.parent\_window = NULL;

window\_info.x = -1;

window\_info.y = -1;

window\_info.width = width;

window\_info.height = height;

// SimpleHandler implements browser-level callbacks.

CefRefPtr<SimpleHandler> handler(new SimpleHandler());

// Specify CEF browser settings here.

CefBrowserSettings browser\_settings;

char \* code = "UTF-8";

cef\_string\_t encode = {};

cef\_string\_utf8\_to\_utf16(code, strlen(code), &encode);

browser\_settings.default\_encoding = encode;

//设置窗口标题

WCHAR window\_name[1024];

GetPrivateProfileStringW(L"GMS", L"title", L"", window\_name, 1024, L".\\gms.ini");

cef\_string\_wide\_to\_utf16(window\_name, wcslen(window\_name), &window\_info.window\_name);

// 设置访问地址

char url[1024];

GetPrivateProfileStringA("GMS", "url", "", url, 1024, ".\\gms.ini");

// Create the first browser window.

CefBrowserHost::CreateBrowser(window\_info, handler.get(), url, browser\_settings, NULL);

}

#### 设置浏览器中文环境，开启npapi插件功能，并注册身份证阅读器插件（见第四节）

void SimpleApp::OnBeforeCommandLineProcessing(const CefString& process\_type, CefRefPtr<CefCommandLine> command\_line) {

//开启插件

command\_line->AppendSwitch("--enable-npapi");

//设置中文语言环境

command\_line->AppendSwitchWithValue("--lang", "zh-CN");

command\_line->AppendSwitchWithValue("register-netscape-plugins","NPRDCARD.dll/nprdcard-plugin");

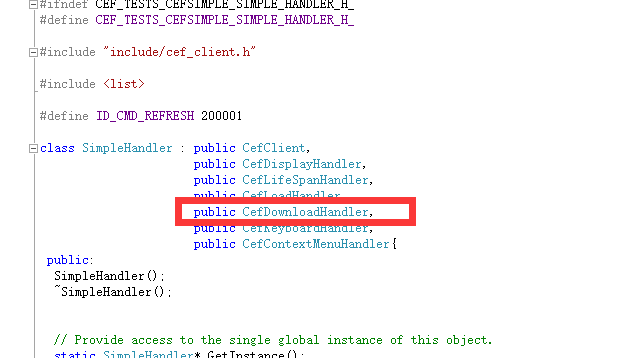
return;

}



### 文件下载

#### 首先浏览器处理类要继承CefDownloadHandler 类



#### 并提供回调接口（继承任何处理类都要提供类似的回调接口）

virtual CefRefPtr<CefDownloadHandler> GetDownloadHandler() //实现

{

return this;

}

#### 文件下载具体重载函数

//重载文件下载方法

virtual void OnBeforeDownload(

CefRefPtr<CefBrowser> browser,

CefRefPtr<CefDownloadItem> download\_item,

const CefString& suggested\_name,

CefRefPtr<CefBeforeDownloadCallback> callback) OVERRIDE;

virtual void OnDownloadUpdated(

CefRefPtr<CefBrowser> browser,

CefRefPtr<CefDownloadItem> download\_item,

CefRefPtr<CefDownloadItemCallback> callback) OVERRIDE;

#### 文件下载重载函数实现

//文件下载方法重载

void SimpleHandler::OnBeforeDownload(

CefRefPtr<CefBrowser> browser,

CefRefPtr<CefDownloadItem> download\_item,

const CefString& suggested\_name,

CefRefPtr<CefBeforeDownloadCallback> callback)

{

callback->Continue(download\_item->GetURL(), true);

}

void SimpleHandler::OnDownloadUpdated(

CefRefPtr<CefBrowser> browser,

CefRefPtr<CefDownloadItem> download\_item,

CefRefPtr<CefDownloadItemCallback> callback)

{

if (download\_item->IsComplete())

{

//MessageBox.Show("下载成功");

if (browser->IsPopup() && !browser->HasDocument())

{

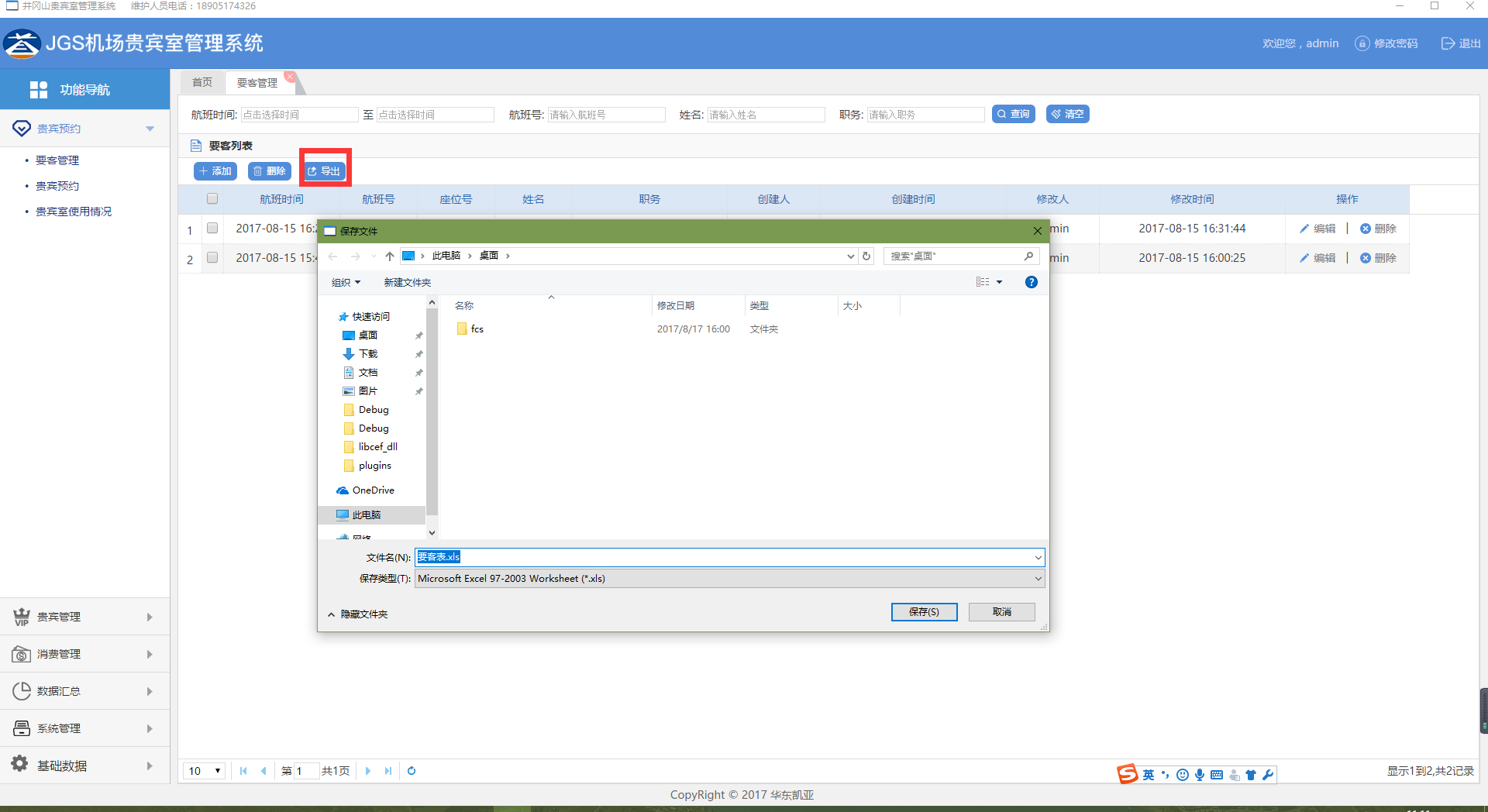
//browser->GetHost()->ParentWindowWillClose();

browser->GetHost()->CloseBrowser(true);

}

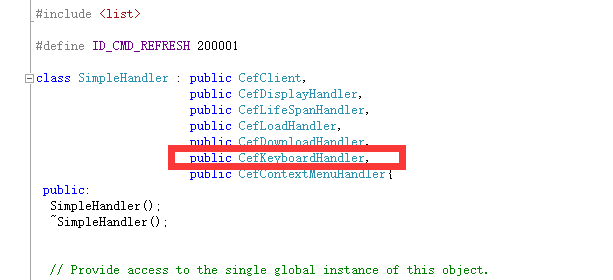
}

}



### F5刷新功能

#### 继承按键处理类



#### 提供回调接口

virtual CefRefPtr<CefKeyboardHandler> GetKeyboardHandler() //实现

{

return this;

