DOI: http://dx.doi.org/10.21501/21454086.2344

Lámpsakos | No.19 | pp. 73-76| enero-junio | 2018 | ISSN: 2145-4086 | Medellín–Colombia

Guía para la configuración de la Interfaz Nativa de Java

Guide for configuration the Java Native Interface

Mailyn Moreno Espino, PhD.
Facultad de Ingeniería Informática
Universidad Tecnológica de La Habana
José Antonio Echeverría, CUJAE
La Habana, Cuba
my@ceis.cujae.edu.cu

Raymel Ramos Guerra, Esp.
Universidad Tecnológica de La Habana
José Antonio Echeverría, CUJAE
La Habana, Cuba
rramosg@ceis.cujae.edu.cu

Yilian Bacallao Leiva, Ing.
Universidad Tecnológica de La Habana
José Antonio Echeverría, CUJAE
La Habana, Cuba
ybacallao@ceis.cujae.edu.cu

(Recibido el 19-07-2017, Aprobado el 31-10-2017, Publicado el 16-01-2018)

Estilo de Citación de Artículo:
M. Moreno, R. Ramos, Y. Bacallao, "Guía para la configuración de la Interfaz Nativa de Java", Lámpsakos, no. 19, pp 73-76, 2018
DOI: http://dx.doi.org/10.21501/21454086.2344

Resumen: En la actualidad existen diversos lenguajes de programación y es necesario contar con una herramienta que permita la integración entre ellos. Java provee de un framework llamado: interfaz nativa de Java, JNI por sus siglas en inglés; para permitir que se puedan escribir programas en otros lenguajes diferentes a Java y mantener la portabilidad entre todas las plataformas. Permite que el código que se ejecuta en la máquina virtual de Java pueda interactuar con aplicaciones y bibliotecas escritas en otros lenguajes, como C, C++ y ensamblador. Esta guía se basa en un ejemplo básico de una aplicación *"Hello Word"*, la cual servirá para de manera sencilla, mostrar los pasos a seguir para la configuración de JNI en los entornos de desarrollo involucrados, que para esta guía se utilizó, para Java: Eclipse y para C++: Visual Studio 2010.

Palabras clave: Eclipse, Framework, JNI, Máquina virtual de Java, Visual Studio.

Abstract: Currently there are several programming languages and it is necessary to have a tool that allows integration between them. Java provides a framework called: Java native interface, JNI; to allow it to be written in languages other than Java and maintain portability across all platforms. It allows the code running on the Java virtual machine to interact with applications and libraries written in other languages, such as C, C ++ and assembler. This manual is based on a basic example of a "Hello Word" application, which serves the simple way, shows the steps to follow for the JNI configuration in the development environments involved, which for this guide were, for Java: Eclipse and for C ++: Visual Studio 2010.

Keywords: Eclipse, Framework, JNI, Java virtual machine, Visual Studio

1 INTRODUCCIÓN

En ocasiones surge la necesidad de integrar diferentes lenguajes de programación debido a las facilidades y las particularidades que brindan cada uno de ellos.

Actualmente existen diversos lenguajes de programación y cada uno cumple un objetivo; entre ellos se encuentra Java. Java es un lenguaje de programación orientado a objetos, que permite al usuario abstraerse del trabajo con la memoria, lo cual en ocasiones es un inconveniente [1].

Existen diferentes herramientas que permiten la integración entre lenguajes de programación una de ellas es el framework JNI. JNI permite al código Java que se ejecuta dentro de la máquina virtual de Java interactuar con aplicaciones y librerías escritas en otros lenguajes, como C/C++ o incluso en lenguaje ensamblador [2]. Incorpora a su vez las herramientas para ejecutar código Java desde

Mailyn Moreno-Espino, PhD, Raymel Ramos-Guerra
Yilian Bacallao-Leiva
DOI: http://dx.doi.org/10.21501/21454086.2344

aplicaciones desarrolladas en otros lenguajes [2]. El código nativo son funciones escritas en un lenguaje de programación como C o C++ para un sistema operativo donde se está ejecutando la máquina virtual [3]. La máquina virtual es un entorno de programación formado por el programa que ejecuta los *bytecodes* Java y las librerías (API) de Java [3].

Los entornos de desarrollo utilizados fueron: para Java Eclipse y para C++ Visual Studio. Está guía expone de manera sencilla los pasos a seguir para configurar JNI en la computadora.

