

# 跨浏览器 NPAPI 插件开发

## 1. 技术实现

Chrome 浏览器可以识别 NPAPI 插件，为了快速开发 NPAPI 插件支持非 IE 浏览器，NPAPI 插件可以通过 COM 方式直接加载调用 ActiveX 控件，NPAPI 插件只需要实现跟浏览器的交互，NPAPI 插件在 Chrome 浏览器上显示的界面实际仍然是 ActiveX 控件界面。本文主要介绍 NPAPI 插件如何封装 ActiveX 控件。

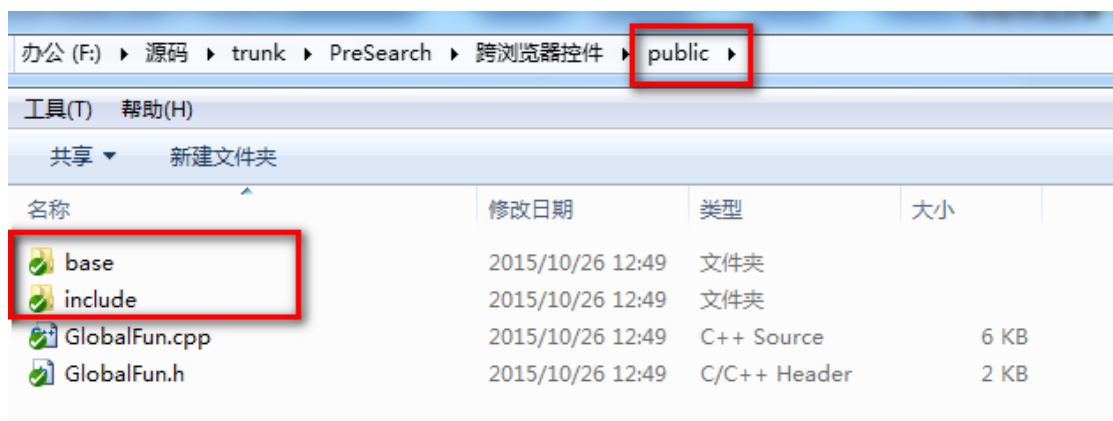
### 1.1 准备材料

NPAPI 开发基本框架代码（包含了 NPAPI 数据结构和接口函数的头文件、np\_entry.cpp、npp\_gate.cpp、npn\_gate.cpp、Plugin.cpp、Plugin.h），控件 AtlDemo.dll 及其生成的 AtlDemo.tlh、AtlDemotli 文件，COM 方式封装控件接口的 NPAtlDemo.cpp、NPAtlDemo.h 文件，封装 NP 数据转换的 GlobalFun.cpp、GlobalFun.h，封装回调事件的 SinkObj.cpp、SinkObj.h；NP 注册脚本 npreg.bat，chrome 浏览器 demo 脚本 np-test.htm，IE 注册脚本 iereg.bat，IE 浏览器 demo 脚本 ie-test.htm。（各文件详细描述见 1.2.5 章节）

### 1.2 实现

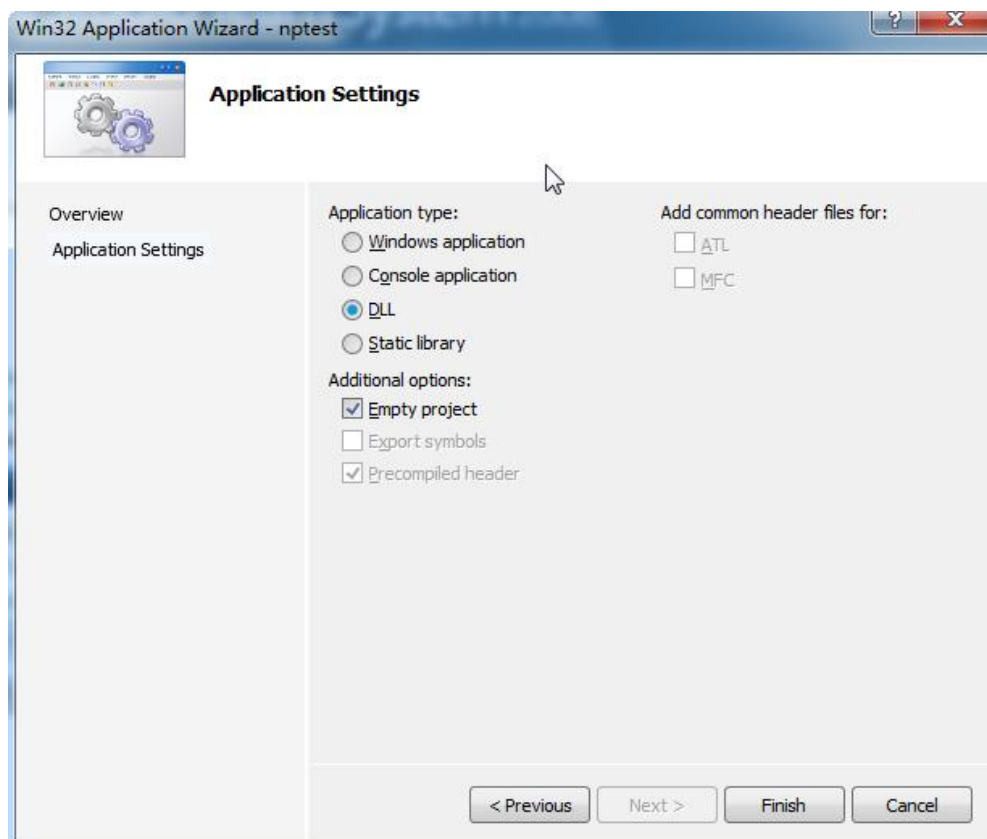
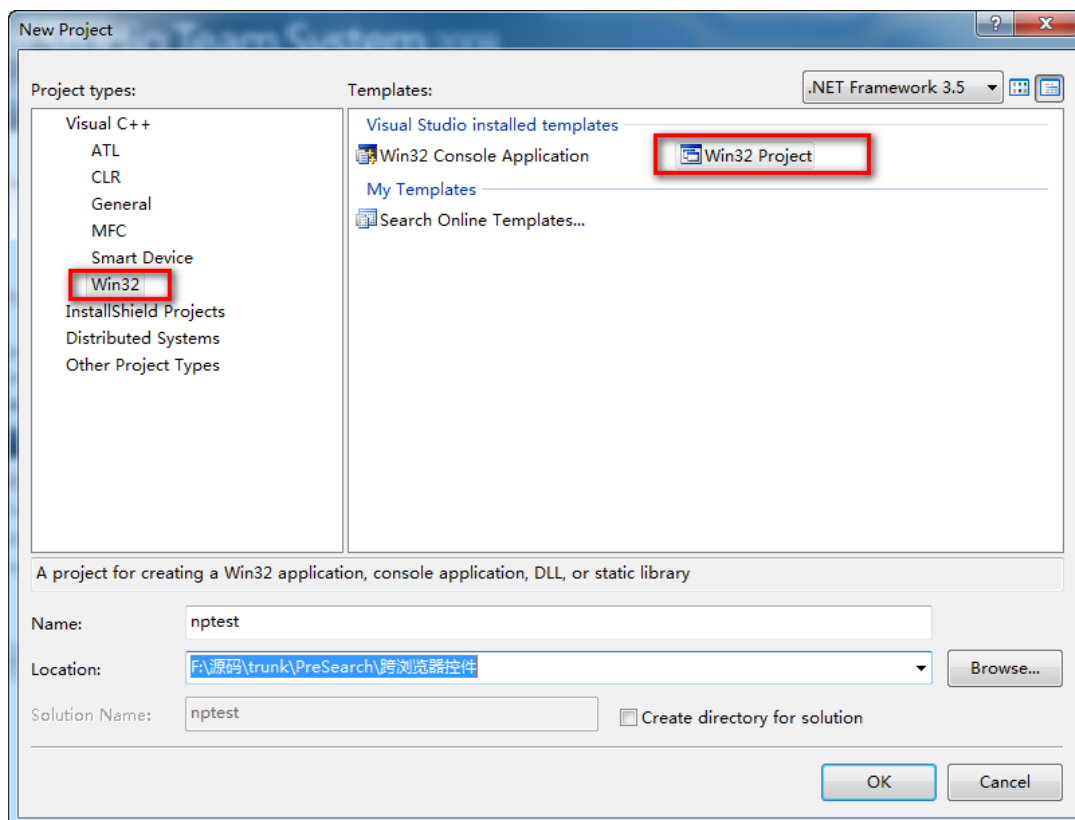
#### 1.2.1 新建 public 文件夹