}

#### 重载声明

//按键处理

virtual bool OnPreKeyEvent(CefRefPtr<CefBrowser> browser,

const CefKeyEvent& event,

CefEventHandle os\_event,

bool\* is\_keyboard\_shortcut) OVERRIDE;

virtual bool OnKeyEvent(CefRefPtr<CefBrowser> browser,

const CefKeyEvent& event,

CefEventHandle os\_event) OVERRIDE;

#### 重载实现

bool SimpleHandler::OnPreKeyEvent(CefRefPtr<CefBrowser> browser,

const CefKeyEvent& event,

CefEventHandle os\_event,

bool\* is\_keyboard\_shortcut) {

if(event.windows\_key\_code==116)//F5刷新

browser->Reload();

return false;

}

bool SimpleHandler::OnKeyEvent(CefRefPtr<CefBrowser> browser,

const CefKeyEvent& event,

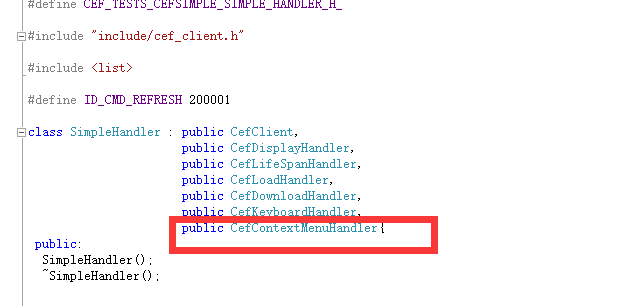
CefEventHandle os\_event) {

return false;

}

### 右键菜单添加刷新按钮

#### 继承菜单处理类



#### 提供回调接口

virtual CefRefPtr<CefContextMenuHandler> GetContextMenuHandler() //实现

{

return this;

}

#### 重载声明

// 右键菜单处理

virtual void OnBeforeContextMenu(CefRefPtr<CefBrowser> browser,

CefRefPtr<CefFrame> frame,

CefRefPtr<CefContextMenuParams> params,

CefRefPtr<CefMenuModel> model) OVERRIDE;

virtual bool OnContextMenuCommand(CefRefPtr<CefBrowser> browser,

CefRefPtr<CefFrame> frame,

CefRefPtr<CefContextMenuParams> params,

int command\_id,

EventFlags event\_flags) OVERRIDE;

#### 重载实现

void SimpleHandler::OnBeforeContextMenu(CefRefPtr<CefBrowser> browser,

CefRefPtr<CefFrame> frame,

CefRefPtr<CefContextMenuParams> params,

CefRefPtr<CefMenuModel> model) {

if (model->GetCount() > 0) {// 刷新菜单

model->AddSeparator();

model->AddItem(ID\_CMD\_REFRESH, \_\_TEXT("刷新"));

}

}

bool SimpleHandler::OnContextMenuCommand(CefRefPtr<CefBrowser> browser,

CefRefPtr<CefFrame> frame,

CefRefPtr<CefContextMenuParams> params,

int command\_id,EventFlags event\_flags) {

switch (command\_id)

{

case ID\_CMD\_REFRESH:

browser->Reload();

break;

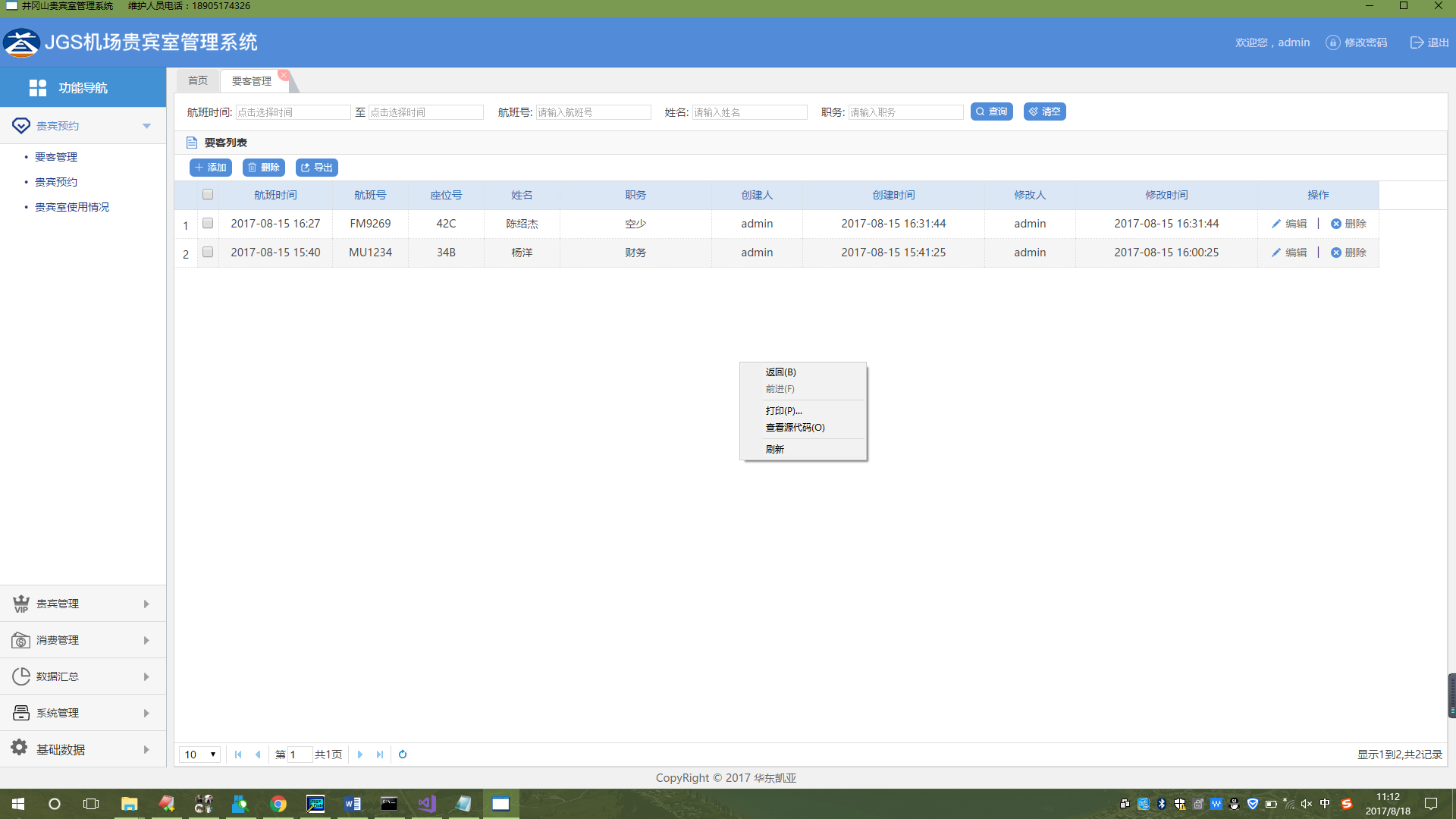
default:

break;

}

return false;