2 DESARROLLO

La configuración de JNI en la herramienta de desarrollo que se vaya a utilizar resulta en ocasiones algo engorrosa y de la cual hay poca información disponible. En esta guía se expone mediante un ejemplo sencillo de una aplicación "Hello Word" los pasos a seguir para la configuración de las herramientas Eclipse y Visual Studio. No es objetivo de este trabajo hacer un resumen de JNI ni de las ventajas y desventajas que este ofrece, ya que solo se enfoca en la configuración de los Entornos de desarrollo integrado IDEs, antes mencionado. Esta guía ha sido confeccionada partiendo de que se tiene un mínimo de conocimiento en estas herramientas de desarrollo.

2.1 Aspectos fundamentales

Para la elaboración de la guía se escogieron los IDEs de desarrollo Eclipse para el lenguaje de JAVA y Visual Studio 2010 para C++. Aunque para esta guía se usó Visual Studio 2010 cabe destacar que estos mismos pasos fueron probados para las versiones 2012 y 2013.

Esta guía presenta la implementación de un ejemplo básico "Hello Word". Este ejemplo consta de 2 aplicaciones una llamada **testJNI** implementada en JAVA la cual se subdivide en un paquete llamado **test** el cual contiene la clase **HelloWord** que va a ser la encargada de invocar el método nativo implementado en C++ y finalmente imprimir en la consola el cartel "*Hola mundo con JNI".* La segunda aplicación llamada **helloWordJNI** está implementada en C++**,** esta va a contener el fichero **helloWordJNI.cpp** el cual tendrá la implementación del algoritmo que devuelve la frase "*Hola mundo con JNI".*

2.2 Pasos para la configuración del Eclipse

En este paso se harán las configuraciones necesarias en el IDE Eclipse, el cual emplearemos para el lenguaje Java.

1) Crear nuevo proyecto

Crear un nuevo proyecto en Eclipse y nombrarlo **testJNI**, posteriormente agregarle un paquete llamado **test** y dentro de este paquete agregarle una clase llamada **HelloWord**.

2) Invocación al método nativo

A la clase **HelloWord** añadirle el siguiente fragmento de código:

```
package test;
public class HelloWord {
 public native static String
 helloWord();  1
 static{
    System.loadLibrary("helloWordJNI");
    2
 }
 public static void main(String[] args)
 {
    System.out.println(helloWord());
 }
}
```

En la sentencia **1** es en donde se hace la invocación al método nativo. Con **2** se carga la dll que contiene el método nativo implementado en C++.

3) Compilar la clase

Al compilar la aplicación, el Eclipse debe lanzar una excepción del tipo UnsatisfiedLinkError. Este error se debe a que el compilador no puede cargar la Librería de enlace dinámico, dll con el método nativo. Es necesario ejecutar este paso ya que al compilar la aplicación se crea el fichero *.class* necesario para poder generar el encabezado.

4) Generar el encabezado(.h) de la función nativa

Ir a **Run configuration ->External Tools Configurations** y crear una nueva configuración.
En *name* se pone nombre cualquiera (el que desee el programador). Para este ejemplo se decidió utilizar **configuration**. En *Location* se pone la dirección donde se tenga instalado la máquina virtual de JAVA, se localiza el fichero **java.h** ubicado dentro de carpeta **bin**; este fichero es el encargado de generar un encabezado .h con la información necesaria que va a necesitar el método nativo que va a ser implementado posteriormente en C++. En *WorkingDirectory* se especifica donde se quiere que

se genere nuestro encabezado que debe ser dentro de la carpeta **bin** del proyecto.

En *Arguments* se escribe:

–jni
nombre_del_paquete.nombre_del_fichero_.class.

Finalmente se da click en el botón *Run.* Lo próximo es ir a la ubicación del proyecto y ver si se generó correctamente dentro de la carpeta **bin** un fichero **.h**.

5) Especificarle al Eclipse donde van a estar nuestras librerías nativas

Se hace *Click* derecho sobre el proyecto y se selecciona *Properties.* En la pestaña *Libreries* desplegar la librería, seleccionar *Nativelibrarylocation* y hacer click en el botón Edit.

En el botón *Workspace* se selecciona la ruta donde se encontrará la librería .dll, y posteriormente se hace *Click* en OK.

Con estos simples pasos ya se tiene preparado el Eclipse para que reconozca nuestras librerías nativas.