把 npapi 开发基本框架代码中的 base 和 include 目录和 GlobalFun.cpp、GlobalFun.h 拷贝到 public 目录下。



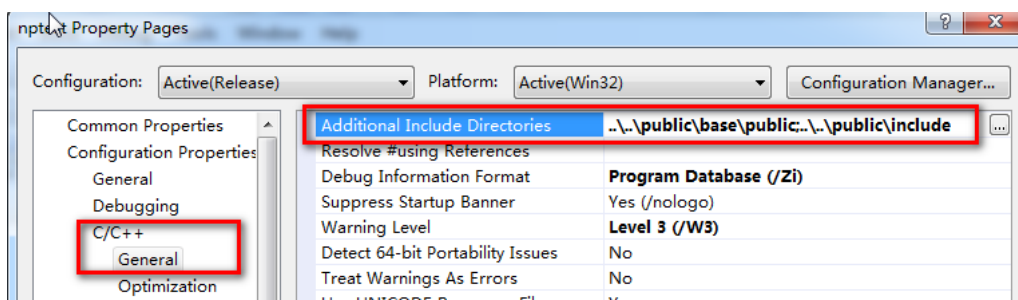
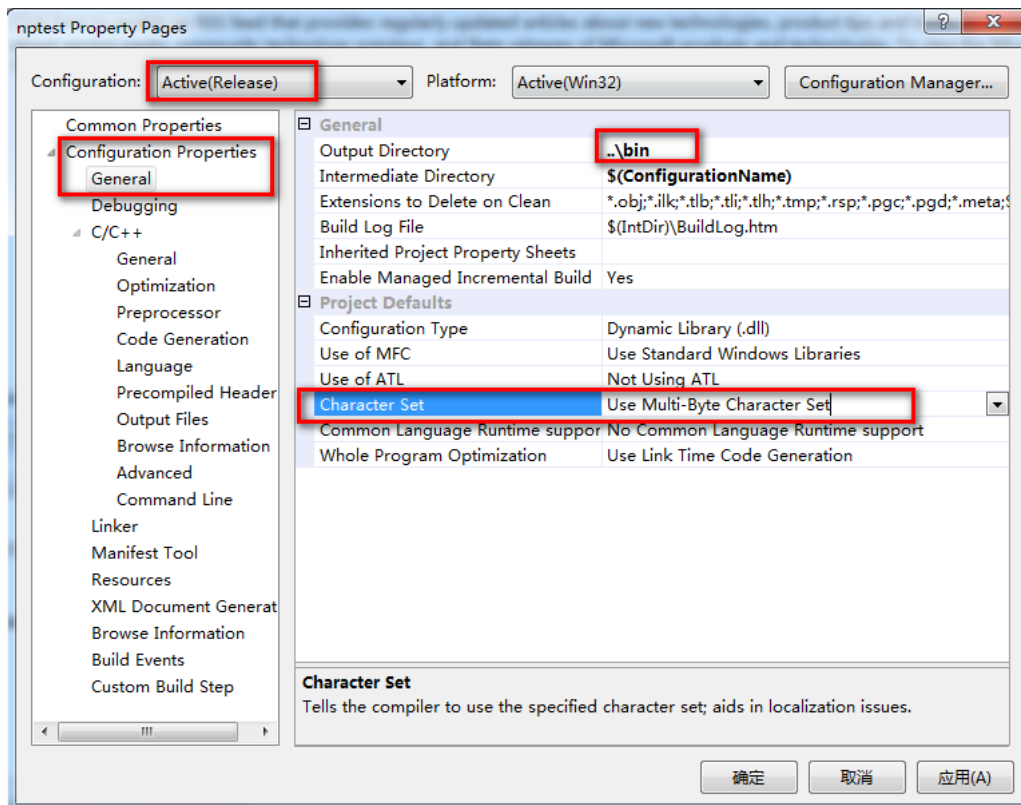
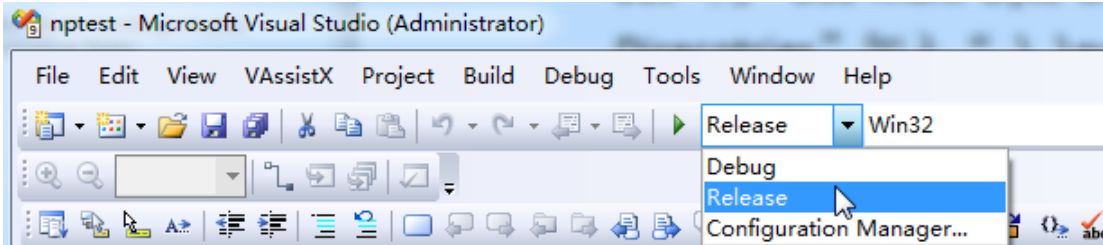
#### 1.2.2 新建 Win32 Dll 空工程

