**[cocos2d-x中有一个JniHelper类详细使用](http://blog.csdn.net/luxiaoyu_sdc/article/details/15874505)**

分类： [cocos2d-x](http://blog.csdn.net/luxiaoyu_sdc/article/category/1395870) [jni](http://blog.csdn.net/luxiaoyu_sdc/article/category/1675069) 2013-11-13 18:58 4169人阅读 [评论](http://blog.csdn.net/luxiaoyu_sdc/article/details/15874505#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/luxiaoyu_sdc/article/details/15874505#report)

目录[(?)](http://blog.csdn.net/luxiaoyu_sdc/article/details/15874505)[[+]](http://blog.csdn.net/luxiaoyu_sdc/article/details/15874505" \o "展开)

**主体思路**

通过JNI获取java虚拟机，再获取当前程序的JNI环境，通过JNI环境获取需要调用的java类信息，再获取需要调用的java类中的函数信息。再通过JNI环境调用，使用类信息、函数信息，调用对应的java函数。  
看起来好像有点复杂，but不用担心，cocos2d-x中有一个JniHelper类(头文件的copyright为：cocos2d-x.org，是Google提供的还是cocos2d-x小组自己封装的我就不清楚了)，它已经把这些工作封装好了。

**JniHelper类的使用**

加入如下头文件：

#include "platform/android/jni/JniHelper.h"

需要使用的接口如下：

static bool getStaticMethodInfo(JniMethodInfo &methodinfo, const char \*className, const char \*methodName, const char \*paramCode); static bool getMethodInfo(JniMethodInfo &methodinfo, const char \*className, const char \*methodName, const char \*paramCode);

实现上我们只需要使用上面这两个接口，就可以获取java类的所有函数信息了。JNI环境的获取、各种错误处理都已经在这两个接口实现中封装好了。  
先上代码，再来依次讲解每个参数的意义和使用方法：

//函数信息结构体 JniMethodInfo minfo; bool isHave = JniHelper::getStaticMethodInfo(minfo,/\*JniMethodInfo的引用\*/ "com/omega/MyApp",/\*类的路径\*/ "getJavaActivity",/\*函数名\*/ "()Ljava/lang/Object;");/\*函数类型简写\*/ jobject activityObj; if (isHave) { //CallStaticObjectMethod调用java函数，并把返回值赋值给activityObj activityObj = minfo.env->CallStaticObjectMethod(minfo.classID, minfo.methodID); }

OK，很简单。上面的代码，就是使用JNI在C++中调用java类静态函数的典型使用方法。只有两步：

* 1. 获取java函数的信息，classid、methodid等等
* 2. 选择JNIEnv中的接口，进行函数调用

**getStaticMethodInfo参数详解**

两个接口的参数一样，意义也相同，详解如下：  
JniMethodInfo &methodinfo JniMethodInfo对象的引用，函数执行中会把jniEvn、classid、methodid写入到引用中。  
const char \*className 类的路径，把类的完整包名写全，用法如以上代码。  
const char \*methodName 函数名，函数名写上就行了。

**const char \*paramCode 函数类型简写**  
这个参数需要单独介绍，它的格式为：(参数)返回类型。  
例如：无参数，void返回类型函数，其简写为 ()V  
java中的类型对应的简写如下：

|  |  |
| --- | --- |
| 参数类型 | 参数简写 |
| boolean | Z |
| byte | B |
| char | C |
| short | S |
| int | I |
| long | J |
| float | F |
| double | D |
| void | V |
| Object | Ljava/lang/String; L用/分割类的完整路径 |
| Array | [Ljava/lang/String; [签名 [I |

**多参数的函数**  
如果函数有多个参数，直接把简写并列即可。注意Object与Array型参数简写结尾的分号，示例：  
IIII //4个int型参数的函数  
ILjava/lang/String;I //整形，string类型，整形组合 (int x, String a, int y)

**通过JNIEnv进行函数调用**

JNIEvn有一系列的CallStatic[返回类型]Method、Call[返回类型]Method接口，需要针对不同的函数返回类型选择调用。  
[返回类型]以函数返回类型的不同，对应不同的函数名。  
例如：  
CallStaticVoidMethod ———void  
CallVoidMethod ———void  
其对应关系如下：

|  |  |
| --- | --- |
| 函数名 | 函数返回值类型 |
| Void | void |
| Object | jobject |
| Boolean | jboolean |
| Byte | jbyte |
| Char | jchar |
| Short | jshort |
| Int | jint |
| Long | jlong |
| Float | jfloat |
| Double | jdouble |

**参数传递**  
调用有参数的java函数时，需要把对应的参数传递进去。需要把参数按顺序加入到classid、methodid后面，并且需要做类型转换。例如：

jint jX = 10; jint jY = 10; minfo.env->CallStaticVoidMethod(minfo.classID, minfo.methodID, jX, jY);

