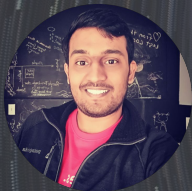


Kotlin MP: Into the Multi-Verse



Nishant Srivastava
@nisrulz

Kotlin MP?



Kotlin MP: Kotlin MultiPlatform

“Utilizing Kotlin language to build for multiple target platforms, enabling code sharing across them while being as flexible as can be”

Kotlin MP: Kotlin MultiPlatform

- ❑ Open Source
- ❑ Uses Kotlin language for common code
- ❑ Multiple target platforms
- ❑ Code sharing of data/domain/presentation layer

Kotlin MP: Kotlin MultiPlatform

- ❑ Integrates with existing native platform
- ❑ Leverage native platform capabilities when needed
- ❑ Optional i.e not opinionated

But why?



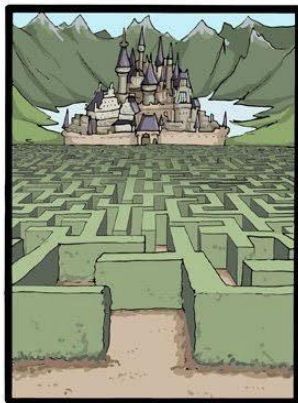
But why?

- ❑ Introduction of Kotlin/Native in the toolchain enabled targeting more platforms than just JS and JVM
- ❑ Existing solutions are very opinionated

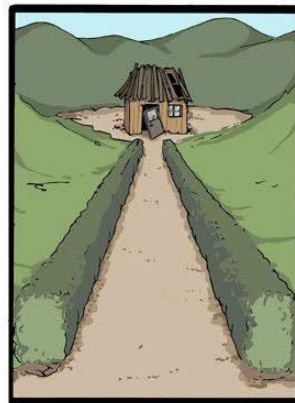
But why?

The dilemma of mobile apps development

Develop a native app for each device and maintain several projects



Use a unique framework (Phonegap, Adobe Air, Appcelerator) and maintain only one project



CommitStrip.com

Into the Multi-Verse of Platforms



...or Target Platforms



Target Platforms

Compiled artifacts to be consumed by specific platform.

`Kotlin/JVM` → JAR/AAR → Java, Android, Spring Boot

`Kotlin/JS` → JS → Javascript, React, Node

Target Platforms

Compiled artifacts to be consumed by specific platform.

Kotlin/Native →

- `androidNativeArm32` and `androidNativeArm64` for Android NDK
- `iosArm32`, `iosArm64`, `iosX64` for iOS
- `watchosArm32`, `watchosArm64`, `watchosX86` for watchOS
- `tvosArm64`, `tvosX64` for tvOS

Target Platforms

Compiled artifacts to be consumed by specific platform.

Kotlin/Native →

- `linuxArm32Hfp`, `linuxMips32`, `linuxMipsel32`, `linuxX64` for Linux
- `macosX64` for MacOS
- `mingwX64` and `mingwX86` for Windows
- `wasm32` for WebAssembly

Code Sharing

Code Sharing

- ❑ Only share common code
 - ❑ Data layer → Networking, Caching, Repositories
 - ❑ Domain layer → Entities, Interactors, Use Cases
 - ❑ Presentation layer → ViewModel, Presenter, Controller
- ❑ Keep the UI separate and native to its respective platform.

...but not Cross Platform

Cross Platform Solutions:

- ❑ Makes you write code in opinionated way i.e Flutter, dart
- ❑ Eventually map all the magic back to native (with or without a bridge) i.e React Native, Xamarin, NativeScript

What's the secret
sauce?

Expect / Actual



Interfaces with SuperPowers 
...and more

Interface

```
interface MyInterface{
    fun platformName(): String
}

class MainActivity : MyInterface {
    override fun platformName(): String = "Android"

    fun createApplicationScreenMessage() : String {
        return "Hello from ${platformName()}"
    }
}
```

Interface

```
interface MyInterface{
    fun platformName(): String // ← Expecting this to be implemented
}

class MainActivity : MyInterface {
    override fun platformName(): String = "Android" // ← Actual implementation

    fun createApplicationScreenMessage() : String {
        return "Hello from ${platformName()}"
    }
}
```

Expect/Actual

```
// Common module
expect fun platformName(): String // ← Expecting this to be implemented

fun createApplicationScreenMessage() : String {
    return "Hello from ${platformName()}"
}

// Platform specific (Android) module
actual fun platformName(): String = "Android" // ← Actual implementation

// Platform specific (iOS) module
actual fun platformName(): String = "iOS" // ← Actual implementation
```

Expect/Actual

```
// Common module
expect class Greeting(name: String) {
    fun greet()
}

// Platform specific (Android) module
actual class Greeting actual constructor(val name: String) {
    actual fun greet() {
        println("Hello $name")
    }
}

// Usage
Greet("FOSDEM").greet() // Hello FOSDEM
```