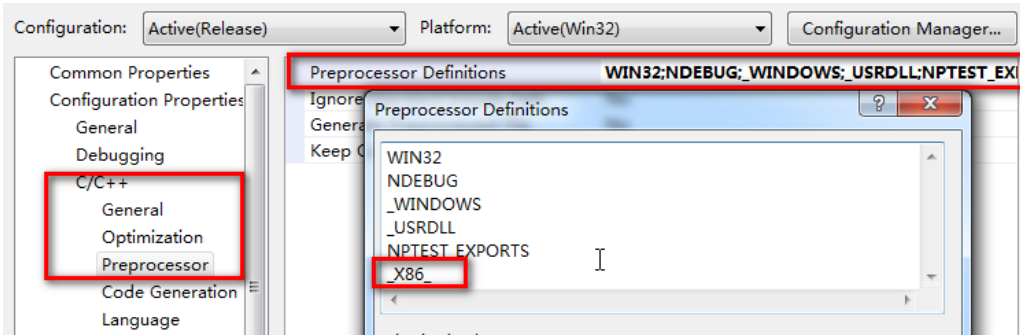
如 nptest，路径与 public 在同级目录。



### 1.2.3 修改工程配置

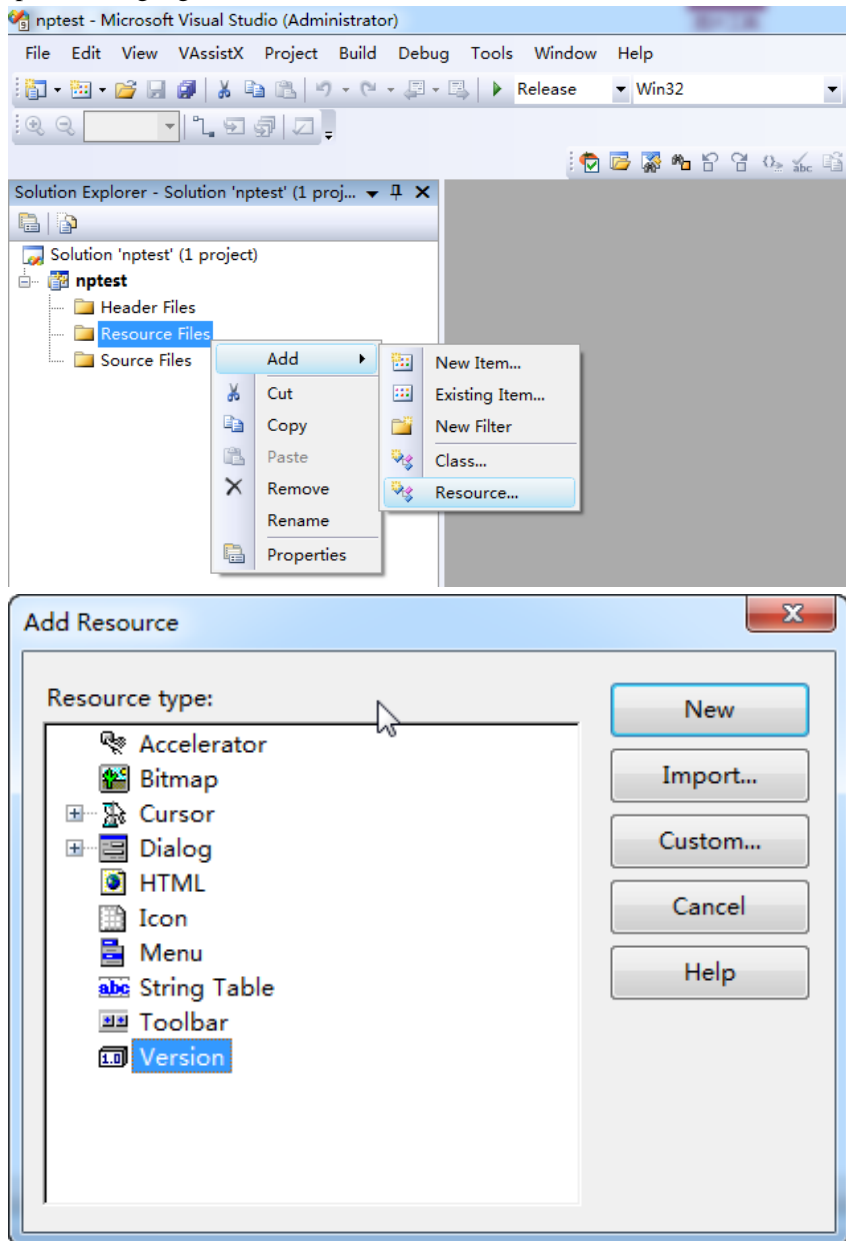
选中工程，工程配置选为Release，单击鼠标右键，选择属性，在“general”中设置“Character Set”为“Use Multi-Byte Character Set”，“C/C++”下“General”设置“Additional Include Directories”加入“..\public\base\public;..\public\include”（相对目录不要搞错），“C/C++”下“Preprocessor”设置“Preprocessor Definitions”中加入“\_X86\_”，成果物生成路径为public同级目录的bin目录。