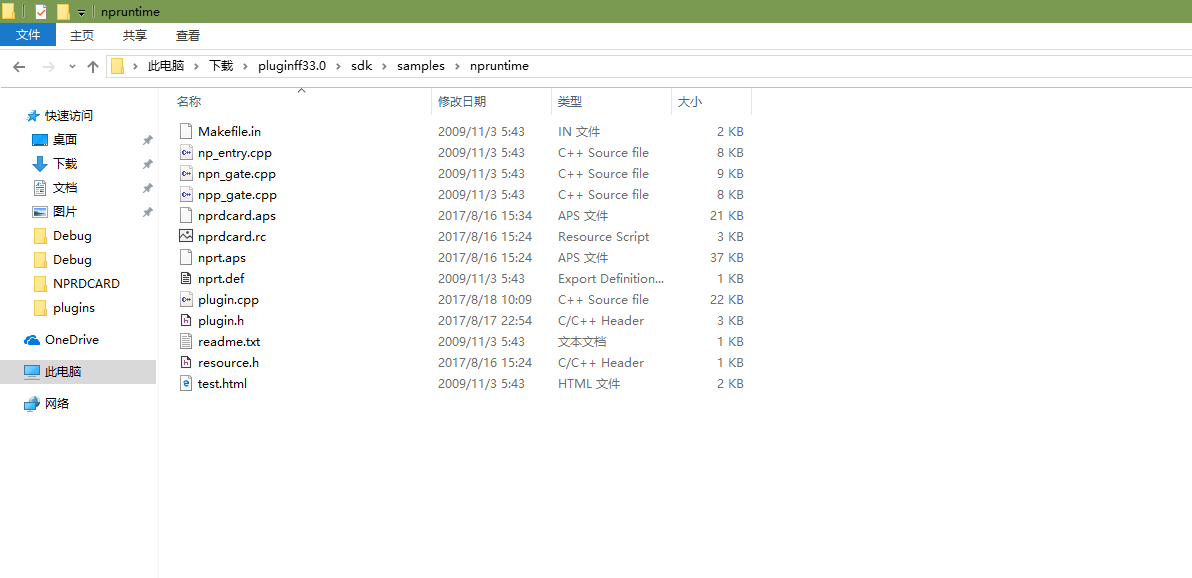
}



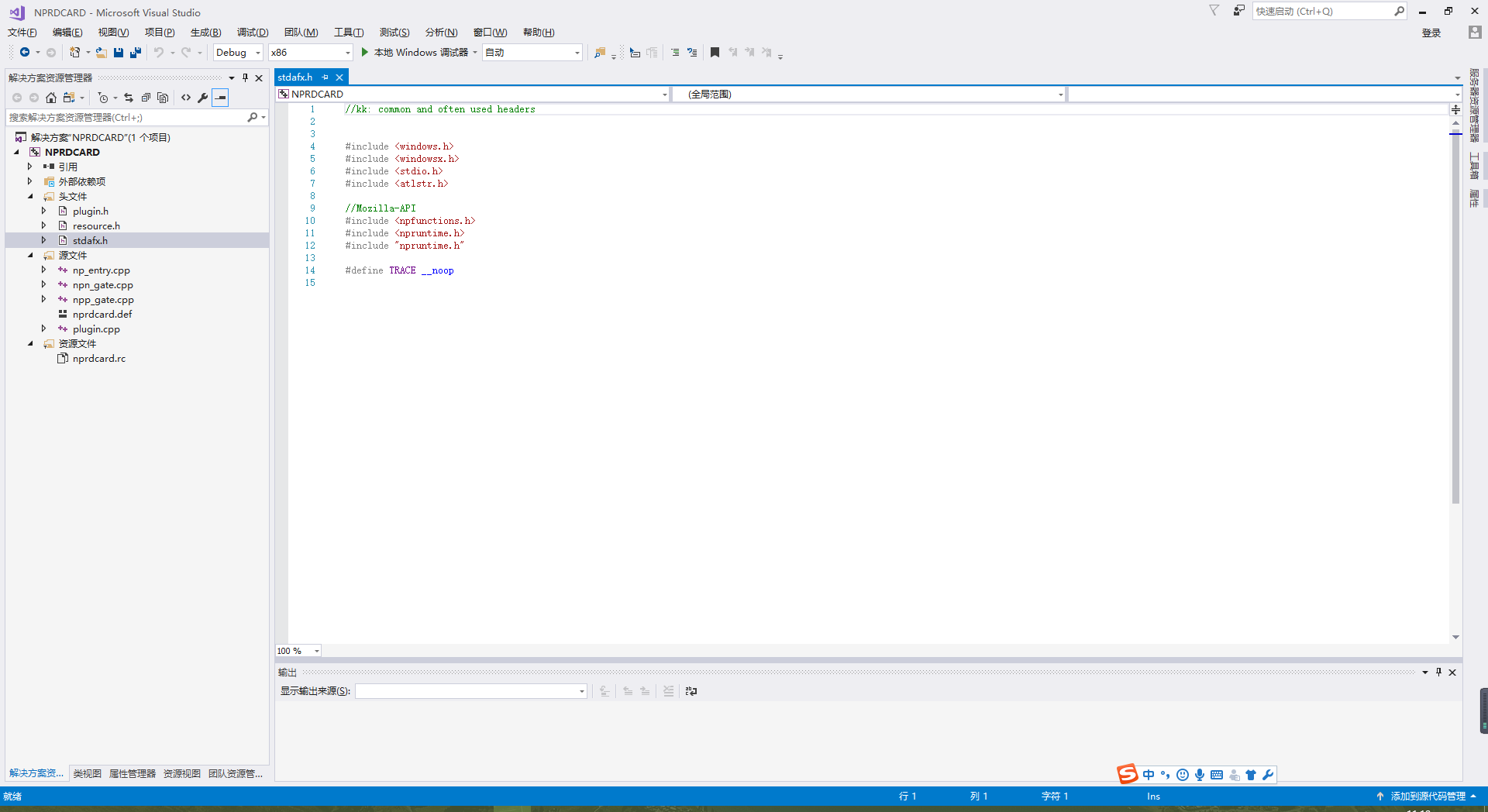
## 四、身份证阅读器插件开发

### 工程建立

#### 首先下载npapi插件开发包，解压进入



#### 新建win32 dll工程，工程名以NP开头，将上图的文件引入工程。



#### 新建stdafx.h头文件，加入

//kk: common and often used headers

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <atlstr.h>

//Mozilla-API

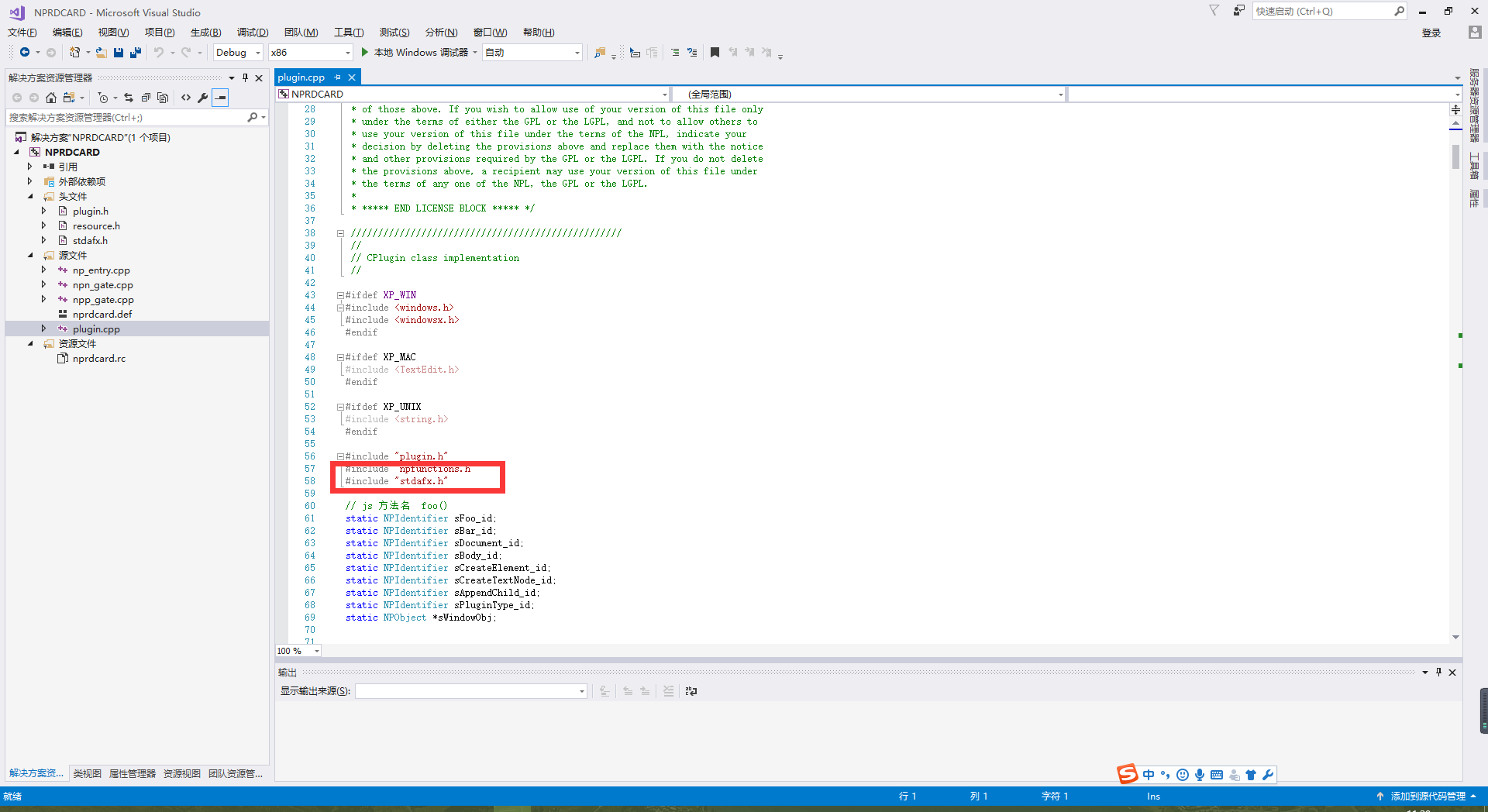
#include <npfunctions.h>

#include <npruntime.h>

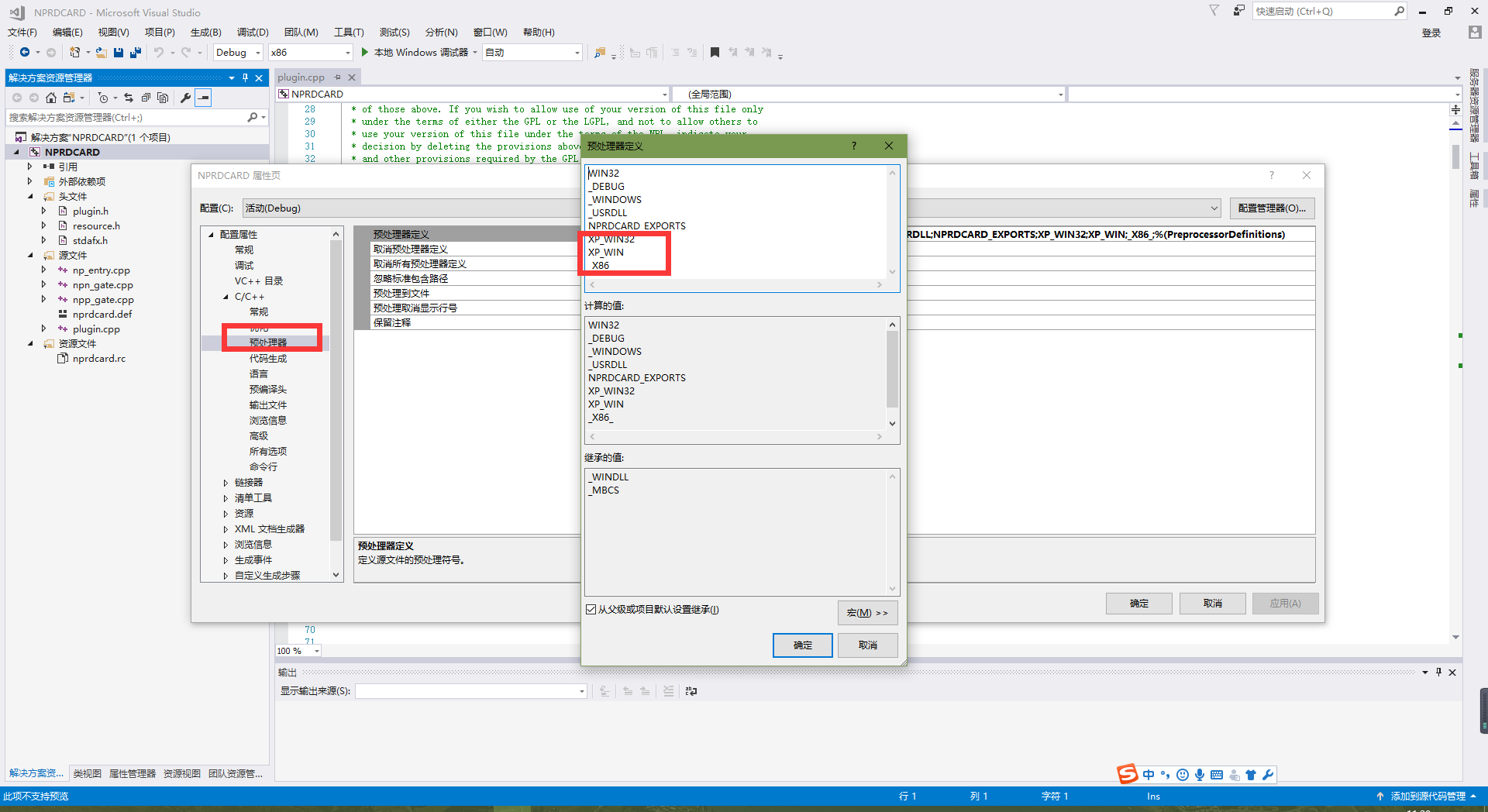
#include "npruntime.h"

#define TRACE \_\_noop

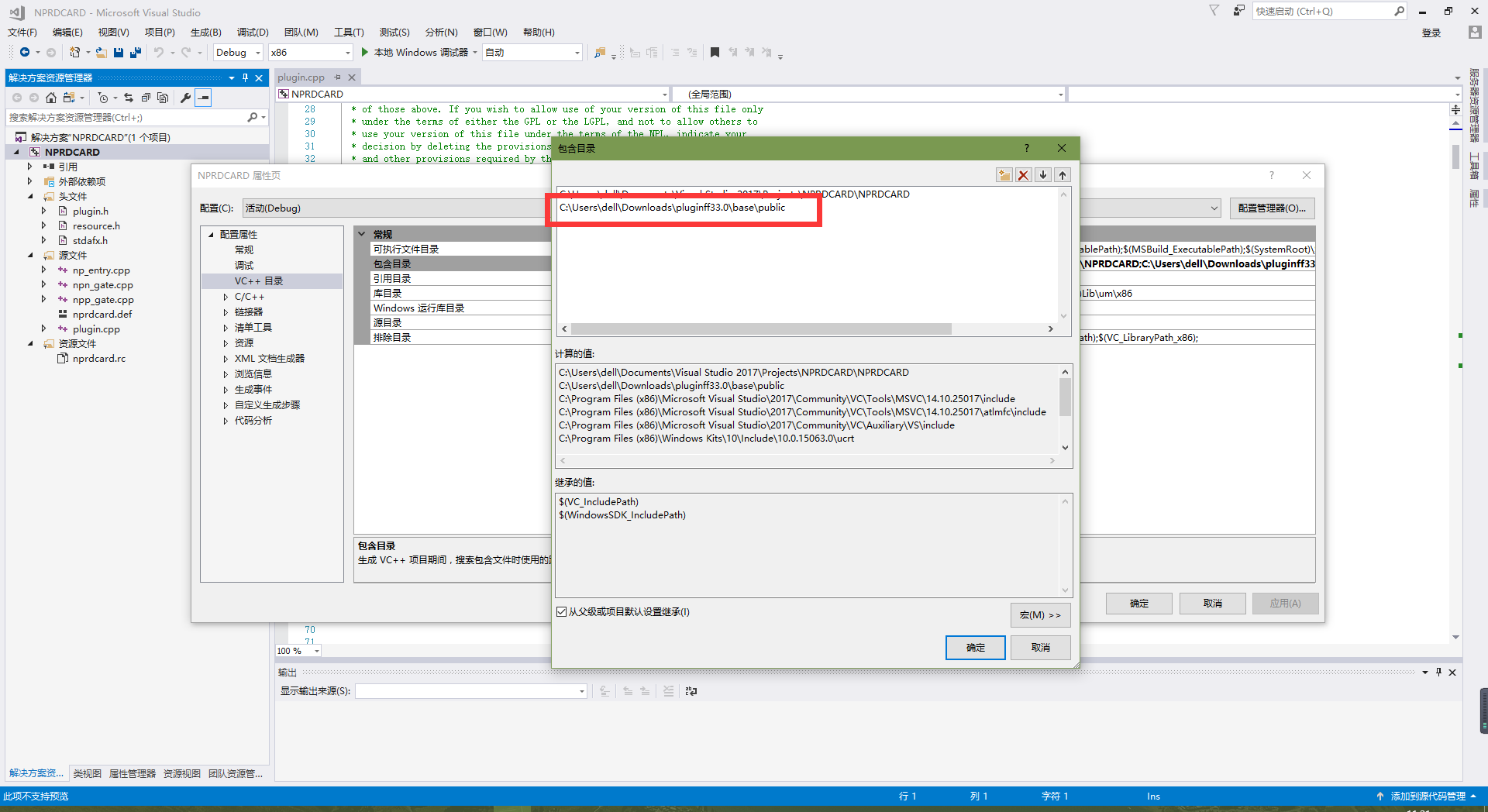
并在plugin.cpp中引用此头文件



#### 设置预处理



#### 设置头文件包含



### 功能实现

#### 加载身份证识别模块

ScriptablePluginObject(NPP npp)

: ScriptablePluginObjectBase(npp)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*加载神思模块\*\*\*\*\*\*\*\*\*\*\*\*\*\*

hInstance = LoadLibrary("RdCard.dll");

if (!hInstance) {

::MessageBox(NULL, \_\_TEXT("加载神思接口失败，请检查 RdCard.dll,重新打开页面"), \_\_T("提示"), MB\_OK);

}

}

~ScriptablePluginObject() {

FreeLibrary(hInstance);

}

#### 定义功能标识

// 拓展 增加方法

// 获取身份证姓名方法

static NPIdentifier sIdCardName\_id;

//测试阅读器是否连接

static NPIdentifier sIdCardStatus\_id;

修改插件方法判断函数

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*测试插件是否有此方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool

ScriptablePluginObject::HasMethod(NPIdentifier name)

{

return (name == sFoo\_id || name == sIdCardStatus\_id || name == sIdCardName\_id);

