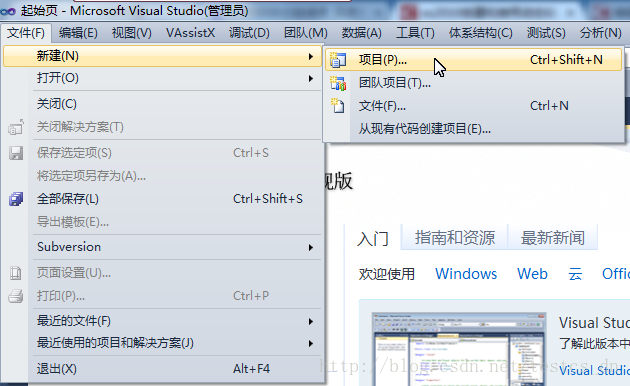
**VS2010编写动态链接库DLL及单元测试用例，调用DLL测试正确性**

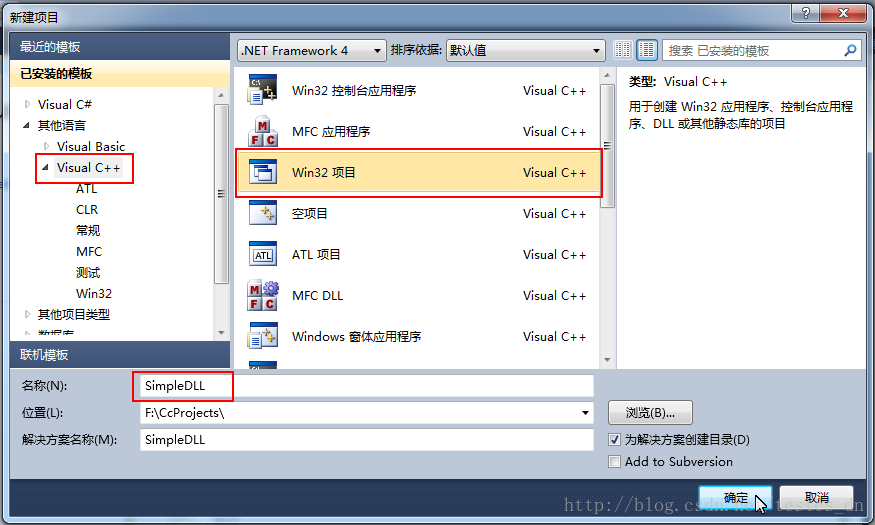
本文将创建一个简单的动态链接库，并编写一个控制台应用程序使用该动态链接库，该动态链接库为“JAVA调用动态链接库DLL之JNative学习”中使用的DLL，只是项目及文件名称不同。

