

# A natív fejlesztés lehetőségei

## *Az NDK bemutatása*

# Android NDK

- NDK: Native Development Kit
- Az NDK lehetővé teszi, hogy natív komponenseket (könyvtárakat) készítsünk és elérjük azokat Android alkalmazásokból
- Megszokott és jól bevált C++ fejlesztés
- Előnyök:
  - > Sebesség növekedés
  - > Kód újrafelhasználás (C++)

# Mit nyújt az NDK?

- Különféle eszközök C és C++ osztálykönyvtárak fordításához
- Natív osztálykönyvtár beágyazási lehetőség standard Android alkalmazásba (apk)
- Natív osztály header-ek és könyvtárak (például 1.5-ös natív Activity támogatás folyamatosan)
- Dokumentáció, példakód és számos tutorial
  - > Vigyázat: sokkal kevesebben használják, ezért jóval kevesebb a segédanyag, és rengeteg elavult!

## Mikor érdemes NDK-ban fejleszteni? 1/2

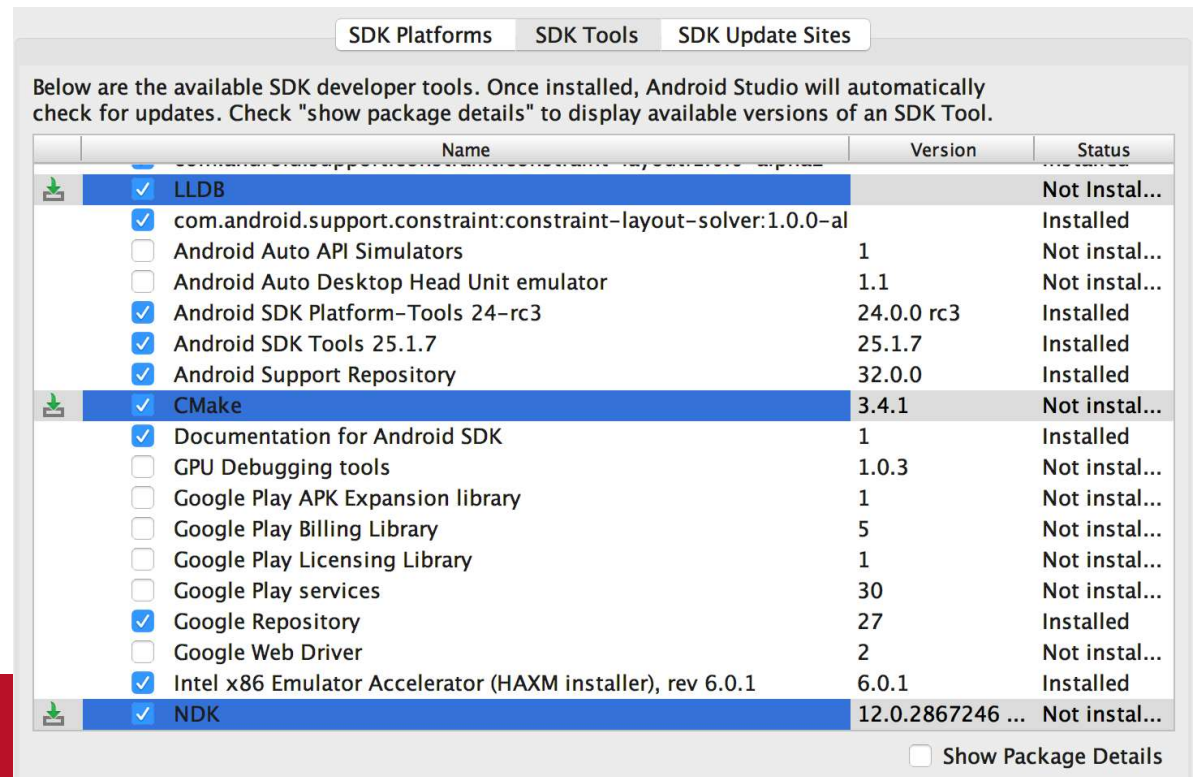
- Gyakran nincs rá különösebb szükség!
- Gondoljuk át az előnyöket és a hátrányokat
- Az NDK használata nem növeli automatikusan a teljesítményt, ugyanakkor a komplexitást mindig
- Csak akkor érdemes használni, ha tényleg indokolható igény van rá, csak azért nem, mert jobban kedveljük a C/C++ fejlesztést

