Android NDK で Rust を使う

Stack	Programming
Notebook	Rust
Tag	NDK
Created	@September 29, 2022 2:56 PM
Edited	@September 30, 2022 1:02 PM
Previous	
Next	
Share to Web	☐

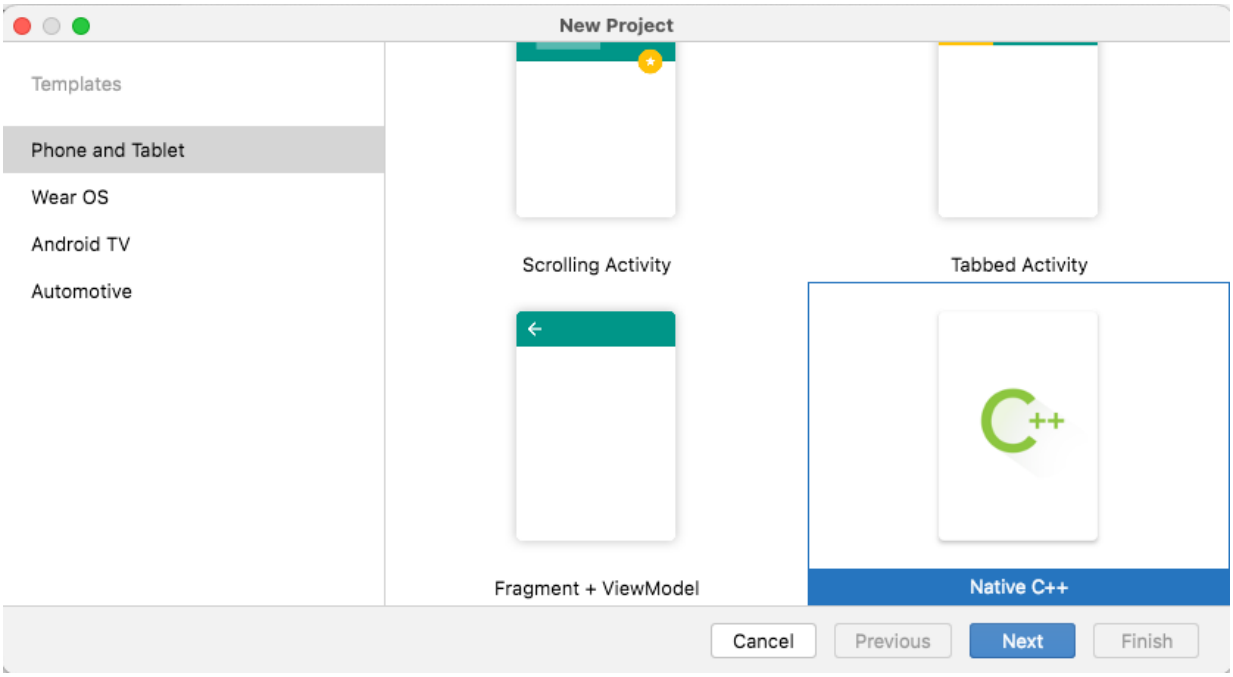
Android NDK に直接 Rust コードを組み込み、JNI から C++ コードではなく Rust コードがコールされるようにする。

参考サイト

https://note.com/navitime_tech/n/n7c758204b362

プロジェクトを作成する

Android Studio を起動し、`Native C++` の雛形で新プロジェクトを作成する。



プロジェクトの設定は以下。

- Application name: Hello Android Rust

- Package name: dev.seabat.android.helloandroidrust

Rust コードを配置する

デフォルトの C++ のネイティブコードは app/src/main/cpp/native-lib.cpp にあるが、Rust コードは app/src/main/rust/ に置くことにする。

```
$ cd hello_android_rust/app/src/main
$ mkdir rust
```

rust/ にライブラリ用の Rust プロジェクトを作る。

```
$ cd rust
$ cargo new --lib hello
```

C++ コードを削除

`Native C++` の雛形で C++ コードを削除する。

app/build.gradleの以下を削除する。

```
android {
  defaultConfig {
    externalNativeBuild {
      cmake {
        cppFlags ''
      }
    }
  }
  externalNativeBuild {
        cmake {
            path file('src/main/cpp/CMakeLists.txt')
            version '3.18.1'
        }
    }
}
```

次に app/src/main/cpp ディレクトリを削除する。

Gradle プラグインをセットアップ

Gradle プラグイン `org.mozilla.rust-android-gradle.rust-android` をインストールする。

以下を プロジェクト直下の build.gradle に追加。

```
// build.gradle
plugins {
  id "org.mozilla.rust-android-gradle.rust-android" version "0.9.3"
}

// app/build.gradle
plugins {
    id 'org.mozilla.rust-android-gradle.rust-android'
}
```