创建动态链接库项目：

1、打开Microsoft Visual Studio 2010，选择文件->新建->项目。

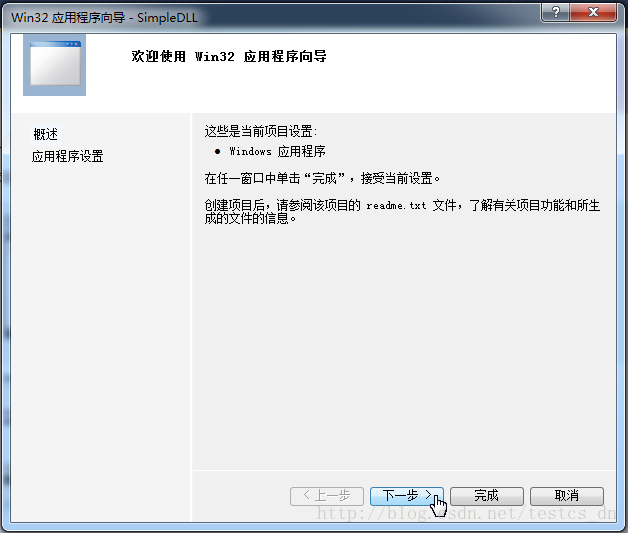


2、在新建项目窗口中选择其他语言->Visual C++->Win32。



3、选择Win32 项目，设置名称：simpleDLL，设置解决方案名：simpleDLL。

4、单击确定，在出现的Win32 应用程序向导的概述对话框中点击下一步。



5、在应用程序设置中，选择应用程序类型下的DLL。

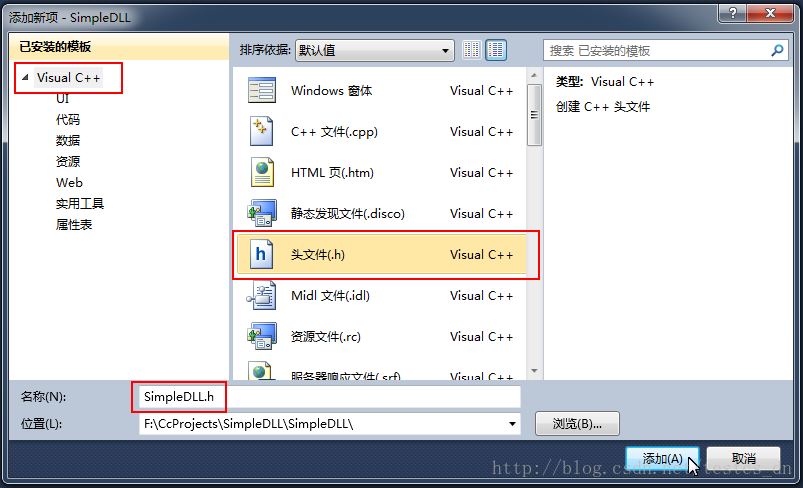
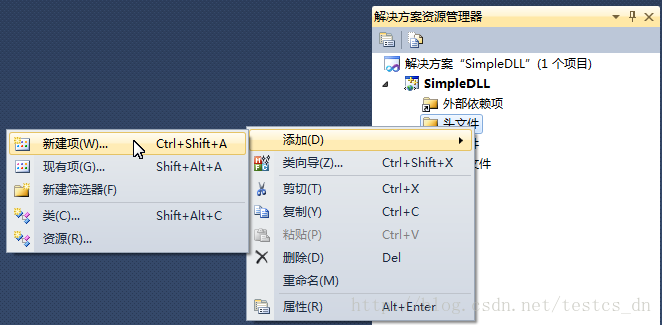


6、勾选附加选项下的空项目。

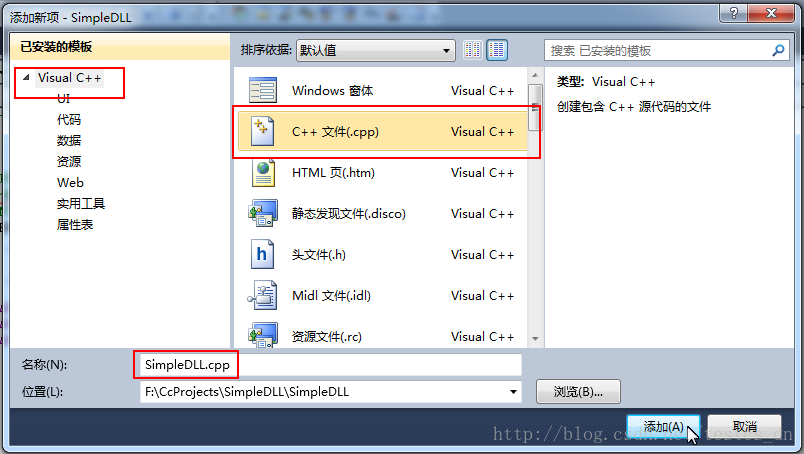
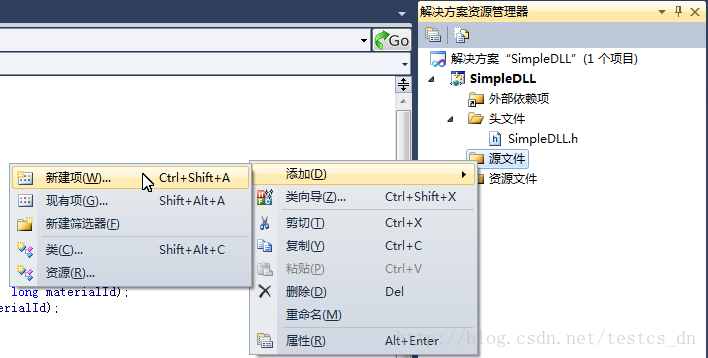
7、单击完成创建项目。