## Mikor érdemes NDK-ban fejleszteni? 2/2

- Érdemes NDK-t használni:
  - > Önálló feladat esetén
  - > CPU intenzív műveleteknél, ami kevés memóriát fogyaszt
  - > Jelfeldolgozás
  - > Fizikai szimuláció
- Érdemes megnézni, hogy nincs-e az adott funkcióra beépített Android API
- Nagyobb C/C++ kódok átemelése esetén hasznos!

# Előkészületek

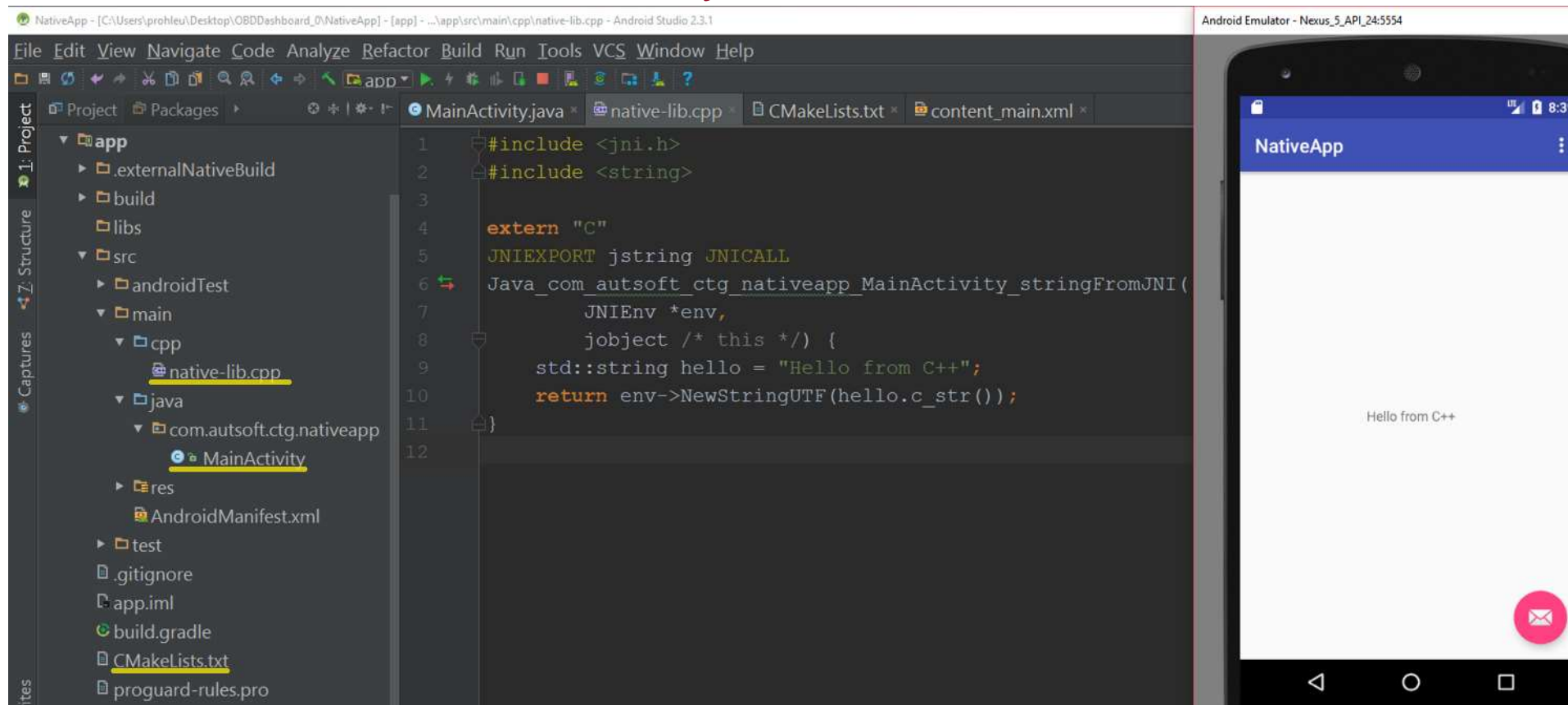
- SDK Manager segítségével letöltjük:
  - > NDK: C/C++ kód használata Androiddal
  - > CMake: külső buildtool, ami fordítja a natív libünket
  - > LLDB: natív kód debugolásához