Gradle プラグインのセットアップ方法は **plugins DSL** と **legacy plugin application** があるが、参考サイト通りの **legacy plugin application** で以下のようにだとエラーが発生する。

```
// build.gradle
+buildscript {
+    repositories {
+        maven {
+            url "https://plugins.gradle.org/m2/"
+        }
+    }
+    dependencies {
+        classpath 'org.mozilla.rust-android-gradle:plugin:0.9.3'
+    }
+}

// app/build.gradle
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
+   id 'org.mozilla.rust-android-gradle.rust-android'
}
```

app/build.gradle に以下を追加する。

```
cargo {
    // Rust モジュールのパス
    module  = "src/main/rust/hello"
    // Rust モジュールの名称
    libname = "hello"
    // ビルド対象のアーキテクチャ
    targets = ["arm", "arm64", "x86", "x86_64"]
}

// ビルド時に Rust コードもビルドする
tasks.whenTaskAdded { task ->
    if (task.name == "javaPreCompileDebug" || task.name == "javaPreCompileRelease") {
        task.dependsOn "cargoBuild"
    }
}
```

Android 向けの Rust ツールチェインをインストール。

```
$ rustup target add aarch64-linux-android     # arm64-v8a
$ rustup target add armv7-linux-androideabi   # armeabi-v7a
$ rustup target add i686-linux-android        # x86
$ rustup target add x86_64-linux-android      # x86_64
```

おそらく　Rust のインストール場所 (例: /Users/XXXXXX/.cargo) に Rust ツールチェイン がインストールされるので上記コマンドはどこで実行してもよい。

app/src/main/rust/hello/Cargo.toml に下記の設定を追加する。

```
[lib]
crate-type = ["cdylib"]

[dependencies]
jni = { version = "0.19.0", default-features = false }
```

Android NDK で Rust は使えるのか？調査してみた の通り以下だとエラーになるので注意。

```
[dependencies]
jni = "0.19.0"
```

`crate-type` に指定する `cdylib` については Rust のライブラリ crate_type をまとめてみた を参照。

JNI 経由でコールされる Rust コードを作る

app/src/main/rust/hello/src/lib.rs に以下を追加。

```
use jni::objects::JObject;
use jni::sys::jstring;
use jni::JNIEnv;

#[no_mangle]
pub unsafe extern "C" fn Java_dev_seabat_android_helloandroidrust_MainActivity_stringFromJNI(
   env: JNIEnv,
   _this: JObject,
) -> jstring {
   let hello = "Hello from Rust";

   env.new_string(hello)
       .expect("Couldn't create Java string!")
       .into_inner()
}
```

JNI 経由で関数を呼ぶためには C 言語形式の関数を作る必要がある。下記のキーワードを付けることで関数を C 言語と互換性のある形式でコンパイルする。

```
#[no_mangle]
extern "C" fn
```

C 言語から Rust を呼ぶ方法は以下を参照。

> The Rustonomicon
>
> This guide will use the snappy compression/decompression library as an introduction to writing bindings for foreign code. Rust is currently unable to call directly into a C++ library, but snappy includes a C interface (documented in). Many of these examples use the libc crate, which provides various type definitions for C types, among other things.
>
> https://doc.rust-lang.org/nomicon/ffi.html#calling-rust-code-from-c

2022/09/30 現在 IntelliJ IDEA(Android Studio) には Rust プラグインはない。

Java 層の準備

`Native C++` の雛形として存在する `MainActivity` を利用する。JNI 関数の `stringFromJNI` はすでにコーディングされているのであとは `System.loadLibrary("hello")` となっていることを確認する。

```
...
    external fun stringFromJNI(): String

    companion object {
        // Used to load the 'helloandroidrust' library on application startup.
        init {
            System.loadLibrary("hello")
        }
    }
```

ビルドしてアプリが起動すれば成功。

ビルドエラー対処

ビルドすると以下のエラーが発生する

```
error: linking with `cc` failed: exit code: 1
  |
linking with `cc` failed: exit code: 1

  = note: "cc" "-m64" "-arch" "x86_64" "-L" "/Users/XXXXXX/
```

macOSをMontereyに上げたらcargo buildで「linking with `cc` failed」となった を参考に以下を実行。

```
Xcode Command Line Tools
```

しかし、エラーは解消されず。うーん、どうも Hello World プログラムも同じエラーが出る。ひょっとすると Rust 環境がまずいのでは？ Homebrew による rust セットアップをやめ、公式の方法でインストールしたら上記エラー解消。

```
$ brew uninstall rustup
$ curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

次は以下のエラーが発生。

```
linking with `/Users/nakanishi/Doc/Rust/hello_android_rust/build/linker-wrapper/linker-wrapper.sh` failed: exit status: 127
```

なぜエラーがになっているのかわからず。`sh gradlew debugAssemble` コマンドでビルドしてみると以下のエラーを取得できた。

```
= note: /Users/nakanishi/Doc/Rust/hello_android_rust/build/linker-wrapper/linker-wrapper.sh: line 4: python: command not found
```

`python` というコマンドが見つからないらしい。確かに `which python` でpython が見つからない。おそらくビルドに使用している Mac PC は python2 は `python` コマンド、 python3 は `python3` コマンドで動作するようになっていて、python2 はもう使わないから python2 をアンインストールしたのだろう。

とすると python3 を `python` コマンド実行する必要がありそうだ。

`linker-wrapper.sh` fails on macOS 12.3 since bare `python` is not found #88 で

```
I added
rust.pythonCommand=/opt/homebrew/bin/python3
to samples/app/local.properties
```

という記述を発見。python3 のパスを取得し、

```
$ which python3
/usr/local/bin/python3
```

これを local.properties に以下のように記載してみた。

```
// local.properties
rust.pythonCommand = /usr/local/bin/python3
```

ビルドが解消された。