向动态链接库添加类：

1、添加新类头文件。右键单击simpleDLL项目，添加->新建项，选择头文件(.h)，设置名称为simpleDLL，单击添加。



2、添加新类源文件。右键单击simpleDLL项目，添加->新建项，选择C++ 文件(.cpp)，设置名称为simpleDLL，单击添加。



3、为新类添加内容。内容如下：

头文件simpleDLL.h：

//------------------ SimpleDLL.h ----------------

#pragma once;

//该宏完成在dll项目内部使用\_\_declspec(dllexport)导出

//在dll项目外部使用时，用\_\_declspec(dllimport)导入

//宏DLL\_IMPLEMENT在SimpleDLL.cpp中定义

#ifdef DLL\_IMPLEMENT

#define DLL\_API \_\_declspec(dllexport)

#else

#define DLL\_API \_\_declspec(dllimport)

#endif

DLL\_API int add(int x, int y); //简单方法

DLL\_API const wchar\_t\* getPlayUrl(const wchar\_t\* mgrIp, long mgrPort, long materialId);

DLL\_API const char\* getUrl(const char\* mgrIp, long mgrPort, long materialId);

源文件simpleDLL.cpp:

//------------------ SimpleDLL.cpp ----------------

//注意此处的宏定义需要写在#include "SimpleDLL.h"之前

//以完成在dll项目内部使用\_\_declspec(dllexport)导出

//在dll项目外部使用时，用\_\_declspec(dllimport)导入

#define DLL\_IMPLEMENT

#include "SimpleDLL.h"

#include<Windows.h>

#include <intrin.h>

#include <stdlib.h>

#include <string.h>

int DLL\_API add(int x, int y)

{

return x+y;

}

DLL\_API const wchar\_t\* getPlayUrl(const wchar\_t\* mgrIp, long mgrPort, long materialId)

{

static wchar\_t url[260] = { 0 };

wcscpy\_s(url, L"http://中文");

wcscat\_s(url, mgrIp);

wcscat\_s(url, L":");

wchar\_t szPort[20] = { 0 };

\_ltow\_s(mgrPort, szPort, 10);

wcscat\_s(url, szPort);

return url;

}

DLL\_API const char\* getUrl(const char\* mgrIp, long mgrPort, long materialId)

{

static char url[260] = { 0 };

strcpy\_s(url, "http://中文");

strcat\_s(url, mgrIp);

strcat\_s(url, ":");

char szPort[20] = { 0 };

\_ltoa\_s(mgrPort, szPort, 10);

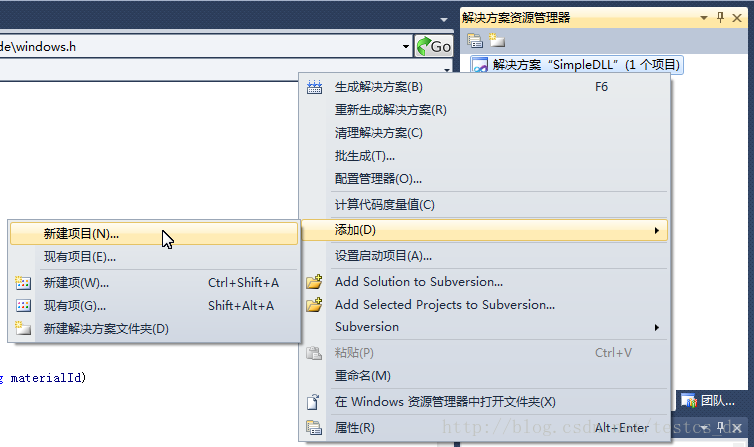
strcat\_s(url, szPort);

return url;