Kickstart KMP Development

Sharing is Caring

Many ways of sharing code

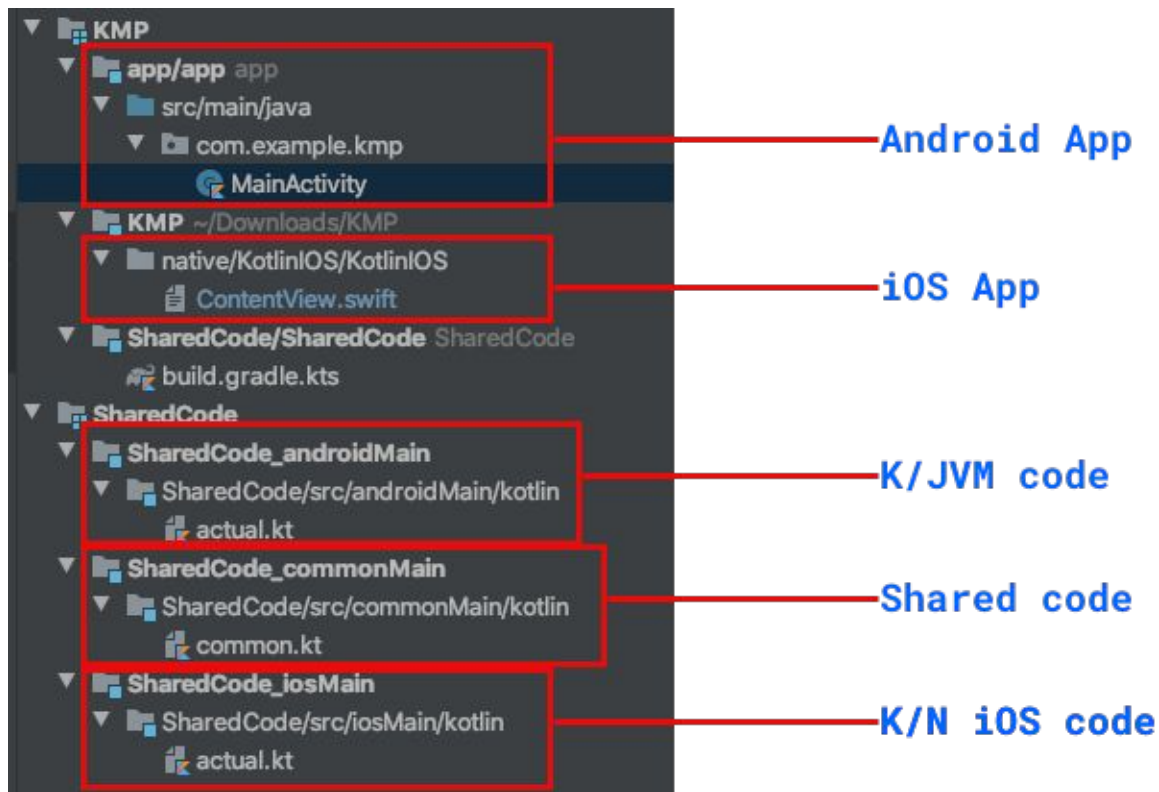
- ❑ Every target platform's code lives in the same repo/project along with the shared code.
 - ❑ Harder to work in a large team.

Sharing is Caring

Many ways of sharing code

- ❑ Everything is in its own repo/project, while Shared code is itself a different project.
 - ❑ Harder to maintain.