# Mivel fordítsuk?

- ndk-build: Android.mk + Application.mk - múlt
  - > Elavult, de sajnos legtöbb helyen még ez szerepel
  - > <https://developer.android.com/ndk/guides/build.html>
- CMake: CMakeLists.txt – jelen
  - > „Android Studio intends CMake to be a permanently supported solution.”
  - > Példaprogramok:  
<https://github.com/googlesamples/android-ndk>
- Kísérleti gradle plugin: jövő?
  - > <https://codelabs.developers.google.com/codelabs/android-studio-jni/>

# Natív kód használati módja – Android Studio NDK/JNI



- Részletes útmutató (CMake):
  - > <https://developer.android.com/studio/projects/add-native-code.html>



# NDK projekt elemei 1

- Új Android projekt szóköz nélküli könyvtárba!
- Új könyvtár létrehozása: *cpp*
- Minden natív kód a *cpp* könyvtárba kerül:
  - > *native-lib.cpp*
- Modul gyöker könyvtárában:
  - > *CmakeLists.txt*: CMake build szkriptje

# NDK projekt elemei 2

- *CmakeLists.txt*:

```
cmake_minimum_required(VERSION 3.4.1)
```

```
add_library( # Sets the name of the library.  
            native-lib
```

```
            # Sets the library as a shared library.  
            SHARED
```

```
            # Provides a relative path to your  
            source file(s).  
            src/main/cpp/native-lib.cpp )
```

# NDK projekt elemei 3

- *CmakeLists.txt* – másik lib hozzáadása a sajátunkhoz :

```
find_library( # Sets the name of the path variable.
              log-lib

              # Specifies the name of the NDK library that
              # you want CMake to locate.

              log )

target_link_libraries( # Specifies the target library.
                       native-lib

                       # Links the target library to the log library
                       # included in the NDK.
                       ${log-lib} )
```