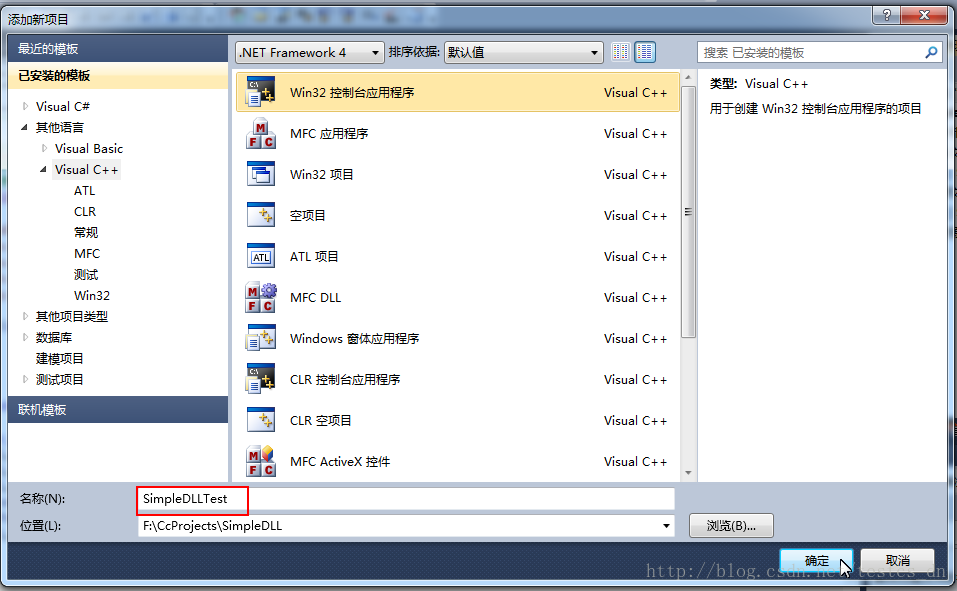
}

创建引用动态链接库的应用程序：

1、在解决方案上单击鼠标右键->添加->新建项目。

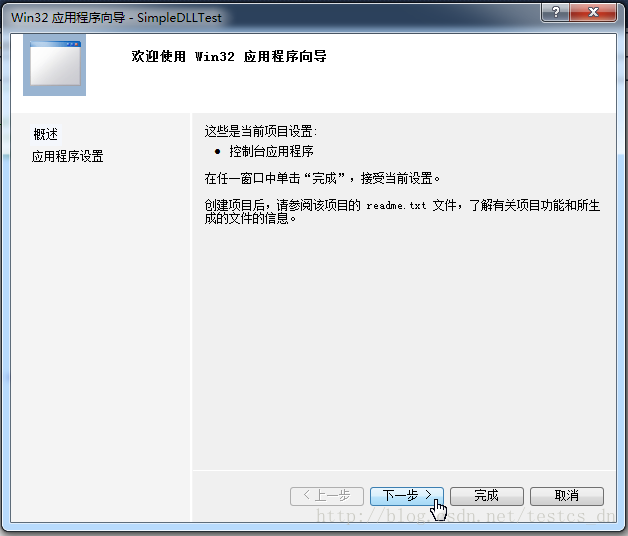


2、在添加新项目中选择其它语言->Visual C++->Win32。



3、选择Win32 控制台应用程序，设置名称：simpleDLLTest。

4、单击确定，在出现的Win32 应用程序向导的概述对话框中点击下一步。



5、在应用程序设置中，选择应用程序类型下的控制台应用程序。



6、单击完成创建项目。

在控制台应用程序中使用类库的功能：

1、为SimpleDLLTest.cpp添加内容。如下所示：

// SimpleDLLTest.cpp : 定义控制台应用程序的入口点。

#include "stdafx.h"

#include "../SimpleDLL/SimpleDLL.h" //添加头文件引用

#pragma comment(lib, "..\\..\\SimpleDLL\\Release\\SimpleDLL.lib") //添加lib文件引用

#include <process.h>

#include <locale.h>

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "chs"); //配置地域化信息为简体中文，否则打印出来的中文是乱码

wprintf(L"getPlayUrl: %s\r\n", getPlayUrl(L"127.0.0.1", 10087, 1));