参数类型转换关系如下：

|  |  |
| --- | --- |
| C++类型 | JAVA类型 |
| boolean | jboolean |
| byte | jbyte |
| char | jchar |
| short | jshort |
| int | jint |
| long | jlong |
| float | jfloat |
| double | jdouble |
| Object | jobject |
| Class | jclass |
| String | jstring |
| Object[] | jobjectArray |
| boolean[] | jbooleanArray |
| byte[] | jbyteArray |
| char[] | jcharArray |
| short[] | jshortArray |
| int[] | jintArray |
| long[] | jlongArray |
| float[] | jfloatArray |
| double[] | jdoubleArray |

**string类型的转换**  
实际上我们最常用的参数类型，主要是内建的数据类型、string字符串类型。数据类型可以直接转为j类型，但是string类型需要做如下处理：

jstring jmsg = minfo.env->NewStringUTF("http://www.baidu.com"); minfo.env->CallStaticVoidMethod(minfo.classID, minfo.methodID, jmsg);

**非静态函数的调用**

非静态函数的调用与静态函数的调用类型，但是需要通过一个静态函数获取java类对象。  
示例：

//C++代码 //1. 获取activity静态对象 JniMethodInfo minfo; bool isHave = JniHelper::getStaticMethodInfo(minfo, "com/omega/MyApp", "getJavaActivity", "()Ljava/lang/Object;"); jobject activityObj; if (isHave) { //调用静态函数getJavaActivity，获取java类对象。 activityObj = minfo.env->CallStaticObjectMethod(minfo.classID, minfo.methodID); } //2. 查找displayWebView接口，获取其函数信息，并用jobj调用 isHave = JniHelper::getMethodInfo(minfo,"com/omega/MyApp","displayWebView", "(IIII)V"); if (!isHave) { CCLog("jni:displayWebView 函数不存在"); } else { //调用此函数 jint jX = (int)tlX; jint jY = (int)tlY; jint jWidth = (int)webWidth; jint jHeight = (int)webHeight; //调用displayWebView函数，并传入参数 minfo.env->CallVoidMethod(activityObj, minfo.methodID, jX, jY, jWidth, jHeight); }

**详尽的示例代码**

最后，放一块比较详细的JNI使用代码，基本上覆盖了的全部使用情况。

JniMethodInfo minfo;//JniHelper /\* 测试用方法 \*/ /\*bool isHave = JniHelper::getStaticMethodInfo(minfo,"com/cocoa/HiWorld","loginGree", "()V"); // if (isHave) { //CCLog("有showText "); minfo.env -> CallStaticVoidMethod(minfo.classID,minfo.methodID); }else { //CCLog("没有方法showText"); }\*/ /\* 分享 \*/ /\*//将c++中的string转换成java中的string //char str[] = "test"; bool isHave = JniHelper::getStaticMethodInfo(minfo,"com/cocoa/HiWorld","shareSina", "(Ljava/lang/String;Ljava/lang/String;)V"); // if (isHave) { //CCLog("有share "); jstring jstr = minfo.env->NewStringUTF("test1 share"); jstring jst = minfo.env->NewStringUTF("/data/data/com.cocoa/cy.png"); //jstring jst = minfo.env->NewStringUTF(""); minfo.env -> CallStaticVoidMethod(minfo.classID,minfo.methodID,jstr,jst); }else { //CCLog("没有方法share"); }\*/ /\* 设置高分 \*/ /\*jint ind = 0; jlong lsre = 2202l; bool isHave = JniHelper::getStaticMethodInfo(minfo,"com/cocoa/HiWorld","setHighScore", "(IJ)V"); if (isHave) { minfo.env -> CallStaticVoidMethod(minfo.classID,minfo.methodID,ind,lsre); }\*/ /\* 成就解锁 \*/ /\*jint aind = 0; bool isHave = JniHelper::getStaticMethodInfo(minfo,"com/cocoa/HiWorld","unLock", "(I)V"); if (isHave) { minfo.env -> CallStaticVoidMethod(minfo.classID,minfo.methodID,aind); }\*/ /\* 测试用方法 \*/ bool isHave = JniHelper::getStaticMethodInfo(minfo,"com/cocoa/HiWorld","rtnActivity","()Ljava/lang/Object;"); jobject jobj; if (isHave) { jobj = minfo.env->CallStaticObjectMethod(minfo.classID, minfo.methodID); } //CCLog(" jobj存在"); /\* 测试用方法,非静态无参数无返回值方法 \*/ /\*isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","showText", "()V"); if (isHave) { minfo.env -> CallVoidMethod(jobj,minfo.methodID); }\*/ /\* 测试用方法,非静态有java类型的String参数无返回值方法 \*/ /\*isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","showText", "(Ljava/lang/String;)V"); if (isHave) { jstring jmsg = minfo.env->NewStringUTF("msg okey!"); minfo.env -> CallVoidMethod(jobj,minfo.methodID,jmsg); }\*/ /\* 测试用方法，返回java类型的String，有java类型的String和int参数方法 \*/ /\*isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","showText", "(Ljava/lang/String;I)Ljava/lang/String;"); if (isHave) { jstring jmsg = minfo.env->NewStringUTF("msg okey! return string"); jint index = 0; minfo.env -> CallObjectMethod(jobj,minfo.methodID,jmsg,index); }\*/ /\* 测试用方法，返回java类型的String[]，有java类型的String[]和int参数方法 \*/ /\*isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","showText", "([Ljava/lang/String;I)[Ljava/lang/String;"); if (isHave) { jobjectArray args = 0; jstring str; jsize len = 5; const char\* sa[] = {"Hi,","World!","JNI ","is ","fun"}; int i = 0; args = minfo.env->NewObjectArray(len,minfo.env->FindClass("java/lang/String"),0); for(i=0;iNewStringUTF(sa[i]); minfo.env->SetObjectArrayElement(args,i,str); } //minfo.env->GetStringArrayRegion(args,0,10,buf); //jintArray jmsg = {1,2,3}; //minfo.env->NewStringUTF("msg okey! return string"); jint index = 0; minfo.env -> CallObjectMethod(jobj,minfo.methodID,args,index); }\*/ /\* 测试用方法，无返回类型，有java类型的int[]和int参数方法 \*/ /\*isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","testArr", "([II)V"); if (isHave) { jint buf[]={7,5,8,9,3}; jintArray jintArr; //定义jint数组 jintArr = minfo.env->NewIntArray(5); minfo.env->SetIntArrayRegion(jintArr,0,5,buf); jint index = 0; minfo.env -> CallVoidMethod(jobj,minfo.methodID,jintArr,index); }\*/ /\* 测试用方法，无返回类型，有java类型的byte[]和int参数方法 \*/ isHave = JniHelper::getMethodInfo(minfo,"com/cocoa/HiWorld","testArr", "([BI)V"); if (isHave) { jbyte buf[]={7,5,8,9,3}; jbyteArray jbyteArr; //定义jbyte数组 jbyteArr = minfo.env->NewByteArray(5); minfo.env->SetByteArrayRegion(jbyteArr,0,5,buf); jint index = 0; minfo.env -> CallVoidMethod(jobj,minfo.methodID,jbyteArr,index); }