}

#### 实现具体功能

// 插件提供方法调用接口

bool

ScriptablePluginObject::Invoke(NPIdentifier name, const NPVariant \*args,

uint32\_t argCount, NPVariant \*result)

{

if (name == sFoo\_id) {// foo 方法

::MessageBox(NULL, \_\_TEXT("开始调用foo方法"), \_\_T("提示"), MB\_OKCANCEL);

char\* m\_szTextGui = "我是foo方法调用返回的结果";

STRINGZ\_TO\_NPVARIANT(toGBK(m\_szTextGui), \*result);

return true;

}

else if(name == sIdCardStatus\_id) {//测试身份证阅读器是否连接

::MessageBox(NULL, \_\_TEXT("开始调用IsConnect方法"), \_\_T("提示"), MB\_OKCANCEL);

typedef int (\*FUN)(...);

FUN fun;

fun = (FUN)GetProcAddress(hInstance, "IsConnect");

if (!fun) {

::MessageBox(NULL, \_\_TEXT(" RdCard.dll 中没有 IsConnect 函数"), \_\_T("提示"), MB\_OK);

return false;

}

bool rst = fun();

BOOLEAN\_TO\_NPVARIANT(rst, \*result);

return true;

}

else if (name == sIdCardName\_id) {// 获取身份证姓名

typedef int (\*FUN)(...);

FUN fun;

fun = (FUN)GetProcAddress(hInstance, "GetName");

if (!fun) {

::MessageBox(NULL, \_\_TEXT(" RdCard.dll 中没有 GetName 函数"), \_\_T("提示"), MB\_OK);

return false;

}

::MessageBox(NULL, \_\_TEXT("开始调用getName方法"), \_\_T("提示"), MB\_OKCANCEL);

char name[20];

fun(name);

STRINGZ\_TO\_NPVARIANT(toGBK(name),\*result);

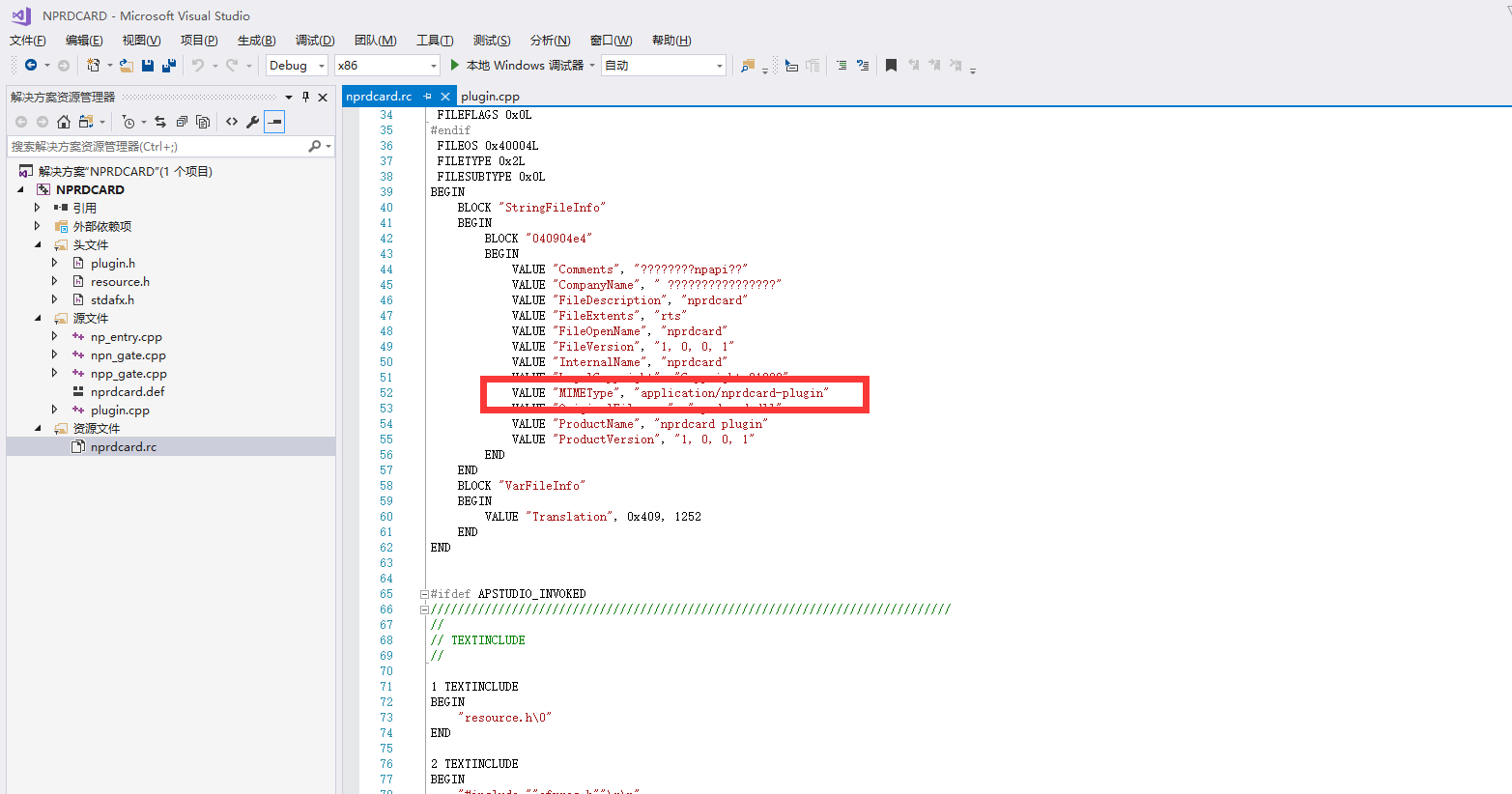
return true;

}

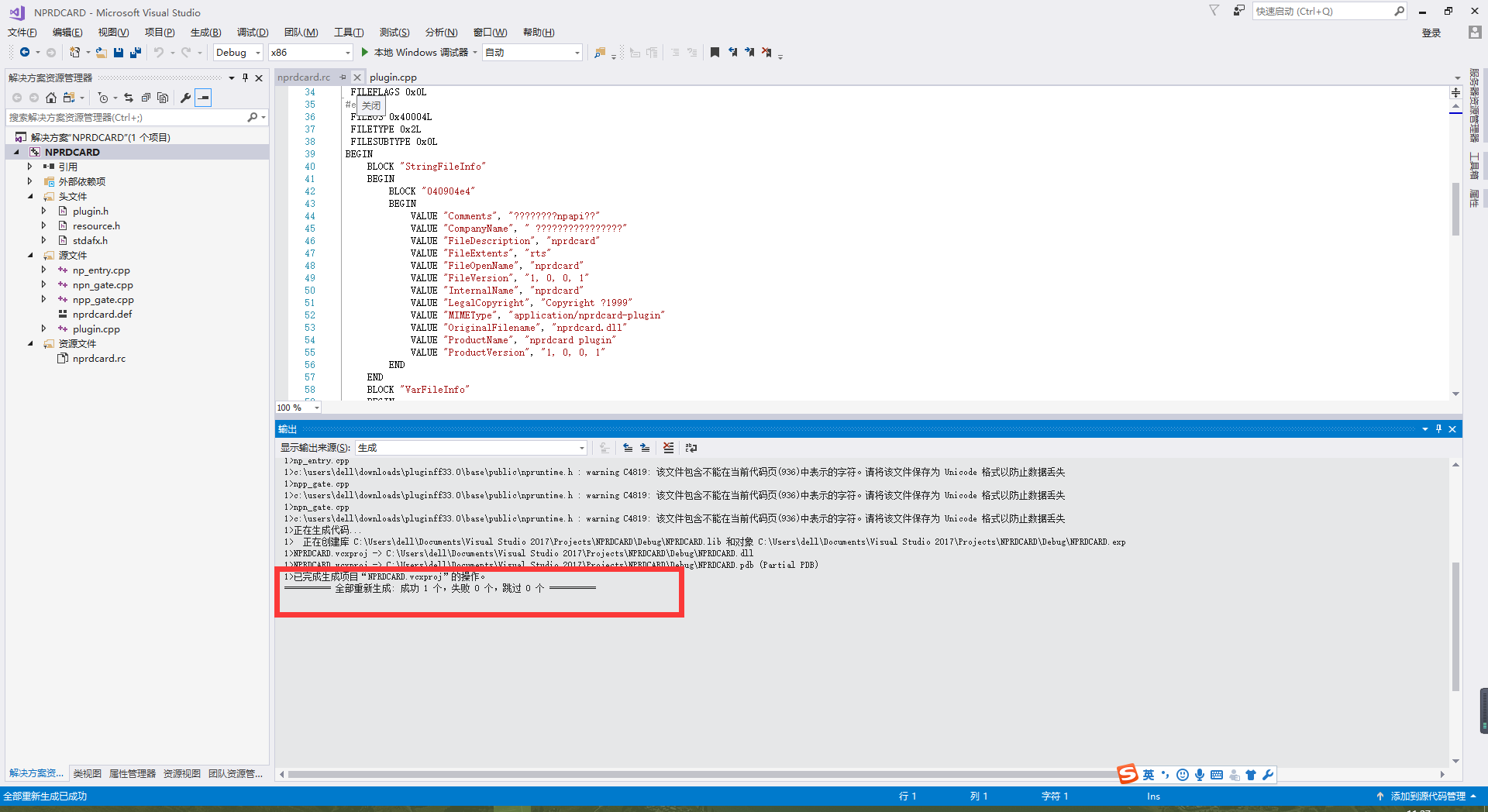
return false;

