

JNI로 C와 Java 연동하기 (예제_HelloWorld)

C코드 → 자바로 웹서버에서 동작되게 하기

참고한 블로그

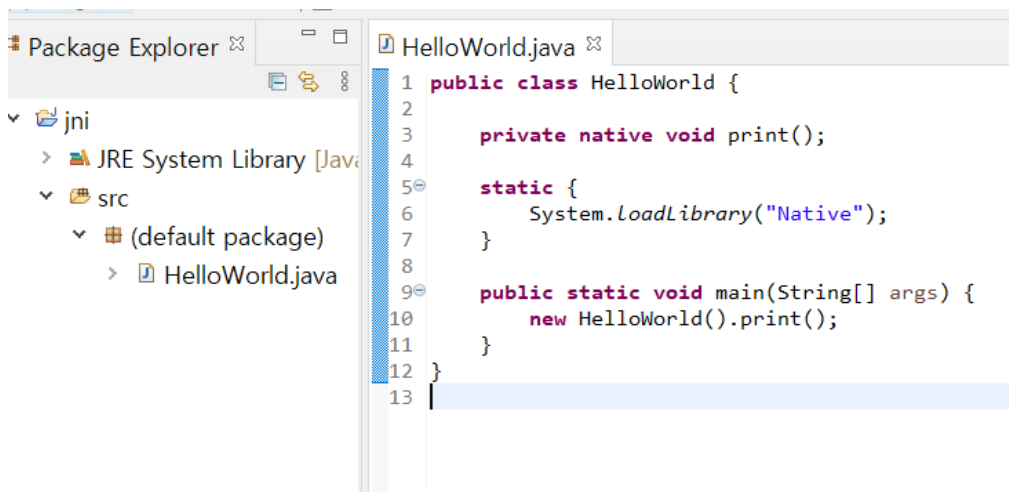
★ <https://huammmm1.tistory.com/444>

<http://lablk.blogspot.com/2018/06/jni-jni-c-java.html>

<https://blog.naver.com/sysganda/30095941540>

1. 자바 프로젝트를 생성한다. (이클립스에서 실행)

2. 자바 파일에 JNI 호출을 위한 코드를 작성 한다.




```
public class HelloWorld {  
  
    private native void print();  
  
    static {  
  
        System.loadLibrary("Native");  
  
    }  
  
    public static void main(String[] args) {  
  
        new HelloWorld().print();  
  
    }  
  
}
```

3. 자바코드 컴파일

이클립스의 경우 원래 저장만 해도 자동으로 컴파일이 되므로, 프로젝트폴더/bin/패키지명/ 에 가보면 class파일이 만들어져 있을 것이다.

« jglor > eclipse-workspace2 > jni > bin

이름

 HelloWorld.class

없으면 javac로 만들어 준다.

```
javac HelloWorld.java
```

4. 헤더파일 작성

c코드 작성에 앞서 헤더파일부터 만들어줘야 한다. javah를 이용해 만든다.



이 때 javah의 대상은 HelloWorld.class파일이므로 javah명령은 bin/패키지명/ 내에서 수행한다.

```
javah -classpath C:\Users\jglor\workspace2\jni\bin HelloWorld
```

```
C:\Users\jglor\workspace2\jni\bin>javah -classpath C:\Users\jglor\workspace2\jni\bin HelloWorld
C:\Users\jglor\workspace2\jni\bin>
```

여기까지 아무 이상 없이 실행되었다면 아래와 같이 헤더 파일이 만들어질 것이다.

로컬 디스크 (C:) > 사용자 > jglor > eclipse-workspace2 > jni > bin

이름		수정한 날짜
	HelloWorld.class	2022-01-
	jni_HelloWorld.h	2022-01-

```

jni_HelloWorld.h
기타 파일 (전역 범위)
1  /* DO NOT EDIT THIS FILE - it is machine generated */
2  #include <jni.h>
3  /* Header for class jni_HelloWorld */
4
5  #ifndef _Included_jni_HelloWorld
6  #define _Included_jni_HelloWorld
7  #ifdef __cplusplus
8  extern "C" {
9  #endif
10 #ifndef __cplusplus
11     /* Class:      jni_HelloWorld
12      * Method:     print
13      * Signature:  ()V
14      */
15     JNIEXPORT void JNICALL Java_jni_HelloWorld_print
16     (JNIEnv *, jobject);
17 #endif
18 #endif
19     /* Class:      jni_HelloWorld
20      * Method:     printHelloWorld
21      * Signature:  ()V
22      */
23     JNIEXPORT void JNICALL Java_jni_HelloWorld_printHelloWorld
24     (JNIEnv *, jobject);
25 #endif
26 #ifdef __cplusplus
27 }
28 #endif
29

```

> 생성된 헤더파일

이 파일은 수정하면 안 되며 c에서 구현해줄 함수의 선언 부를 볼 수가 있다.
이 헤더파일을 include하여 c파일을 만들 것이다.