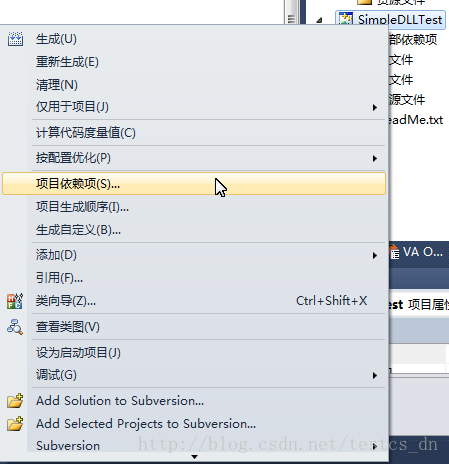
printf("getUrl: %s\r\n", getUrl("127.0.0.1", 10087, 1));

system("pause");

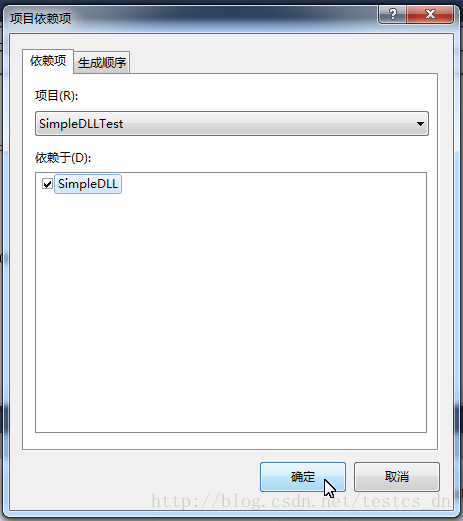
return 0;

}

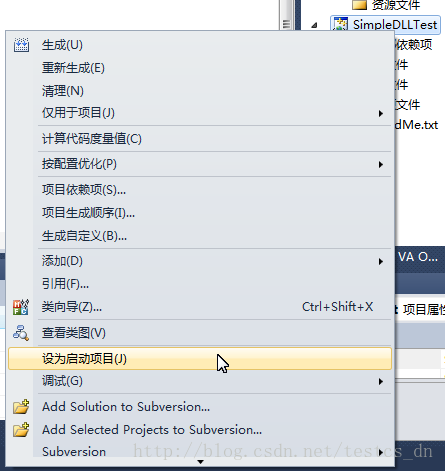
2、引用simpleDLL项目。右键单击SimpleDLLTest项目，选择项目依赖项。