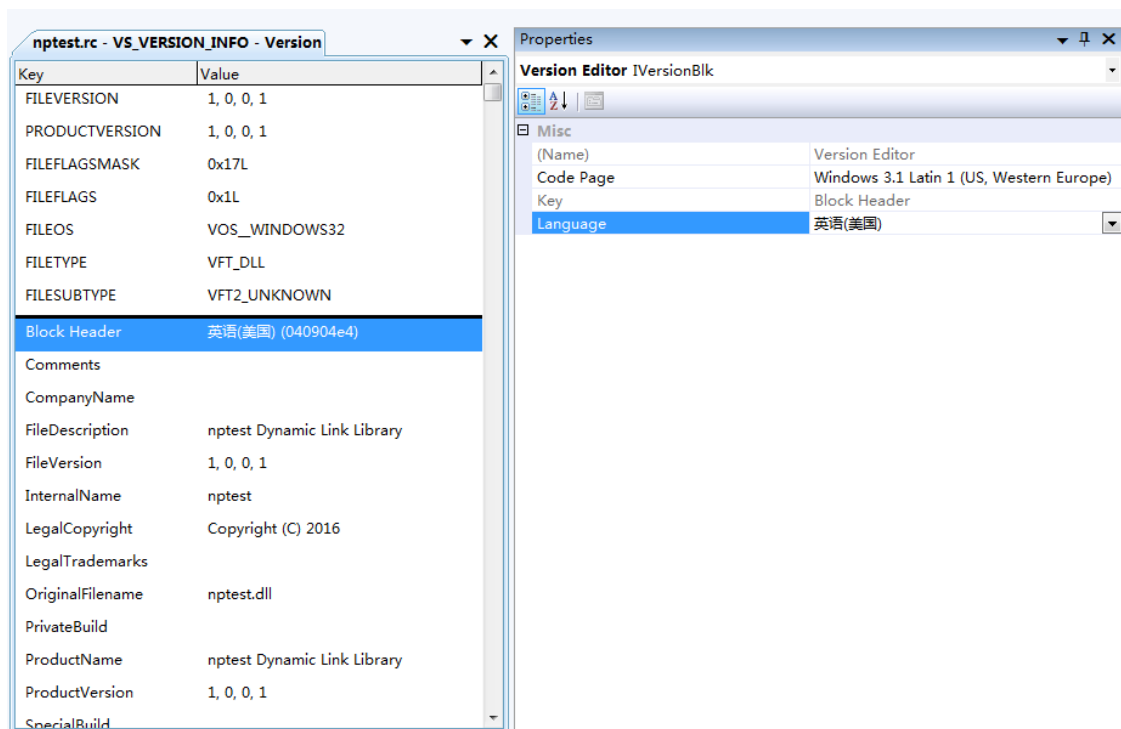




## 1.2.4 新增并修改资源文件

新增资源-Version, 双击选中 Block Header, 修改“Code page”为“Windows 3.1 Latin 1 (US, Western Europe)”, Language 为“英语(美国)”, 保存;





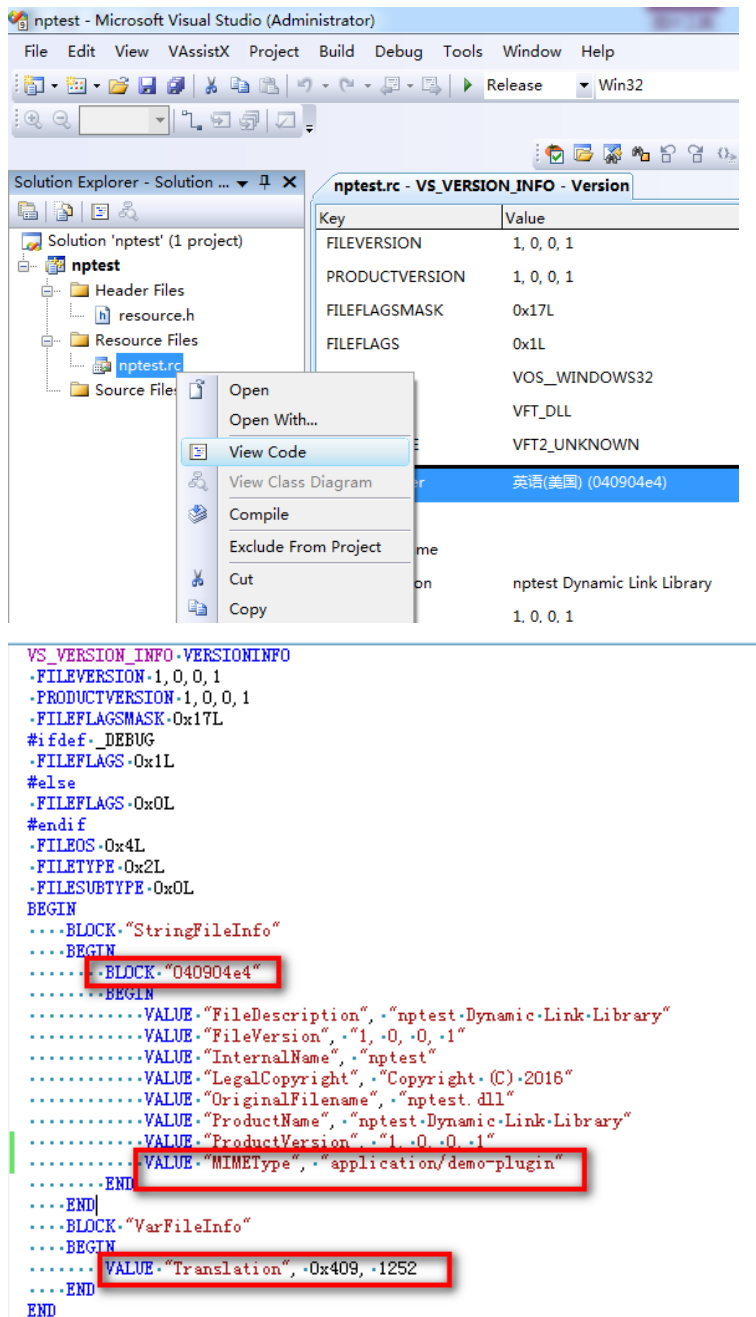
然后选中资源文件nptest.rc，单击鼠标右键，选择“View Code”，在VS\_VERSION\_INFO新增一行：

