[**Java通过JNI调用C++程序**](http://www.cnblogs.com/liuling/p/2013-12-20.html)

　　JNI是Java Native Interface的缩写，中文为JAVA本地调用。使用JNI可以很方便的用我们的Java程序调用C/C++程序。很多时候，某些功能用Java无法实现，比如说涉及到底层驱动的一些功能，这时候我们就可以利用JNI来调用C或者C++程序来实现，这就是JNI的强大之处。但是JNI也有它的缺点，使用java与本地已编译的代码交互，通常会丧失平台可移植性。

　　下面是一个JNI例子，调用C++输出"hello world"：

第一步：创建Java类，在里面定义一个本地方法（用native关键字修饰的方法）

　　public native void sayHello();

第二步：使用javah命令（javah  类的全路径）生成本地方法的C++头文件

　　在DOS窗口中进入工程所在目录，然后执行javah com.test.TestNative命令，执行完之后就会在当前目录生成一个后缀名为.h的头文件，如com\_test\_TestNative.h，这个头文件是根据包名和类名来命名的。

[复制代码](javascript:void(0);)

1 /\* DO NOT EDIT THIS FILE - it is machine generated \*/

2 #include <jni.h>

3 /\* Header for class com\_test\_TestNative \*/

4

5 #ifndef \_Included\_com\_test\_TestNative

6 #define \_Included\_com\_test\_TestNative

7 #ifdef \_\_cplusplus

8 extern "C" {

9 #endif

10 /\*

11 \* Class: com\_test\_TestNative

12 \* Method: sayHello

13 \* Signature: ()V

14 \*/

15 JNIEXPORT void JNICALL Java\_com\_test\_TestNative\_sayHello

16 (JNIEnv \*, jobject);

17

18 #ifdef \_\_cplusplus

19 }

20 #endif

21 #endif

[复制代码](javascript:void(0);)

　　15、16行是对TestNative类中的本地方法sayHello()的声明。这个h文件相当于我们在java里面的接口，这里声明了一个 Java\_com\_test\_TestNative\_sayHello (JNIEnv \*, jobject);方法，然后在我们的本地方法里面实现这个方法，也就是说我们在编写C/C++程序的时候所使用的方法名必须和这里的一致。

第三步：编写C/C++本地代码，生成动态链接库文件

　　首先在VC6.0（当然也可以用其他工具）中创建一个dll工程---Win32 Dynamic-Link Library工程。然后将上面生成的头文件com\_test\_TestNative.h添加到该工程中，然后创建一个源文件引用该头文件并且实现头文件中本地函数的功能：

[复制代码](javascript:void(0);)

1 #include<iostream.h>

2 #include"com\_test\_TestNative.h"

3

4 JNIEXPORT void JNICALL Java\_com\_test\_TestNative\_sayHello(JNIEnv \*env, jobject obj)

5 {

6 cout<<"hello world!"<<endl;

7 }

[复制代码](javascript:void(0);)

　　这里因为com\_test\_TestNative.h中引入了jni.h所以要将jni.h加入到VC6.0安装目录下的Include目录中。jni.h在JDK安装目录下的include中，同时得件include/win32中的两个头文件jawt\_md.h、jni\_md.h也导入到VC6.0中。

　　将所依赖的头文件导入之后，我们就可以构建该工程了，按F7就行了，完了会在工程目录中的Degug目录下生成一个动态链接库文件，我这里生成的是NativeCode.dll。我们就可以将该dll文件拷贝到环境变量path所包含的目录下给咱们的Java程序调用了，为了方便，我们也可以将dll所在的工程目录加入到环境变量path中去，这样可以避免每次都要拷贝的麻烦。注意修改环境变量之后要重启myeclipse。

第四步：Java调用本地函数

[复制代码](javascript:void(0);)

1 package com.test;

2

3 public class TestNative {

4 public native void sayHello();

5

6 /\*\*

7 \* @param args

8 \*/

9 public static void main(String[] args) {

10 System.loadLibrary("NativeCode");

11 TestNative tNative = new TestNative();

12 tNative.sayHello();

13 }

14 }

[复制代码](javascript:void(0);)

　　第10行是加载动态链接库，JVM只需要加载一次就可以调用了，“NativeCode”是上面生成的动态链接库的名字，不含后缀名。

　　运行该程序，成功打印输出了"hello world"。