



# JNI 简介与实现

作者：牛地印章

转载于：<http://blog.chinaunix.net/u3/90973/index.html>

## 一 JNI 简介

JNI (Java Native Interface), 即 Java 本地接口, 是为 java 编写本地方法和 jvm 嵌入本地应用程序的标准的应用程序接口。首要的目标是在给定的平台上采用 JAVA 通过 JNI 调用本地方法, 而本地方法是以库文件的形式存放的 (在 WINDOWS 平台上是 DLL 文件形式, 在 UNIX 机器上是 SO 文件形式)。通过调用本地的库文件的内部方法, 使 JAVA 可以实现和本地机器的紧密联系, 调用系统级的各接口方法。有的 jvm 来实现兼容的二进制编码本地方法库。

## 二 环境介绍

Java Development Kit (JDK) 版本 1.6.0,  
Microsoft Visual Studio2005 编译器编译生成 C 语言的动态链接库 dll。

## 一 JNI 简介

JNI (Java Native Interface), 即 Java 本地接口, 是为 java 编写本地方法和 jvm 嵌入本地应用程序的标准的应用程序接口。首要的目标是在给定的平台上采用 JAVA 通过 JNI 调用本地方法, 而本地方法是以库文件的形式存放的 (在 WINDOWS 平台上是 DLL 文件形式, 在 UNIX 机器上是 SO 文件形式)。通过调用本地的库文件的内部方法, 使 JAVA 可以实现和本地机器的紧密联系, 调用系统级的各接口方法。有的 jvm 来实现兼容的二进制编码本地方法库。

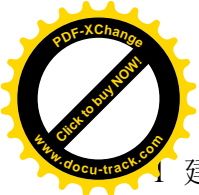
## 二 环境介绍

Java Development Kit (JDK) 版本 1.6.0,  
Microsoft Visual Studio2005 编译器编译生成 C 语言的动态链接库 dll。

## 三 简叙步骤

- 编写带有 native 声明的方法的 java 类
- 使用 javac 命令编译所编写的 java 类
- 使用 javah java 类名生成扩展名为 h 的头文件
- 使用 C/C++ 实现本地方法, 并生成动态链接库
- 将库文件拷贝到 java 工程目录下, 运行 java 程序

## 四 具体实现



## 1 建立 java 工程

在 JAVA 程序中，首先需要在类中声明所调用的库名称，如下：

```
static {  
    System.loadLibrary("dllname");  
}
```

库的扩展名字可以不用写出来，究竟是 DLL 还是 SO，由系统自己判断。接着对将要调用的方法做本地声明，关键字为 `native`。并且只需要声明，而不需要具体实现。如下：

```
public native static void fn1(int i);  
public native static int  fn2(void );
```

然后编译该 JAVA 程序文件，生成 CLASS，再用 `JAVAH` 命令，JNI 就会生成 C/C++ 的头文件。

例如建立 java 文件 `TestDel.java`，内容为：

```
public class TestDel  
{  
    static  
    {  
        System.loadLibrary("cjw"); //库名 cjw  
    }  
  
    public native static void creFolder();  
    public native static void delFolder1();  
    public native static void delFolder2();  
    public static void main(String[] args)  
    {  
        TestDel test = new TestDel();  
  
        System.out.println("start create  Folder...");  
        test.creFolder();  
        System.out.println("create Folder finished.");  
  
        long stime = System.currentTimeMillis();  
        //test.delFolder1();  
        test.delFolder2();  
        long etime =System.currentTimeMillis();  
  
        System.out.println(etime-stime);  
    }  
}
```

用 `javac TestDel.java` 编译它，生成 `TestDel.class`。

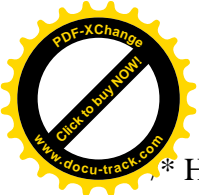
再用 `javah TestDel`，则会在当前目录下生成 `TestDel.h` 文件，这个文件需要被 C/C++ 程序调用来生成所需的库文件。

## 2 生成动态链接库

对于已生成的.h 头文件，C/C++ 所需要做的，就是把它的各个方法具体的实现。然后编译连接成库文件。

接上例子。我们先看一下 `TestDel.h` 文件的内容：

```
/* DO NOT EDIT THIS FILE - it is machine generated */  
#include <jni.h>
```



```
/* Header for class TestDel */
#ifndef _Included_TestDel
#define _Included_TestDel
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:      TestDel
 * Method:     creFolder
 * Signature:  ()V
 */
JNIEXPORT void JNICALL Java_TestDel_creFolder
(JNIEnv *, jclass);
/*
 * Class:      TestDel
 * Method:     delFolder1
 * Signature:  ()V
 */
JNIEXPORT void JNICALL Java_TestDel_delFolder1
(JNIEnv *, jclass);
/*
 * Class:      TestDel
 * Method:     delFolder2
 * Signature:  ()V
 */
JNIEXPORT void JNICALL Java_TestDel_delFolder2
(JNIEnv *, jclass);
#ifdef __cplusplus
}
#endif
#endif
```