VALUE "MIMETYPE", "application/demo-plugin"

这个值作为NPAPI插件的唯一标识，类似于ActiveX控件的ClassID，需要区分设置；

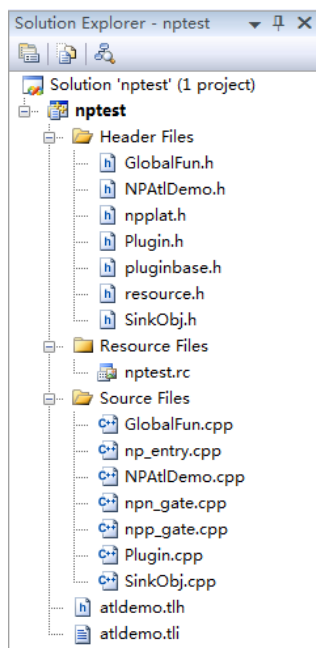
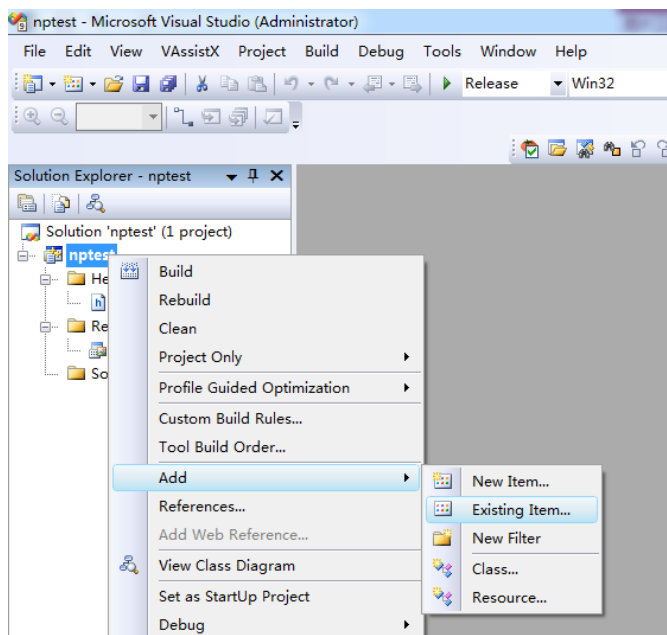
VarFileInfo下Translation的值必须改成“0x409, 1252”，否则插件在chrome中无法识别，“0x409, 1252”的含义是对应的，1252的十六进制表示就是4e4，Translation字段的第一个值表示语言409表示英语，而默认的804表示中文(简体)；Translation字段的第二个值表示所采用的字符集，1200(0X04B0)表示Unicode，1252（0X04E4）表示多字节字符集，如下：

```
BLOCK "VarFileInfo"
BEGIN
    VALUE "Translation", 0x409, 1252
END
```



### 1.2.5 导入核心代码

把np\_entry.cpp、nnp\_gate.cpp、npp\_gate.cpp、Plugin.cpp、Plugin.h、AtlDemo.tlh、AtlDemotli、NPAtlDemo.cpp、NPAtlDemo.h、SinkObj.cpp、SinkObj.h文件拷贝到工程目录并导入工程，再导入include目录的applat.h/pluginbase.h。