}

#### 定义插件模块，其中MIMEType，是在html中调用的关键字

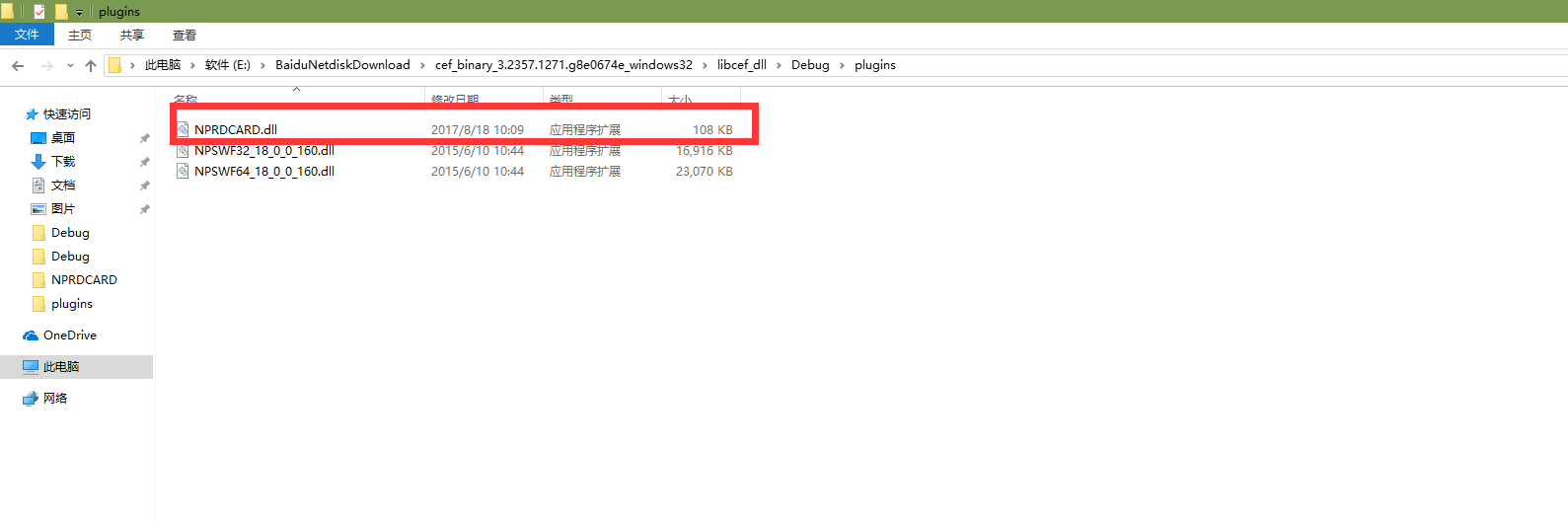


#### 生成插件

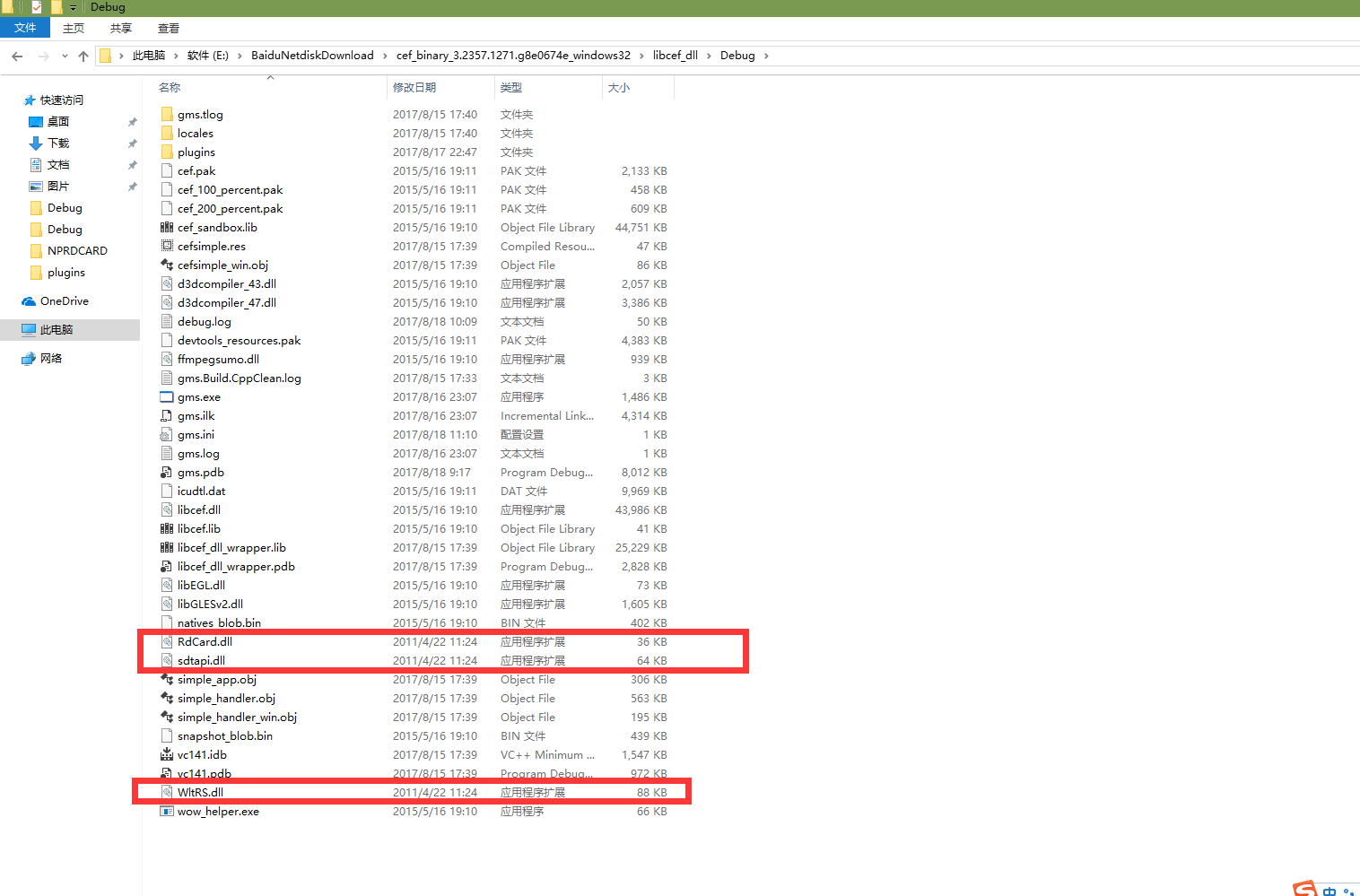


#### 插件部署

将生成的NPRDCARD.dll放入浏览器目录的plugins目录下