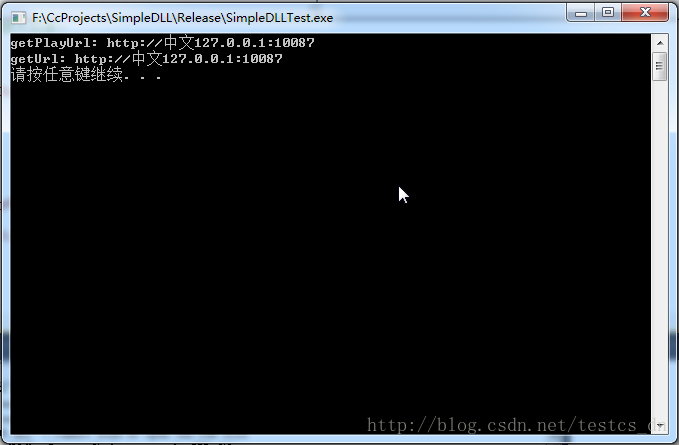
3、依赖于窗口中勾选SimpleDLL，单击确定。



4、设置SimpleDLLTest项目为活动项目。右键单击SimpleDLLTest项目，选择设为启动项目。

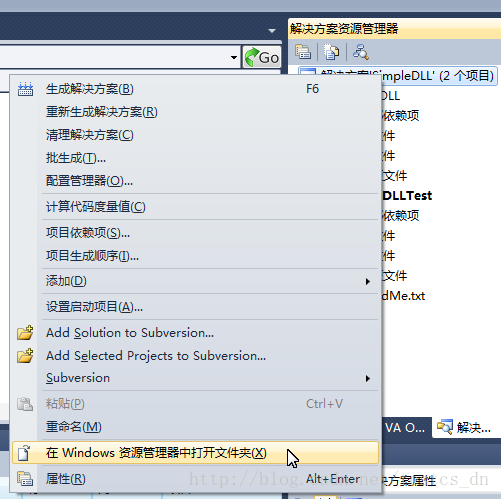


6、生成解决方案。Debug运行结果如下：

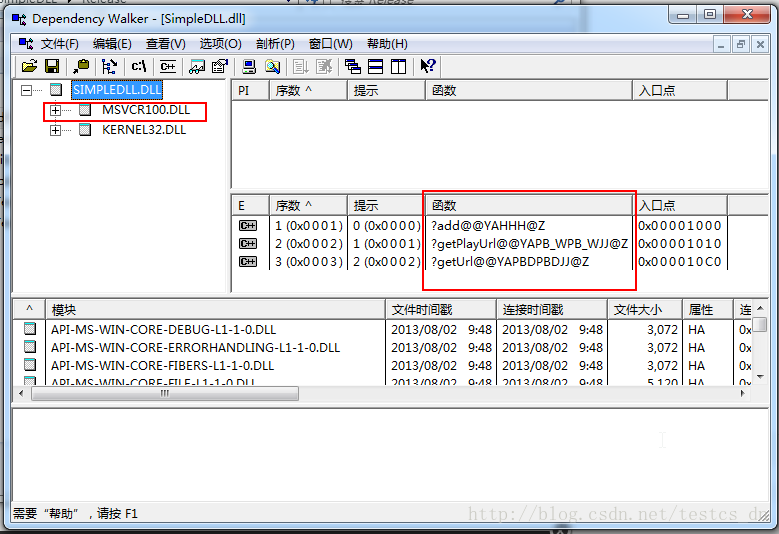


注意：现在创建的DLL只能由c++调用，C语言等其它语言是调用不了的！

我们来用工具看一下，在解决方案上单击鼠标右键，在Windows资源管理器中打开文件夹

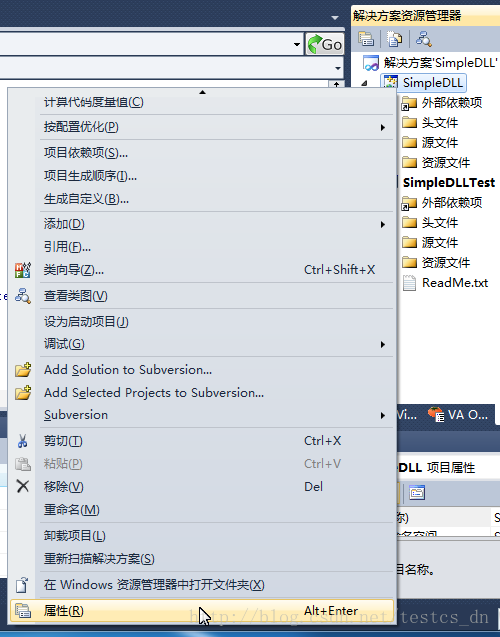


我编译的是Release版本，所以打开Release文件夹，找到SimpleDLL.dll文件，用Depends工具打开它

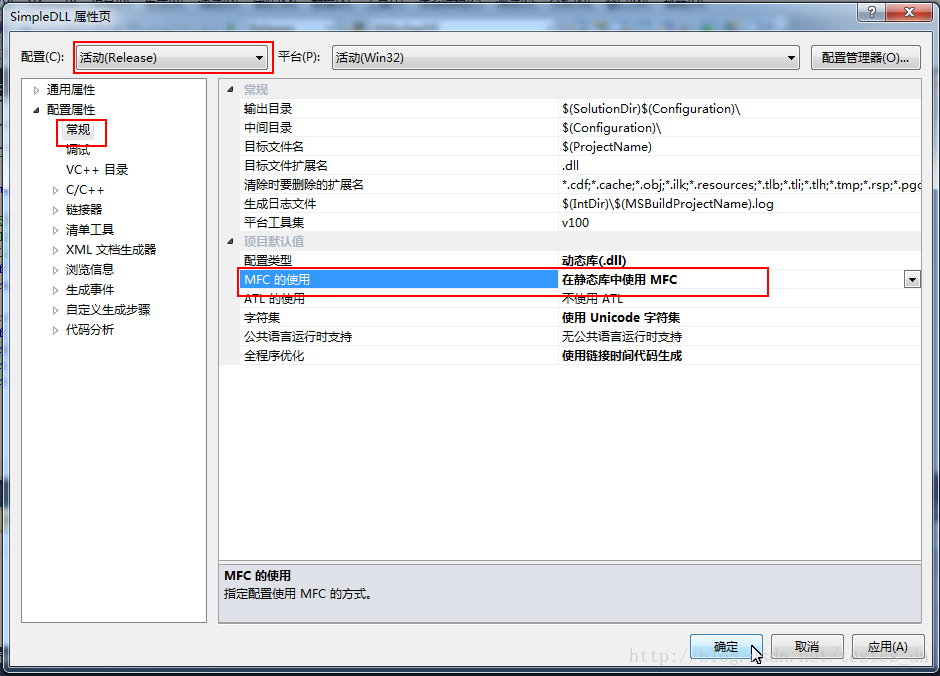


目前编译的版本对MSVCR100.DLL还有依赖，这样复制到其它没有安装VS2010的电脑上是用不了的；

在项目上单击鼠标右键，属性：



在左侧的配置属性中选择常规，在右侧找到MFC的使用，选择在静态库中使用MFC



这里是修改MFC的使用，还有另一种方法：vs2010发布时去除msvcp100.dll和msvcr100.dll图解说明

再看函数，带有一些特殊字符，这样c++之外其它语言是调用不了的；我们需要在头文件的函数声明中添加extern "C"标志，代码如下：

//------------------ SimpleDLL.h ----------------

#pragma once;

//该宏完成在dll项目内部使用\_\_declspec(dllexport)导出