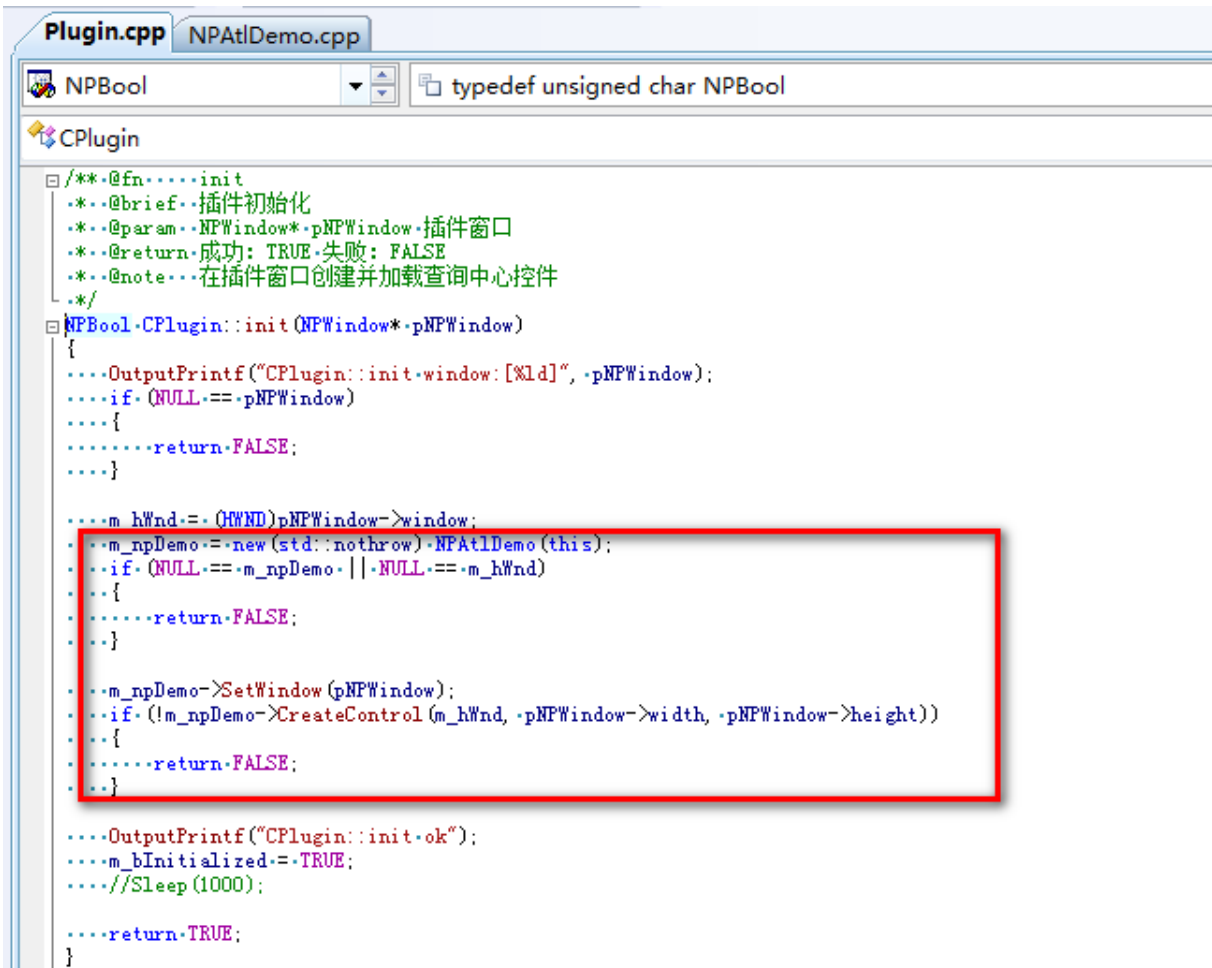


其中各代码文件说明：

- 1) ActiveX控件生成的tlh文件中稍作修改（删除命名空间），它们记录了控件的所有接口和事件；
- 2) np\_entry.cpp、npp\_gate.cpp、npp\_gate.cpp 是 NPAPI 框架封装 NPAPI 插件与浏览器相互调用的代码；
- 3) NPAtlDemo.h/NPAtlDemo.cpp 的 NPAtlDemo 类通过 ATL.dll 根据 ActiveX 控件的 ProgID 加载控件并封装了控件接口；
- 4) Plugin.h/Plugin.cpp 含有两个继承于 np 框架（基类为 nsPluginInstanceBase、NPObject）的两个类（CPlugin、CPluginScriptObject），通过调用 NPAtlDemo 类来创建 ATL 控件并调用 ATL 控件接口；
- 5) GlobalFun.h/GlobalFun.cpp 是公共文件；
- 6) SinkObj.h/SinkObj.cpp 是接收控件事件的类，供 NPAtlDemo 类调用，如果 ActiveX 控件没有回调事件，一定要删掉对 SinkObj 文件的引用。

重点在于 NPAtlDemo、CPlugin、CPluginScriptObject 三个类，NPAtlDemo 加载并封装了控件接口；CPlugin、CPluginScriptObject 是继承 np 框架的类，CPlugin 类的 init 函数创建 NPAtlDemo 对象，进而调用到 ActiveX 控件；CPluginScriptObject 类的 hasMethod 函数判断页面调用的接口是否存在于控件里，如果存在，hasMethod 函数返回 true，之后就会进入 invoke 函数，去通过 CPlugin 类调用到 ActiveX 控件接口；如果不存在，hasMethod 返回 false，不会进入 invoke 函数。（控件属性对应函数为 hasProperty、setProperty）

关键函数代码实现如下：



```
Plugin.cpp NPAtlDemo.cpp
NPBool
typedef unsigned char NPBool
CPlugin
/**.@fn.....init
  *..@brief..插件初始化
  *..@param..NPWindow*.pNPWindow..插件窗口
  *..@return..成功: TRUE.失败: FALSE
  *..@note...在插件窗口创建并加载查询中心控件
  */
NPBool CPlugin::init(NPWindow*.pNPWindow)
{
    ....OutputPrintf("CPlugin::init-window:[%ld]", .pNPWindow);
    ....if(.NULL==.pNPWindow)
    ....{
    ....    ....return.FALSE;
    ....}

    ....m_hWnd.=.(HWND)pNPWindow->window;
    ....m_npDemo.=.new(std::nothrow).NPAtlDemo(this);
    ....if(.NULL==.m_npDemo. || .NULL==.m_hWnd)
    ....{
    ....    ....return.FALSE;
    ....}

    ....m_npDemo->SetWindow(pNPWindow);
    ....if.(!m_npDemo->CreateControl(m_hWnd, .pNPWindow->width, .pNPWindow->height))
    ....{
    ....    ....return.FALSE;
    ....}

    ....OutputPrintf("CPlugin::init-ok");
    ....m_bInitialized.=.TRUE;
    ....//Sleep(1000);

    ....return.TRUE;
}
```



Plugin.cpp NPAtDemo.cpp

CPluginScriptObject.hasN bool CPluginScriptObject::hasMethod(NPIdentifier

CPluginScriptObject

```

/**@fn.....hasMethod
...@brief..判断是否有该方法
...@param..NPIdentifier.methodName.函数名称
...@return..存在返回true, 不存在返回false
...@note...与method相关的函数为hasMethod和Invoke, 如果要为插件创建一个method,
*/
bool CPluginScriptObject::hasMethod(NPIdentifier.methodName)
{
    ...char *pFunc=-NPN_UTF8FromIdentifier(methodName);
    ...OutputPrintf("hasMethod:%s", pFunc);
    ...if (strcmp(pFunc, "Test")==0)
    ...{
    .....return true;
    ...}
    ...return false;
}

```

Plugin.cpp NPAtDemo.cpp

CPluginScriptObject.invol if (NULL == npp)

