Java 通过 JNI 调用 C++程序

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。

```
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
```

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：

```
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     }
```

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

```
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 }
```

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"。