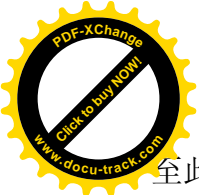
在具体实现的时候，我们只关心几个函数原型：

```
JNIEXPORT void JNICALL Java_TestDel_creFolder
(JNIEnv *, jclass);
JNIEXPORT void JNICALL Java_TestDel_delFolder1
(JNIEnv *, jclass);
JNIEXPORT void JNICALL Java_TestDel_delFolder2
(JNIEnv *, jclass);
```

这里 **JNIEXPORT** 和 **JNICALL** 都是 JNI 的关键字，表示此函数是要被 JNI 调用的。如果 java 中声明的函数原型包含参数，例如 **int**，那么生成的头文件中变成 **jint**，它是以 JNI 为中介使 JAVA 的 **int** 类型与本地的 **int** 沟通的一种类型，我们可以视而不见，就当做 **int** 使用。其它参数类似。函数的名称是 **JAVA\_** 再加上 java 程序的 **package** 路径再加函数名组成的。参数中，我们也只需要关心在 JAVA 程序中存在的参数，至于 **JNIEnv\*** 和 **jclass** 我们一般没有必要去碰它。

好，下面我们使用 **dll** 具体实现这几个函数（一个函数是创建目录，另两个是两种删除目录的方法）。

注意，本例是通过 **VS2005** 建立 **dll** 的，需要安装 "**WebDeploymentSetup.msi**"，New 一个工程选择 **VC++ 建立 win32 project**，选中 **dll 建立 Empty project**。将 java 中生成的头文件拷贝

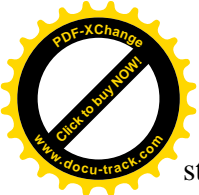


至此工程添加，再建立新文件 test.cpp(必须以 cpp 后缀，若以 c 后缀则 builder 会出错)test.cpp 代码如下（注意这句#include "TestDel.h"）:

```
#include <stdio.h>
#include <direct.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include "TestDel.h"

JNIEXPORT void JNICALL Java_TestDel_creFolder(JNIEnv *, jclass)
{
    int i,j;
    char buf1[20];
    char buf2[20]="./Folder/";
    char buf3[20];
    mkdir("Folder");
    for(i=0;i<100;i++)
    {
        itoa(i,buf1,10);
        mkdir(strcat(buf2,buf1));
        strcpy(buf3,strcat(buf2,"\\"));
        for(j=0;j<100;j++)
        {
            itoa(j,buf1,10);
            mkdir(strcat(buf2,buf1));
            strcpy(buf2,buf3);
        }
        strcpy(buf2,"./Folder/");
    }
}

JNIEXPORT void JNICALL Java_TestDel_delFolder1
(JNIEnv *, jclass)
{
    int i,j;
    char buf1[20];
    char buf2[20]=".\\Folder\\";
    char buf3[20];
    for(i=0;i<100;i++)
    {
        itoa(i,buf1,10);
        strcat(buf1,"\\");
        strcpy(buf3,strcat(buf2,buf1));
        for(j=0;j<1000;j++)
        {
            itoa(j,buf1,10);
            rmdir(strcat(buf2,buf1));
            strcpy(buf2,buf3);
        }
        rmdir(buf3);
    }
}
```



```
strcpy(buf2, ".\\Folder\\");
}

}
JNIEXPORT void JNICALL Java_TestDel_delFolder2
(JNIEnv *, jclass)
{
    char    cmd[50];
    strcpy(cmd, "rmdir /s/q ");
    strcat(cmd, "Folder");
    system(cmd);
}
}
```

注 意：一定要把 SDK 中的 include 文 件夹中(和它下面的 win32 文件夹下的头文件)的几个头文件拷贝到 VC 的 include 文件夹中。或者在 VS 的 tools\options -> Projects and Solutions\VC++ Project Settings, 修改 directories 中 include 的设置，把头文件../sdk1.5.0/include 和 ../sdk1.5.0/include/win32 给包含进来。

编译连接成库文件 dll(在 debug 文件夹中)，注意名称要与 JAVA 中需要调用库名的一致，这里可以修改为 cjw.dll。

### 3 运行程序

把 cjw.dll 拷贝到 TestDel.class 的目录下，java TestDel 运行它，就可以观察到结果了。这里实现的功能是统计删除文件夹的时间。