CPluginScriptObject invoke

```

/**@Function:.....CPluginScriptObject::invoke
...@Description:.....NPAPI接口调用
...@param-input:.....methodName:.....方法名
...@param-args:.....接口参数列表
...@param-argCount:.....接口参数个数
...@param-output:.....result:.....接口返回值
...@return-value:.....bool:.....true-成功, false-失败
...@note:.....<member-Func>
*/
bool CPluginScriptObject::invoke(NPIdentifier.methodName, const NPVariant *args, uint32_t argCount, NPVariant *result)
{
    ...if (NULL==npp)
    ...{
    .....return false;
    ...}
    ...CPlugin *pPlugin=(CPlugin*)npp->pdata;
    ...if (NULL==pPlugin)
    ...{
    .....return false;
    ...}

    ...char *pFunc=-NPN_UTF8FromIdentifier(methodName);
    ...OutputPrintf("invoke:%s", pFunc);
    ...long nRet;

    ...if (strcmp(pFunc, "Test")==0)
    ...{
    .....if (args!=NULL && argCount>=2)
    .....{
    .....std::string first=-NPVARIANT_TO_GBK(args[0]);
    ....._bstr_t szXml=-_com_util::ConvertStringToBSTR(first.c_str());
    .....long nData=-NPVARIANT_TO_LONG(args[1]);

    .....nRet=-pPlugin->Test(-szXml, nData);
    .....INT32_TO_NPVARIANT(nRet, result);
    .....return true;
    .....}
    .....OutputPrintf("%s.function.argument.error", pFunc);
    .....return false;
    ...}

    ...return false;
}

```



```

..//CSinkObj用来接收控件回调事件, 如果没有回调事件, AtlGetObjectSourceInterface会出现异常, 请删除跟CSinkObj有关的以下代码
..m_pSVIIEvents=new(std::nothrow)CSinkObj();
..if(NULL==m_pSVIIEvents)
..{
.....OutputPrintf("Fail to Create CSinkObj");
.....MessageBox(NULL, "Error", "Fail to Create CSinkObj", MB_OK);
.....CoUninitialize();
.....return FALSE;
..}
..hr=AtlGetObjectSourceInterface(m_pIDemo, &m_pSVIIEvents->m_libid,
.....&m_pSVIIEvents->m_iid, &m_pSVIIEvents->m_wMajorVerNum, &m_pSVIIEvents->m_wMinorVerNum);
..if(FAILED(hr))
..{
.....sprintf_s(szMsg, strlen(szMsg), "CreateControlAtlGetObjectSourceInterface-Fail: %d", GetLastError());
.....OutputPrintf(szMsg);
.....MessageBox(NULL, "Error", szMsg, MB_OK);
.....CoUninitialize();
.....return FALSE;
..}

..m_pSVIIEvents->m_pPlugin=m_pPlugin;

..//为事件源建立连接
..hr=m_pSVIIEvents->DispEventAdvise(m_pIDemo, &m_pSVIIEvents->m_iid);
..if(FAILED(hr))
..{
.....sprintf_s(szMsg, strlen(szMsg), "CreateControlDispEventAdvise-Fail: %d", GetLastError());
.....OutputPrintf(szMsg);
.....MessageBox(NULL, "Error", szMsg, MB_OK);
.....CoUninitialize();
.....return FALSE;
..}

```

## 1.2.6 新建 def 文件，导出 NPAPI 插件 3 个关键函数

新建一个\*\*\*.def文件，内容为：

```
LIBRARY "npctest"
```

```
EXPORTS
```

```
NP_GetEntryPoints @1
```

```
NP_Initialize @2
```

```
NP_Shutdown @3
```

注：npctest对应为成果物的名称（npctest.dll）。

## 1.2.7 修改 NPAPI 插件唯一标识

修改npp\_gate.cpp的NPP\_GetMIMEDescription函数的返回值，应与rc文件中的“MIMEType”一致。

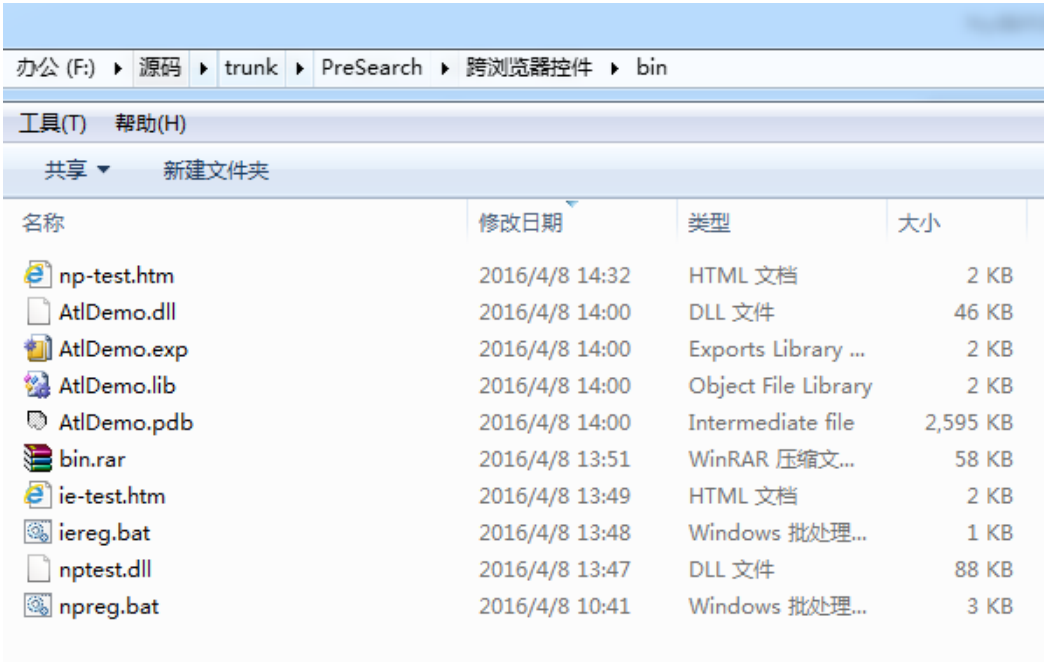
## 1.2.8 修改核心代码，实现控件接口

依据AtlDemo.tlh文件修改具体接口，对应为Plugin.h、Plugin.cpp、NPAtlDemo.cpp、NPAtlDemo.h、SinkObj.cpp、SinkObj.h文件，NPAtlDemo.cpp、NPAtlDemo.h加载ATL插件，并一一封装AtlDemo.tlh的对应接口，SinkObj.cpp、SinkObj.h对应AtlDemo控件的回调函数，Plugin.h、Plugin.cpp则实现与JS页面交互的接口，特别注意hasMethod、hasProperty、getProperty、setProperty、invoke函数对函数名、变量名的比对不要出现错误。

