NDk动态注册native方法

前边一致：

5.在main目录中新建jni子目录，在其中创建c文件，编写代码

```
//定义函数
JNIEXPOET jstring JNICALL getNativeString(JNIEnv *env,jclass clazz){
        return   (*env)->NewStringUTF(env,"动态注册native 方法");
}
```

7.与java对应绑定起来

```
//设置类名
static  const char* gClassName="com/bluelesson/hellondk2/NdkLib";
//Java和JNI函数的绑定表
static JNINativeMethod method_table[]={
   {"getNativeString","()Ljava/lang/String;",(void*)getNativeString},
};
```

8.做好准备后，在JNI_OnLoad中注册函数

```
//加载so文件时，初始化函数，类似dllmain
JNIEXPORT jint JNI_OnLoad(JavaVM *vm,void *reserved){
     JNIEnv* env=NULL;
     jint  result=-1;
     if((*vm)->GetEnv(vm,(void**)&env,JNI_VERSION_1_4)!=JNI_OK){
       return result;
}
  registerNativeMethods(env,gClassName,method_table,
                  sizeof(method_table/sizeof(method_table[0]))
        //返回jni版本
   return JNI_VERSION_1_4;
}
```

9.注册方法的具体实现

```
//注册native方法到Java中
static int     registerNativeMethod()
{



}
```

```
#include <jni.h>


JNIEXPORT jstring JNICALL
Java_com_bluelesson_hellondk_MainActivity_SetInfo(JNIEnv *env, jobject instance,
                                                  jstring name_) {
    const char *name = env->GetStringUTFChars(name_, 0);

    // TODO

    env->ReleaseStringUTFChars(name_, name);

    return env->NewStringUTF("hello");
}

JNIEXPORT jstring JNICALL SetInfo2(JNIEnv *env, jobject instance,
                                   jstring name_,
                                   jint num) {
    const char *name = env->GetStringUTFChars(name_, 0);

    // TODO
    jstring jstring1 = env->NewStringUTF(name);
```

```
    env->ReleaseStringUTFChars(name_, name);

    return jstring1;
}

extern "C" JNIEXPORT jstring JNICALL
Java_com_bluelesson_hellondk_MainActivity_getNativeString(
            JNIEnv *env, jobject instance)
{
    jstring jstring1;
    jstring1 = env->NewStringUTF("Hello World");
    return jstring1;
}

// 1. 动态注册的Native方法的定义
extern "C" JNIEXPORT jstring JNICALL getNativeString2(
        JNIEnv *env, jobject instance)
{
    jstring jstring1;
    jstring1 = env->NewStringUTF("Hello getNativeString2");
    return jstring1;
}

extern "C" JNIEXPORT jstring JNICALL getNativeString3(
        JNIEnv *env, jobject instance)
{
    jstring jstring1;
    jstring1 = env->NewStringUTF("Hello getNativeString3");
    return jstring1;
}

extern "C" JNIEXPORT jstring JNICALL getNativeString4(
        JNIEnv *env, jobject instance)
{
    jstring jstring1;
    jstring1 = env->NewStringUTF("Hello getNativeString4");
    return jstring1;
}

extern "C" JNIEXPORT jstring JNICALL hello(
        JNIEnv *env, jobject instance)
{
    jstring jstring1;
    jstring1 = env->NewStringUTF("Hello getNativeString5");
    return jstring1;
}


//2. 动态注册的类信息、静态方法数组
static const char* className = "com/bluelesson/hellondk/MainActivity";
// JNINativeMethod 结构体
// 字段1 方法名称，Java层定义的natve方法名
// 字段2 方法签名，方法描述符
// 字段3 本地定义的函数指针
static JNINativeMethod methods[] = {
        {"getNativeString2","()Ljava/lang/String;", (void*)getNativeString2},
        {"getNativeString3","()Ljava/lang/String;", (void*)getNativeString3},
        {"getNativeString4","()Ljava/lang/String;", (void*)getNativeString4},
        {"getNativeString5","()Ljava/lang/String;", (void*)hello},
        {"SetInfo2","(Ljava/lang/String;I)Ljava/lang/String;",  (void*)SetInfo2}
};

#define NULL 0;
JNIEXPORT jint JNICALL JNI_OnLoad(JavaVM* vm, void* reserved){

    // 1. 获取JNI环境以及版本
    JNIEnv *env = NULL;
    JNI_OK;
```

```
    jint nRet = vm->GetEnv((void**)&env,JNI_VERSION_1_6);
    if (nRet != JNI_OK){
        return nRet;
    }
    // 2.注册Native方法
    jclass clazz = env->FindClass(className);// 返回类信息
    env->RegisterNatives(clazz,methods, sizeof(methods)/ sizeof(JNINativeMethod));

    return JNI_VERSION_1_6;
}
```