private static HiWorld hiWorld = null; protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); hiWorld = this; if (detectOpenGLES20()) { // get the packageName,it's used to set the resource path String packageName = getApplication().getPackageName(); super.setPackageName(packageName); // set content setContentView(R.layout.game\_demo); getWindow().setFeatureInt(Window.FEATURE\_CUSTOM\_TITLE, R.layout.window\_title); mGLView = (Cocos2dxGLSurfaceView) findViewById(R.id.game\_gl\_surfaceview); mGLView.setTextField((Cocos2dxEditText) findViewById(R.id.textField)); mGLView.setEGLContextClientVersion(2); mGLView.setCocos2dxRenderer(new Cocos2dxRenderer()); task = new TimerTask() { @Override public void run() { // HiWorld.shoot(hiWorld); Log.e("-------------------", "-------------------"); // 调用c++中的方法 System.out.println("------------------------" + stringZjy1()); } }; timer = new Timer(); timer.schedule(task, 5000); } else { Log.d("activity", "don't support gles2.0"); finish(); } static { System.loadLibrary("game"); } // c++中調用的方法 public static Object rtnActivity() { System.out.println("----------rtnActivity"); return hiWorld; } // c++中調用的方法,传String类型 public void showText(final String msg) { // 添加到主线程 hiWorld.runOnUiThread(new Runnable() { public void run() { System.out.println("----------msg:"+msg); } }); } //c++中調用的方法,传String类型和int类型 public String showText(final String msg,final int index) { // 添加到主线程 hiWorld.runOnUiThread(new Runnable() { public void run() { System.out.println("----------msg:"+msg+"; index="+index); } }); return "okey String showText(final String msg,final int index)"; } //c++中調用的方法,传String[]类型和int类型 public String[] showText(final String[] msg,final int index) { String[] strArr = {"1","2","3","4","5"}; // 添加到主线程 hiWorld.runOnUiThread(new Runnable() { public void run() { for(String \_str:msg){ System.out.println("----------String[] msg:"+\_str+"; index="+index); } } }); return strArr; } //c++中調用的方法,传int[]类型和int类型 public void testArr(final int msg[],final int index) { // 添加到主线程 hiWorld.runOnUiThread(new Runnable() { public void run() { System.out.println("----------int[] msg len："+msg.length); for(int \_bl:msg){ System.out.println("----------int[] msg:"+\_bl+"; index="+index); } } }); } //c++中調用的方法,传int[]类型和int类型 public void testArr(final byte msg[],final int index) { // 添加到主线程 hiWorld.runOnUiThread(new Runnable() { public void run() { System.out.println("----------byte[] msg len："+msg.length); for(int \_bl:msg){ System.out.println("----------byte[] msg:"+\_bl+"; index="+index); } } }); }