Symantec Visual Cafe (Java 1.1) and Microsoft Visual C++ 5.0

A COMPLETE step-by-step guide to creating a simple example program.

This guide will describe precisely the steps needed to connect up a native Java call to a C dll generated using Microsoft Visual C++ 5.0. Try this example, then you can pattern-match to your heart's content!

1. Start Visual Cafe. Go to the menubar and select File/New Project. Double-click Basic Applet
2. Select File/Save as and save the project as sample1 (in its own new directory)
3. Select Insert/Form and double-click Frame.
4. Now go to the objects window on the left and change the name of this new object from Frame1 to NativeClass.
5. Now double-click on Applet1 and double-click on the visual window to get to the code.
6. Right after the class declaration, put the line: NativeClass myclass; So, the resulting few lines around this declaration will look like:
7.

```
public class Applet1 extends Applet
{
    NativeClass myclass;
      public void init()
```

8. Now cause an instance to be created in the init function. Put the line, `myclass = new NativeClass();` in the `init()` function, at the end. The surrounding lines will look like:

```
    //{{INIT_CONTROLS
    setLayout(null);
    setSize(426,266);
    //}}
    myclass = new NativeClass();
}

//{{DECLARE_CONTROLS
```

9. Now double-click on the NativeClass object to get to the Form Designer. Now we will add a button and a textfield to this form. The button and textfield icons are under the Standard tab above your work area. Click on the button (far right) and then drag a button into existence on the form. Click on the textfield (6^{th} from the left) and drag a textfield in the form as well.

10. Now double-click no the background of the Form Designer to get to the code. In the NativeClass constructor, add `setVisible(true);` The surrounding will look like:

```
add(textField1);
setTitle("Untitled");
//}}
setVisible(true);
```

```
//{{INIT_MENUS
//}}
```

11. Now select Project/Execute and you should get the Applet window and, on top, the untitled window with a button and a textfield. Exit this execution by clicking the X at the top right of the Applet window.

12. From the top left Objects selection on the NativeClass.java code select `button1`. Then, to its right, from Event/Methods select `MouseClick`.

13. Inside the `button1_MouseClick` function add lines to result in the following function definition. The function addone is going to be written in C:

```
void button1_MouseClick(java.awt.event.MouseEvent
event)
{
   // to do: code goes here.
   int returnval;
   returnval = addone(5);
   returnval = returnval + 1;
   textField1.setText(Integer.toString(returnval));
}
```

14 After this function definition, but before the final curly brackets of the class definition, add:

```
public native int addone(int num);

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

These lines declare that addone will be a native call and the static paragraph loads the .dll file at runtime. Now select Project/Execute. The Applet will appear and then you'll get an error on the outputs window: `java.lang.NoClassDefFoundError`. Quit the program, same as before.

15 Now you need to generate the C header (.h) file based on the Java code you wrote. Go into MS-DOS and get to the directory with your Java code in it. Make sure you see NativeClass.java. Now run javah to create a JNI header file by typing:

```
javah -jni NativeClass
```

16. Now do "`type NativeClass.h`" and you should see this:

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

#ifndef _Included_NativeClass
#define _Included_NativeClass
```

```
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     NativeClass
 * Method:    addone
 * Signature: (I)I
 */
JNIEXPORT jint JNICALL Java_NativeClass_addone
  (JNIEnv *, jobject, jint);

#ifdef __cplusplus
}
#endif
#endif
```

If for some reason javah was unrecognized, note that it's in your `?:\VisualCafedbDE\Java\Bin` directory and you should have that in your path for executables. The other MS-Dos executable we'll be using also are there. By the way, I'm using Visual Cafe database edition (that's why there's the `dbDE` on the folder name).

17 Now we can do the C side! Start up Microsoft Visual C++ 5.0. On the menubar, click `File/New`. Select `Win32 Dynamic-Link Library`. Give the project the name, `csample` and click OK. Now make sure you're in File View on the left side tab.

18 Click `File/New` again. Select `Text File`, name it `csample.c` and click OK. Now, inside csample.c, put the following two lines:

```
#include <jni.h>
#include "NativeClass.h"
```

19 Now click `Project/Settings`. Go to the C/C++ tab. Select `Category: Preprocessor`. In `Additional include directories:` put the following:

```
D:\VisualCafedbDE\Java\Include,D:\VisualCafedbDE\Java\Include\win32
```

(in my case, Visual Cafe is installed on the D drive)

20 Now go to your Java project directory and grab NativeClass.h. Put it in your csample directory.

21 Back to Visual C++ 5.0. Open NativeClass.h in here, and copy and paste the function declaration into csample.c. Type lines into csample.c to send up with the following code:

```
#include <jni.h>
#include "NativeClass.h"

JNIEXPORT jint JNICALL Java_NativeClass_addone
  (JNIEnv * jnienv, jobject javaobj, jint parm1)
{
```

```
    int n;
    n = parm1 + 1;
    return (n);
}
```

22 Now make your dll. Select Build/Buildcsample.dll. It generate a .lib that is useless and the .dll file in the Debug (or Release) subdirectory of your project.

23 Copy csample.dll from the C side to your Java project directory.

24 Start up MS-DOS. Go to your Java project directory. Type: "`implib`." For the lib file, specify `csample.lib`. For the dll, specify `csample.dll`. When it's done, then type "`libunres -p csample.lib`" and you should see the function in the library! Strange name, eh?

25 Last step: Now go back to Visual Cafe. Click `Project/Execute`. Click your button and, voila, the number 6 appears in the textfield. You have communication from your C dll to Java!

The End