//在dll项目外部使用时，用\_\_declspec(dllimport)导入

//宏DLL\_IMPLEMENT在SimpleDLL.cpp中定义

#ifdef DLL\_IMPLEMENT

#define DLL\_API \_\_declspec(dllexport)

#else

#define DLL\_API \_\_declspec(dllimport)

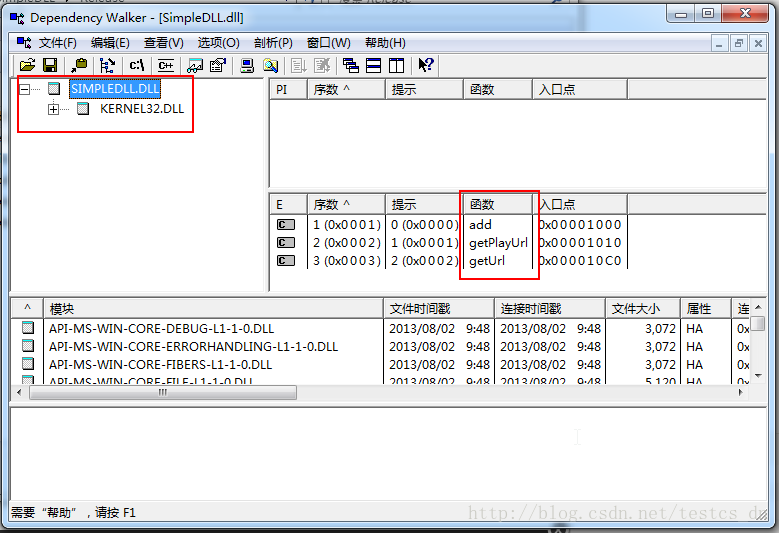
#endif

extern "C" DLL\_API int add(int x, int y); //简单方法

extern "C" DLL\_API const wchar\_t\* getPlayUrl(const wchar\_t\* mgrIp, long mgrPort, long materialId);

extern "C" DLL\_API const char\* getUrl(const char\* mgrIp, long mgrPort, long materialId);

重新编译，再用Depends工具打开它，可以发现依赖项已经没有了，函数名称也正常了：



参考：演练：创建和使用动态链接库 (C++)

解决方案源码下载：http://download.csdn.net/detail/testcs\_dn/7411383

---------------------

作者：微wx笑

来源：CSDN

原文：https://blog.csdn.net/testcs\_dn/article/details/27237509

版权声明：本文为博主原创文章，转载请附上博文链接！

project dependency

多个工程时， 会先编译依赖的工程，同时会自动link依赖工程生成的lib文件. 会根据依赖关系来顺序编译的.