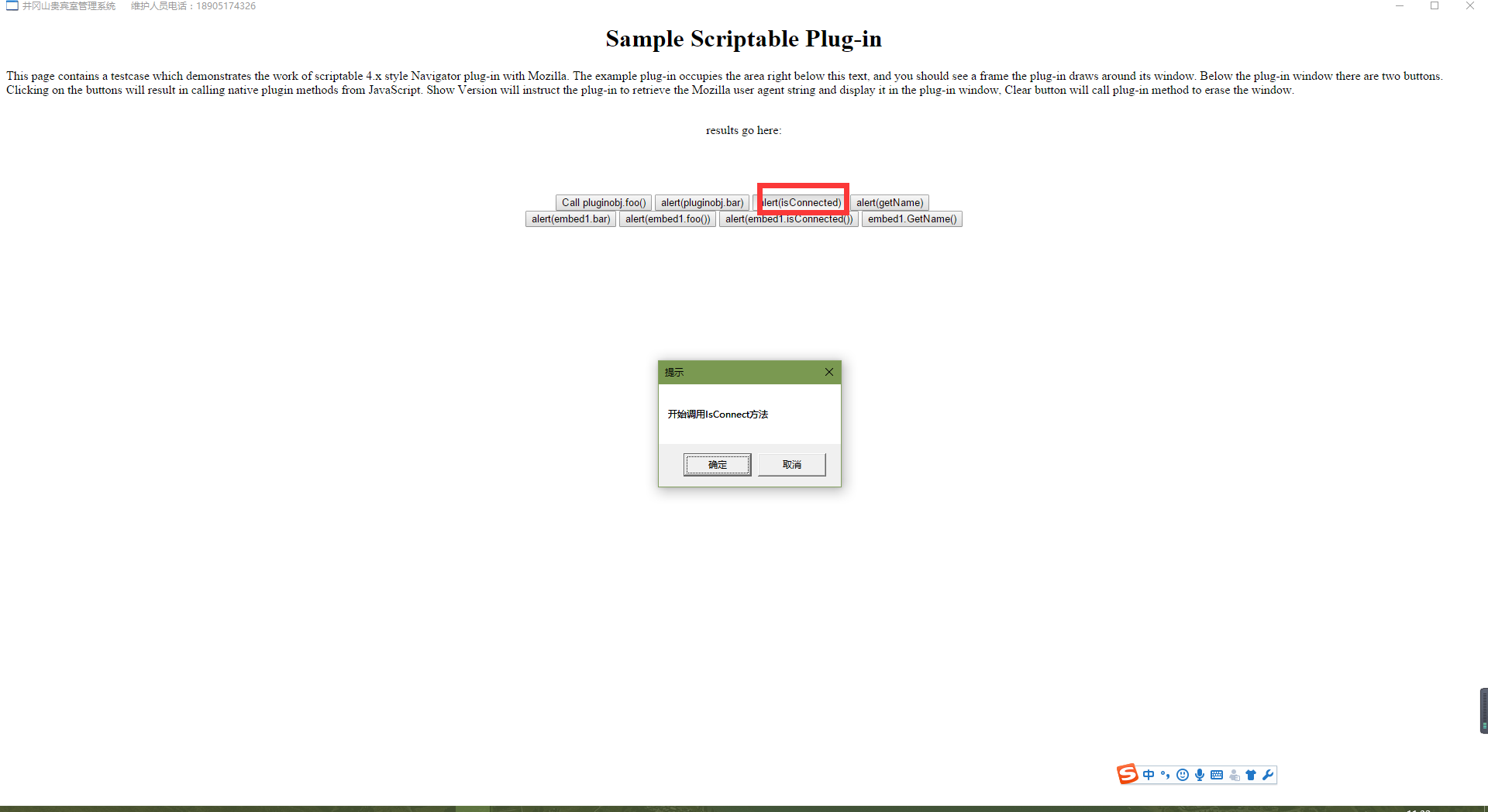
身份证阅读器依赖的dll放在浏览器可执行文件目录下

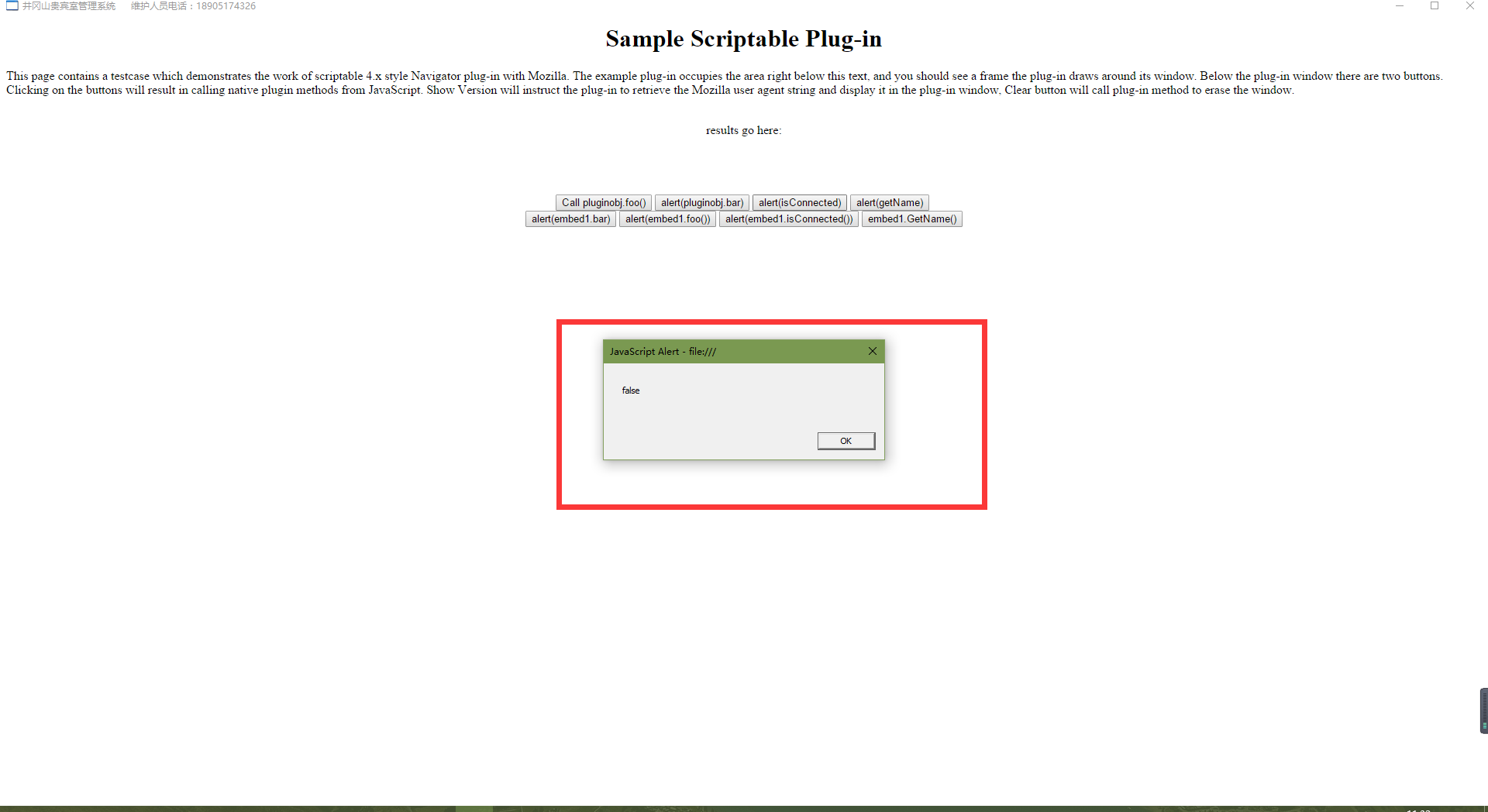


#### 插件测试



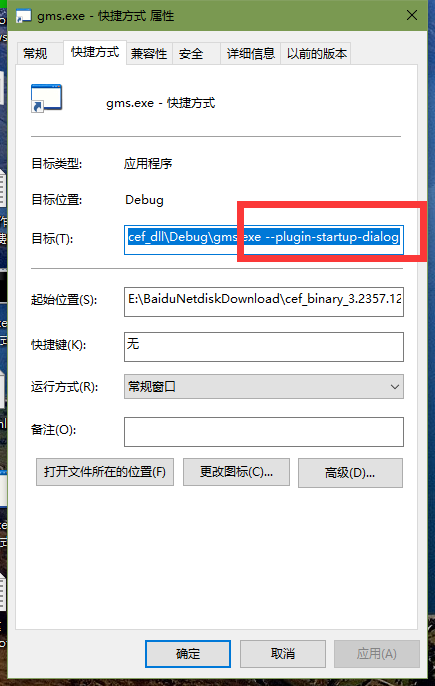
测试身份证阅读器是否连接