2.3 Pasos para la configuración del Visual Studio

En este paso se harán las configuraciones necesarias en el IDE Visual Studio, el cual se empleará para el lenguaje C++.

1) Crear un nuevo proyecto

Crear un nuevo en Visual Studio y ponerle el nombre de **helloWordJNI**, posteriormente agregarle al archivo **helloWordJNI.cpp** el siguiente código:

```
#include"stdafx.h"
usingnamespace::std;
JNIEXPORT jstring JNICALL
Java_test_HelloWord_helloWord
(JNIEnv * env, jobject obj){
  string name="Hola mundo con JNI";
  jstring js=  env-
>NewStringUTF(name.c_str());
  return js;
}
```

Al fichero **stdafx.h** agregarle el siguiente código:

```
#pragmaonce
#include"targetver.h"
#define WIN32_LEAN_AND_MEAN
#include<windows.h>
#include"test_HelloWord.h"
#include<jni.h>
#include<string>
```

2) Configurar Visual Studio para que reconozca las librerías JNI

Hacer click derecho sobre el proyecto y seleccionar *properties.* En la ventana *helloWordJNIPropertyPages,* desplegar *C/C++*y seleccionar *General*, en *AdditionalIncludeDirectories*hacer click, desplegar y seleccioanr*<Edit…>.*

Hacer click en el icono  y agregar:

- Ruta del encabezado generado desde el Eclipse.
- Incluir las rutas en estos directorios es en donde se encuentra la información relacionada con JNI:

  C:\Program Files\Java\jdk1.8.0\include\win32
  C:\Program Files\Java\jdk1.8.0\include..

Hacer click en *OK* y luego en *Aceptar.*

3) Generar la librería con el código nativo

Hacer click en *startdebugging* o presionar la tecla F5.

4) Agregar la librería de código nativo al proyecto implementado en JAVA

Ir a la ubicación del Workspace de Visual Studio. Ir a la ruta *Projects\helloWordJNI\Debug*y copiar el archivo .dll generado, en este caso **helloWordJNI.dll**.

Ir a la ubicación del Workspace del Eclipse. Ir a la ruta **testJNI*bin*** *y pegar el archivo .dll copiado.*

2.4 Validación de la aplicación

En el Eclipse hacer click en *RunHelloWord* o presionar Ctrl+F11. Se debe mostrar en la consola de JAVA la siguiente frase: “Hola mundo con JNI”.

DOI: http://dx.doi.org/10.21501/21454086.2344

3 TRABAJOS FUTUROS

Actualmente se está desarrollando un sistema multi-agente en JADE para la verificación de integridad de procesos en ejecución en los sistemas operativos Windows y Linux. Los agentes inteligentes están creados dentro de la plataforma JADE la cual está implementada en JAVA y debido a que este lenguaje te abstrae del trabajo con la memoria se hizo imprescindible disponer de otro lenguaje que si lo permitiera, el cual fue C++. De esta necesidad surge el objetivo de esta guía: configurar y aprender a utilizar JNI.

4 CONCLUSIONES

Al término de esta guía se puede concluir lo siguiente:

- El framework JNI es de gran ayuda a todo aquel que quiere integrar código nativo a sus aplicaciones creadas en JAVA.
- La configuración de JNI en los IDEs de desarrollo no son muy complejas.

AGRADECIMIENTOS

Los autores agradecen a la Dra. C. Maylin Moreno Espino por los aportes a esta investigación. A la Universidad Tecnológica de la Habana “José Antonio Echeverría” CUJAE. A la Facultad de Ingeniería Informática de la CUJAE.

REFERENCIAS

[1] P. Deitel,H. Deitel, “Java How to Program”: Prentice Hall Press, 2011.

[2] O. B. Fernández, Introducción al lenguaje de programación Java.: Una guía básica vol. 9, 2005.

[3] R. McNab and F. Howell, "Using java for discrete event simulation," in Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop, 1996, pp. 219-228.

[4] S. Stricker. (2002, Java programming with JNI. Developer works, 22.

[5] D. G. Javier and R. J. Ignacio, "Aitor, Imaz," Aprenda Java como si estuviera en primero. Escuela Superior de Ingenieros Industriales. San Sebastian, 2000.

[6] S. Liang, The Java Native Interface: Programmer's Guide and Specification: Addison-Wesley Professional, 1999.

[7] L. F. Evgeniy Gabrilovich, "JNI – C++ integration made easy", 2007.