## 1.3 浏览器页面调用差异

所有工程设置成果物生成到 bin 目录下，页面脚本及 demo 都拷贝到 bin 目录。iereg.bat 注册海康威视版权所有

ActiveX 控件；npreg.bat 注册 NPAPI 插件；ie-test.htm 是 ie 示例页面，调用 ActiveX 控件（前提是 ActiveX 控件已注册）；np-test.htm 是 chrome 示例页面，调用 NPAPI 插件（前提是 ActiveX 控件和 NPAPI 插件均已注册）。



The screenshot shows a Windows File Explorer window with the address bar displaying the path: 办公 (F:) > 源码 > trunk > PreSearch > 跨浏览器控件 > bin. The window title is '工具(T) 帮助(H)'. Below the title bar, there are buttons for '共享' and '新建文件夹'. The main area displays a list of files and folders with columns for '名称' (Name), '修改日期' (Modified Date), '类型' (Type), and '大小' (Size).

名称	修改日期	类型	大小
np-test.htm	2016/4/8 14:32	HTML 文档	2 KB
AtlDemo.dll	2016/4/8 14:00	DLL 文件	46 KB
AtlDemo.exp	2016/4/8 14:00	Exports Library ...	2 KB
AtlDemo.lib	2016/4/8 14:00	Object File Library	2 KB
AtlDemo.pdb	2016/4/8 14:00	Intermediate file	2,595 KB
bin.rar	2016/4/8 13:51	WinRAR 压缩文...	58 KB
ie-test.htm	2016/4/8 13:49	HTML 文档	2 KB
iereg.bat	2016/4/8 13:48	Windows 批处理...	1 KB
np-test.dll	2016/4/8 13:47	DLL 文件	88 KB
npreg.bat	2016/4/8 10:41	Windows 批处理...	3 KB

特别注意 IE 示例和 NP 示例页面有两处差别：

### 1.3.1 加载插件脚本差异

IE 示例通过 object 加载控件，**classid** 为 ActiveX 控件唯一标识：

```
<div id="ocxContainer" style="width:100%;height:70%">
  <object classid="clsid:8F55A11E-35B9-4B5F-98BD-53502C1590DD" id="ocx" width="100%"
height="100%" name="ocx" >
  </object>
</div>
```

NP 示例通过 embed 加载 NPAPI 插件，**type** 为 NPAPI 插件唯一标识 MIMETYPE：

```
<div id="ocxContainer" style="width:100%;height:70%">
  <embed type="application/hik-demo-plugin" id="ocx" width="800" height="600"
name="ocx"></embed>
</div>
```

### 1.3.2 回调函数处理

IE 示例中回调函数在单独的 script 中接收 id 为 ocx 的控件的事件：

```
<script language="javascript" for="ocx" event="MsgNotify(iMsg,szDetail)">
  szMsg = "msg:" + iMsg + ",detail:" + szDetail;
  alert(szMsg);
</script>
```

NP 示例中回调函数跟普通 js 函数一样：

```
function MsgNotify(iMsg,szDetail)
{
    szMsg = "msg:" + iMsg + ",detail:" + szDetail;
    alert(szMsg);
}
```

## 2. 经验总结

开发 NPAPI 插件过程中，碰到不少问题，这里总结如下：

- 1、浏览器不能识别 NPAPI 插件，存在的几点原因如下：
  - 1) Chrome 版本问题，Chrome45 及以上版本已经不支持 NPAPI 插件，Chrome42-Chrome44 版本需在页面 `chrome://flags/#enable-npapi` 设置启用 npapi，其他 Chrome 版本及 Firefox 暂不限制 NPAPI 的使用。
  - 2) 资源文件有问题，如资源文件 Code page 或 Language 设置不对，或者 VarFileInfo 下 Translation 的值不对，具体见 1.2.4 章节；
  - 3) 没有增加 def 文件，导致 NPAPI 插件的 3 个关键函数没有导出，具体见 1.2.6 章节；
- 2、ActiveX 控件有回调函数，但 NPAPI 插件加载回调函数处理类执行 `AtlGetObjectSourceInterface` 函数时会崩溃，可能是由 ActiveX 控件引用基类 `IProvideClassInfo2Impl` 的第二个参数为 NULL 导致，需传对应回调事件类的名称，如 “`public IProvideClassInfo2Impl<&CLSID_IAtlDemoCtrl, &__uuidof(_IIAtlDemoCtrlEvents), &LIBID_AtlDemoLib>`”；如果控件没有控件事件，`NPAtlDemo` 类中一定要删除对 `SinkObj` 的引用。
- 3、ActiveX 控件回调函数中含有字符串的情况，如果某个回调消息的字符串参数为空，不要传 `L””`，而应该直接传 `NULL`，以防出现 NPAPI 插件偶现崩溃的现象。
- 4、`npp_gate.cpp` 的 `NPP_New` 函数比对传入的属性名称（`argn[argCur]`）时，注意该接口传入的属性参数字母都是小写。
- 5、Chrome 浏览器，页面传给控件的字符串必须有终止符 `\0`，不然控件收到的字符串后面会是乱码。
- 6、代码中替换控件时，注意 `NPAtlDemo::CreateControl` 函数中要更新控件的 `ProgID`。

经验案例地址：

<http://rd.hikvision.com.cn/hikvision/rdweb/wfexpercase.nsf/c358ca19074fe6ca4825797b003854af/96ad39391a5cc30048257fba0031dd50?OpenDocument>