# NDK projekt elemei 4

- *native-lib.cpp*:

```
#include <jni.h>
#include <string>

extern "C"

JNIEXPORT jstring JNICALL
Java_com_autsoft_ctg_nativeapp_MainActivity_stringFromJNI(
    JNIEnv *env,
    jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

# A fordítás eredménye

- *APK Analyzer: Build > Analyze APK*

Raw File Size: 2,2 MB, Download Size: 2 MB Compare with...

| File             | Raw File Size | Download Size | % of Total Download s. |
|------------------|---------------|---------------|------------------------|
| classes.dex      | 3,7 MB        | 1,2 MB        | 62,4%                  |
| lib              | 1,2 MB        | 459 KB        | 24%                    |
| mips64           | 236,1 KB      | 72,8 KB       | 3,8%                   |
| libnative-lib.so | 236,1 KB      | 72,8 KB       | 3,8%                   |
| mips             | 230,3 KB      | 71,3 KB       | 3,7%                   |
| libnative-lib.so | 230,3 KB      | 71,3 KB       | 3,7%                   |
| arm64-v8a        | 178,1 KB      | 67,4 KB       | 3,5%                   |
| libnative-lib.so | 178,1 KB      | 67,4 KB       | 3,5%                   |
| x86_64           | 174,5 KB      | 69,4 KB       | 3,6%                   |
| libnative-lib.so | 174,5 KB      | 69,4 KB       | 3,6%                   |
| x86              | 173,7 KB      | 72 KB         | 3,8%                   |
| libnative-lib.so | 173,7 KB      | 72 KB         | 3,8%                   |
| armeabi          | 109,7 KB      | 53,9 KB       | 2,8%                   |

# Első NDK alkalmazás 1

- *NativeLib.java saját állomány:*

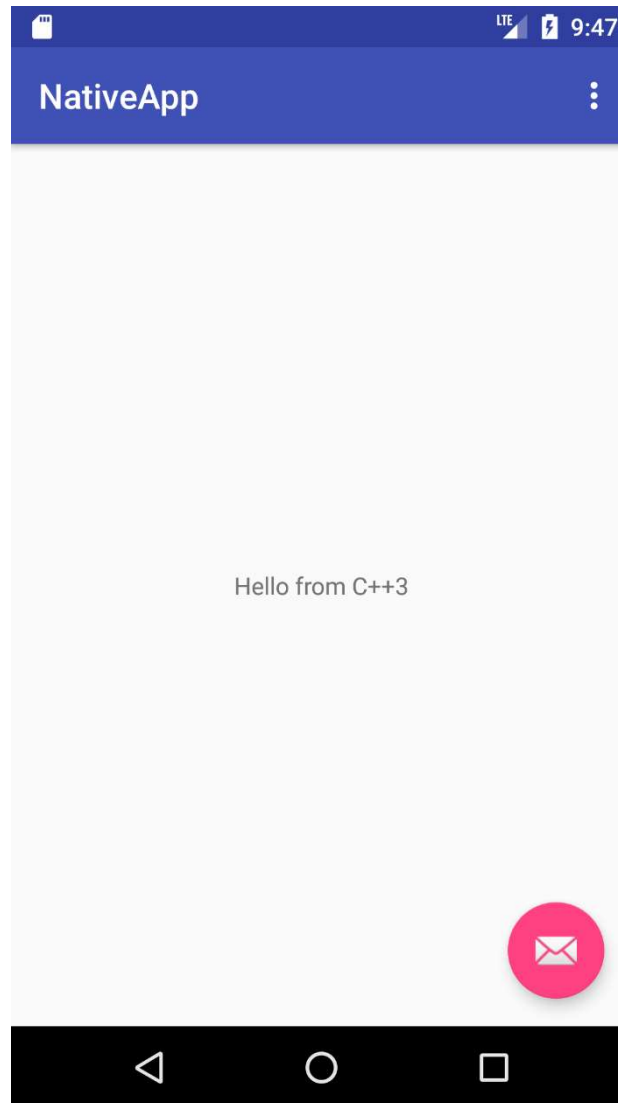
```
public class NativeLib {  
  
    static {  
        System.loadLibrary("native-lib");  
    }  
  
    public native String stringFromJNI();  
  
    public native int add(int a, int b);  
}
```

# Első NDK alkalmazás 2

- *Használat Activity-ből:*

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ...  
        NativeLib nativeLib = new NativeLib();  
  
        TextView tv = (TextView) findViewById(R.id.sample_text);  
        tv.setText(nativeLib.stringFromJNI() +  
                  nativeLib.add(1,2));  
    }  
    ...  
}
```

# Első NDK alkalmazás 3





# Gyakoroljunk!

- Készítsünk egy Android alkalmazást, amely natív oldali függvényeket hív:
  - > sayHello()
  - > add(int a, int b)
  - > pi(int iteration)
    - PI kiszámítása adott iteráción keresztül