5. C 함수 몸체 코드 작성

이제 헤더파일에 대한 C 함수 몸체를 작성한다. 대충 메모장으로 다음과 같이 입력해주자.

```

#include<jni.h>

#include "jni_HelloWorld.h"

#include<stdio.h>

JNIEXPORT void JNICALL Java_jni_HelloWorld_print(JNIEnv *env, jobject obj)
{
    printf("Hello world!\n");

    return;
}

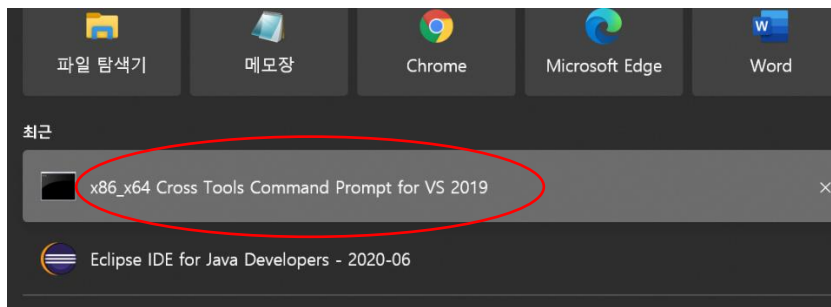
```

* #include "jni_HelloWorld.h" 부분은 자신이 생성한 헤더파일의 이름으로 놓는다
그리고 아무 이름으로 c파일로 저장하고, 헤더파일과 동일한 bin/패키지명/에 놔준다.
(편의상 HelloWorld.c로 저장)

6. dll파일 생성

이제 만들어진 c파일을 토대로 dll파일을 생성할 것이다.

VS용 개발자 명령 프롬프트 - 관리자 권한으로 실행



C:\> x86_x64 Cross Tools Command Prompt for VS 2019

```
*****
** Visual Studio 2019 Developer Command Prompt v16.7.5
** Copyright (c) 2020 Microsoft Corporation
*****
[vcvarsall.bat] Environment initialized for: 'x86_x64'
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community>
```

앞에서 만들어둔 .c파일이 위치한 path로 이동한다. (bin/패키지명)

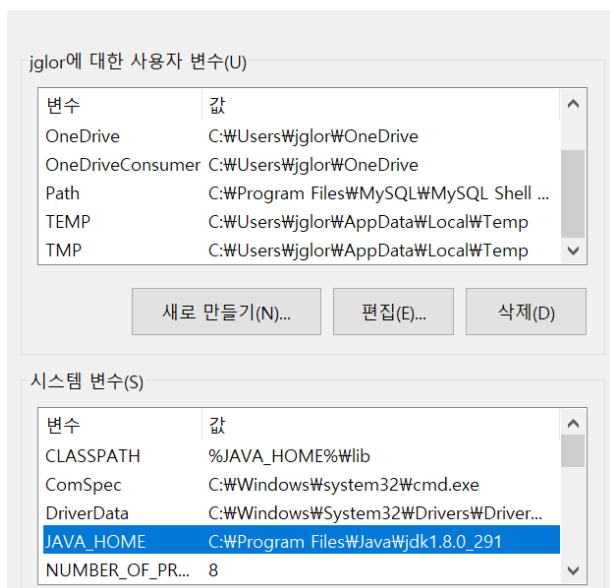
```
cd -I"C:\Program Files\Java\jdk1.8.0_291\include" -I"C:\Program Files\Java\jdk1.8.0_291\include\win32"
-LD HelloWorld.c -FeNative.dll
```

* -I 옵션 뒤의 path는 자신이 설치한 jdk의 path로 설정해야 한다

어디에 설치했는지 기억이 안나면, 고급시스템속성 - 환경변수 에서 JAVA_HOME로 확인 가능

* 첫번째 -I 옵션 뒤에는 include폴더 까지 입력, 두번째 -I 옵션 뒤에는 win32까지 입력

환경 변수



```

C:\Users\jglor\eclipse-workspace2\jni\bin>cl -I"C:\Program Files\Java\jdk1.8.0_291\include" -I"C:\Program Files\Java\jdk1.8.0_291\include\win32" -LD HelloWorld.c -FeNative.dll
Microsoft (R) C/C++ 최적화 컴파일러 버전 19.27.29112(x64)
Copyright (c) Microsoft Corporation. All rights reserved.

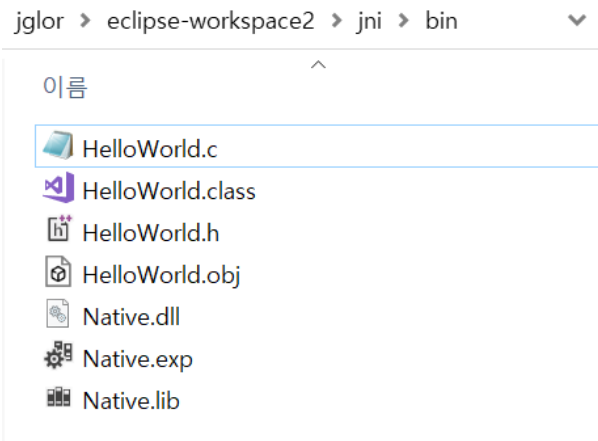
HelloWorld.c
Microsoft (R) Incremental Linker Version 14.27.29112.0
Copyright (C) Microsoft Corporation. All rights reserved.

/dll
/implib:Native.lib
/out:Native.dll
HelloWorld.obj
Native.lib 라이브러리 및 Native.exp 개체를 생성하고 있습니다.

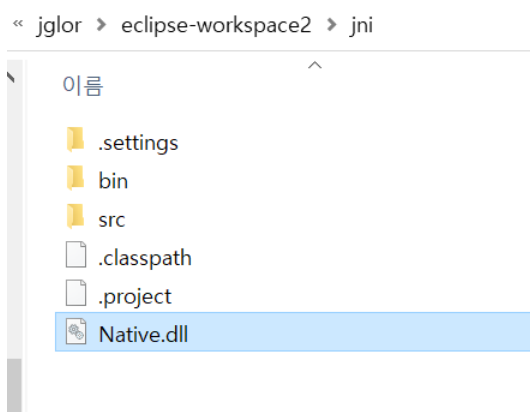
C:\Users\jglor\eclipse-workspace2\jni\bin>

```

이제 bin/jni 폴더로 가보면 Native.dll 파일이 생성되어 있을 것이다.



Native.dll 파일을 복사한 후 프로젝트 폴더의 루트에 복사해준다.



7. 이클립스로 돌아와 실행을 시켜본다.