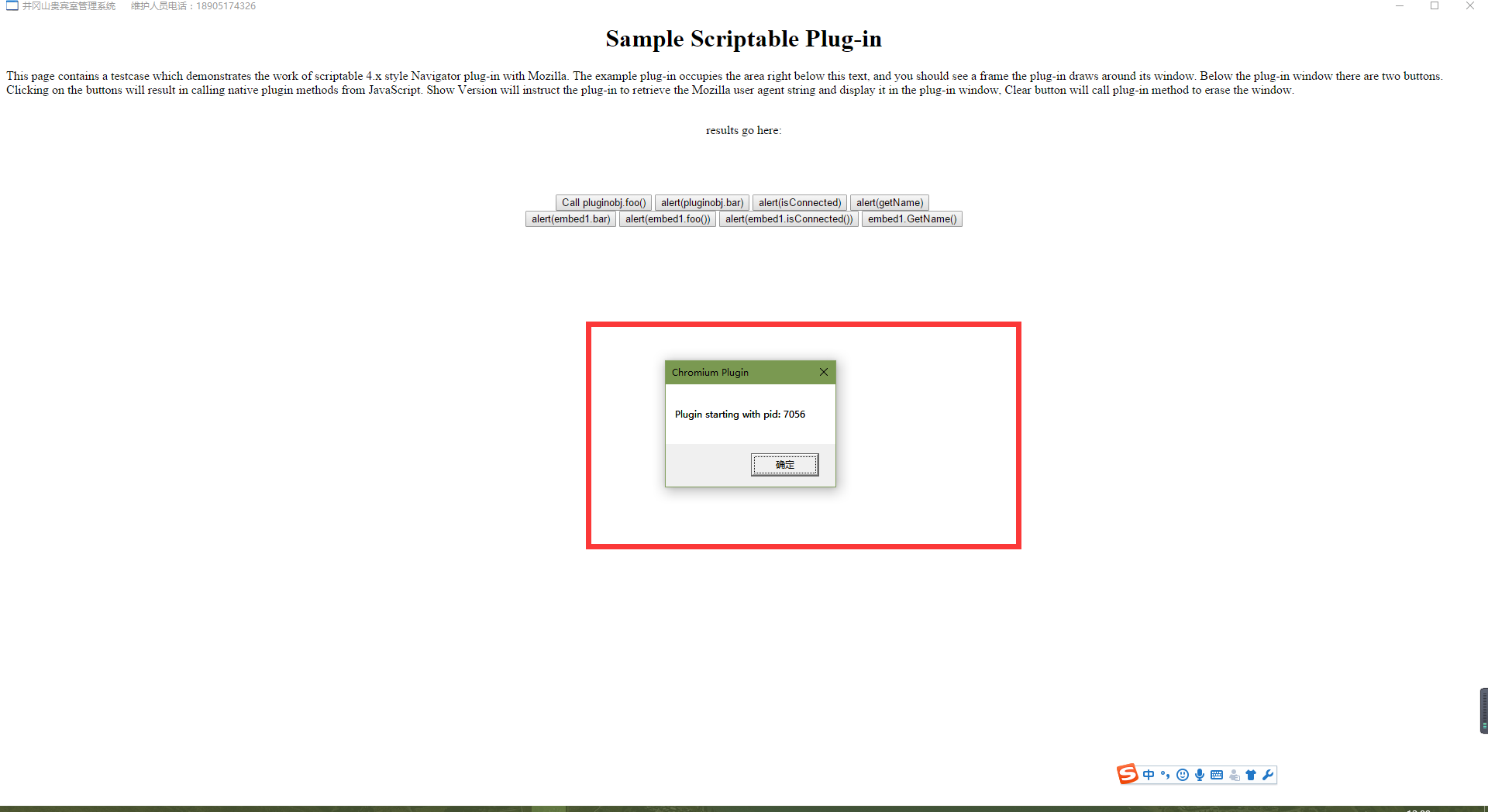


### 插件调试

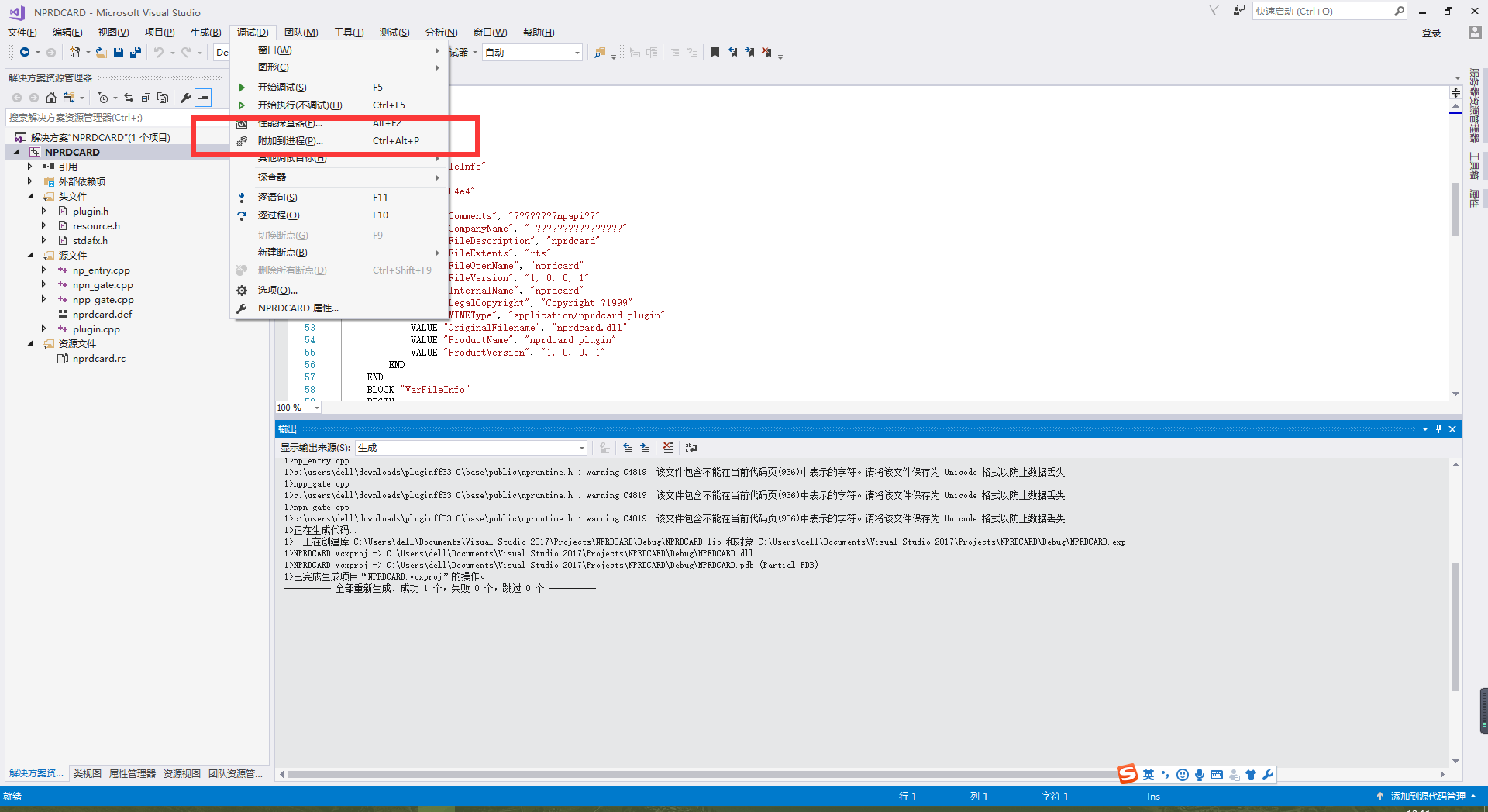
#### 浏览器增加启动参数

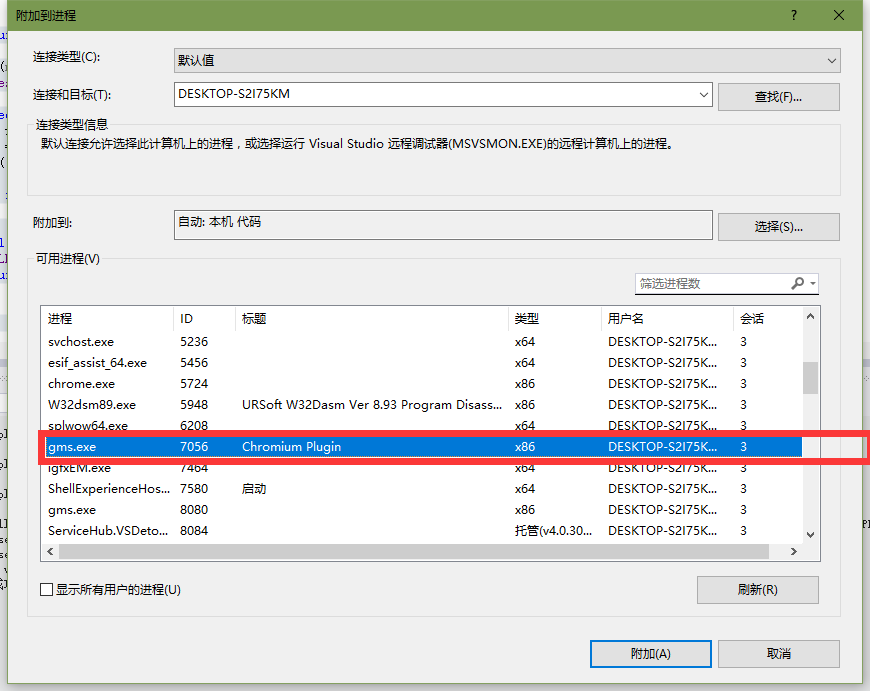


#### 启动浏览器



#### 在vs中打好断点，附加到进程





点击测试是否连接设备



#### 进入断点