通过RegisterNatives注册原生方法

## 1.概述

在上一次的笔记[《JNI学习笔记》](https://link.jianshu.com?t=http:/jellypaul.github.io/java/2016/08/08/JNI%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0.html)中介绍了Native程序与Java程序的互相调用。其中Java调用Nativie方法通常的步骤是：

* 1. 声明native方法： private native void sayHello();
  2. 通过javah 生成native程序的头文件HelloJNI.h
  3. 实现对应的navtive方法

JNIEXPORT void JNICALL Java\_HelloJNI\_sayHello(JNIEnv \*env, jobject thisObj) {

printf("Hello World!\n");

return;

}

这种方法每次增加方法时，native方法名称都需要带上长长的一坨JNIEXPORT void JNICALL Java\_HelloJNI\_，包名很长的话更是不能忍。除此之外对于Native方法的方法名也不能修改。

如果你想摆脱这种冗长的规则，不妨尝试在JNI的JNI\_OnLoad()方法中通过RegisterNatives注册Java需要调用的Native方法。

## 2.JavaVM和JNIEnv

注册navtive方法之前我们需要了解JavaVM, JNIEnv：

JavaVM和JNIEnv是JNI提供的结构体。

JavaVM提供了允许你创建和销毁JavaVM的"invokation interface"。理论上在每个进程中你可以穿件多个JavaVM，但是Android只允许创造一个。

JNIEnv提供了大部分JNI中的方法。在你的Native方法中的第一个参数就是JNIEnv。

JNIEnv用于线程内部存储。因此，不能多个线程共享一个JNIEnv。在一段代码中如果无法获取JNIEnv，你可以通过共享JavaVM并调用GetEnv()方法获取。

## 3.JNI\_OnLoad()方法

当Java层代码中执行：

System.loadLibrary("NativeLib"); //NativeLib 为native模块名称

Native中的JNI\_OnLoad(JavaVM \*vm, void \*reserved)方法会被调用。此时可以注册对应于Java层调用的navtive方法。

## 4.JNINativeMethod结构体

typedef struct {

const char\* name;

const char\* signature;

void\* fnPtr;

} JNINativeMethod;

以上代码为jni.h中的源码，可见JNINativeMethod包含三个元素：方法名，方法签名，native函数指针。

该结构体用于描述需要注册的方法信息。

## 5.RegisterNatives方法

jint RegisterNatives(jclass clazz, const JNINativeMethod\* methods, jint nMethods)

该方法是JNI环境提供的用于注册Native方法的方法。

## 6.完整实现

### 6.1.Java代码实现

Java层代码不需要做任何调整

public class NativeLib{

public static native String getNativeString();

static {

System.loadLibrary("NativeLib");

}

}

### 6.2.C++代码实现

#### 6.2.1实现Native方法

//可以随意定义方法名

jstring getString()

{

JNIEnv \*env = NULL;

g\_jvm->AttachCurrentThread(&env, NULL); //g\_jvm为JavaVM指针

return env->NewStringUTF("This is Natvie String!");

}

#### 6.2.2定义JNINativeMethod数组，声明需要注册的方法

static JNINativeMethod methods[] = {

{"getNativeString", "()Ljava/lang/String;", reinterpret\_cast<void\*>(getString)}

};

其中getNativeString为Java类中定义的Native方法名。

()Ljava/lang/String;为方法的签名，()表示该方法无参数，Ljava/lang/String;表示返回值为Java中的String类型。具体签名规则请参考[《JNI学习笔记》](https://link.jianshu.com?t=http:/jellypaul.github.io/java/2016/08/08/JNI%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0.html)中的内容。

reinterpret\_cast<void\*>(getString)为Native实现的方法名。这里强制转换成了函数指针。

#### 6.2.3JNI\_OnLoad()方法实现

JNIEXPORT jint JNI\_OnLoad(JavaVM \*vm, void \*reserved)

{

JNIEnv\* env;

if (JNI\_OK != vm->GetEnv(reinterpret\_cast<void\*\*> (&env),JNI\_VERSION\_1\_4)) {

LOGW("JNI\_OnLoad could not get JNI env");

return JNI\_ERR;

}

g\_jvm = vm; //用于后面获取JNIEnv

jclass clazz = env->FindClass("com/example/myndkproj/NativeLib"); //获取Java NativeLib类

//注册Native方法

if (env->RegisterNatives(clazz, methods, sizeof(methods)/sizeof((methods)[0])) < 0) {

LOGW("RegisterNatives error");

return JNI\_ERR;

}

return JNI\_VERSION\_1\_4;

}