Sharing is Caring

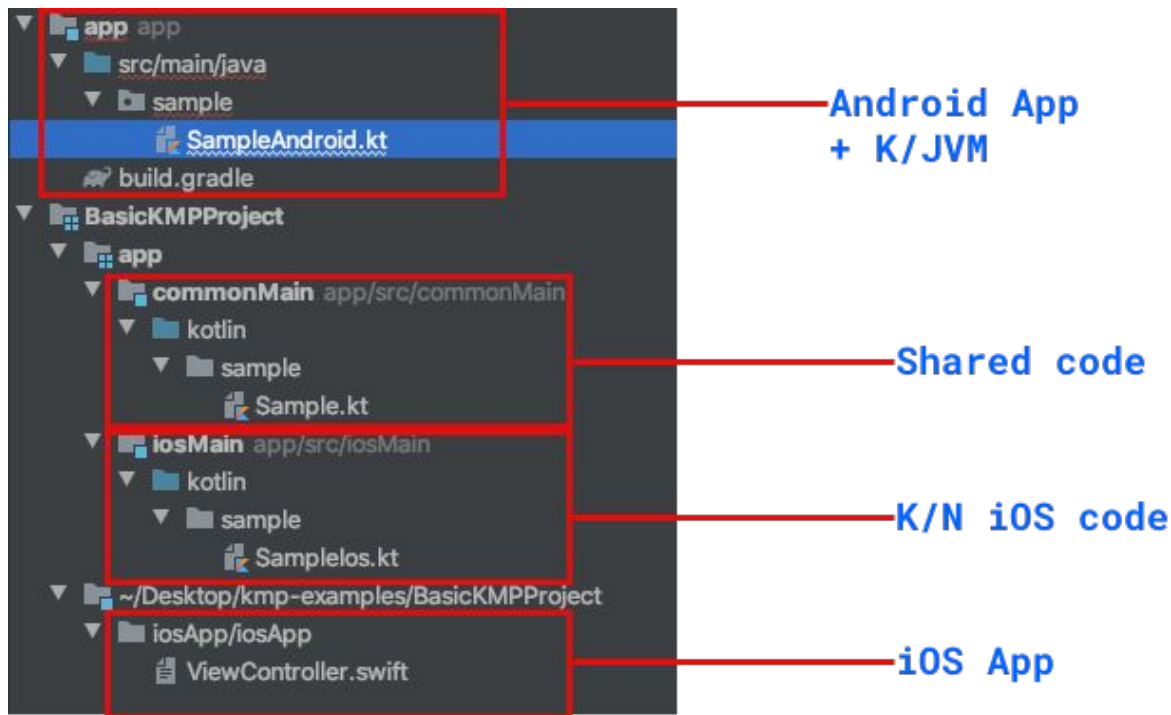


Sharing is Caring

Many ways of sharing code

- ❑ One of the platform includes the shared code, while others refer to it from outside.
 - ❑ Better change/test cycle

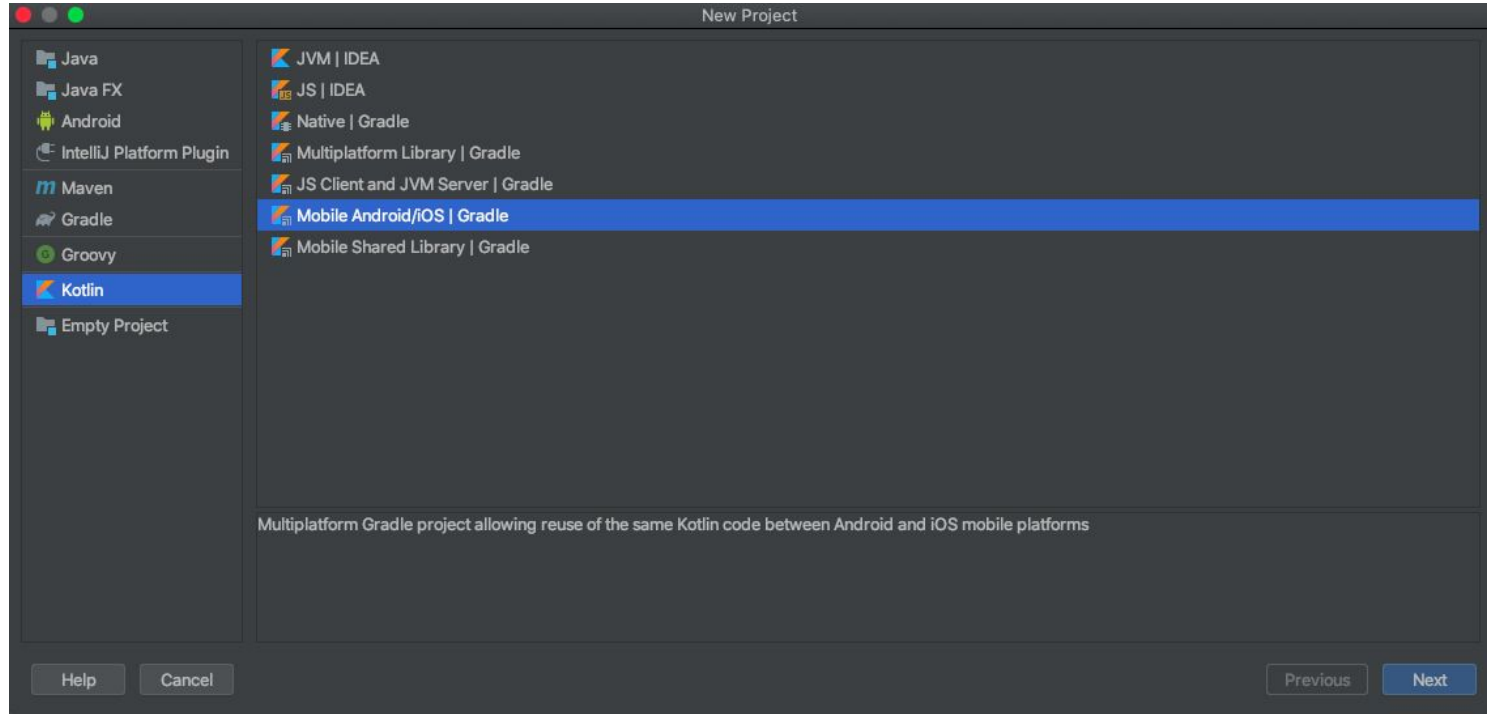
Sharing is Caring



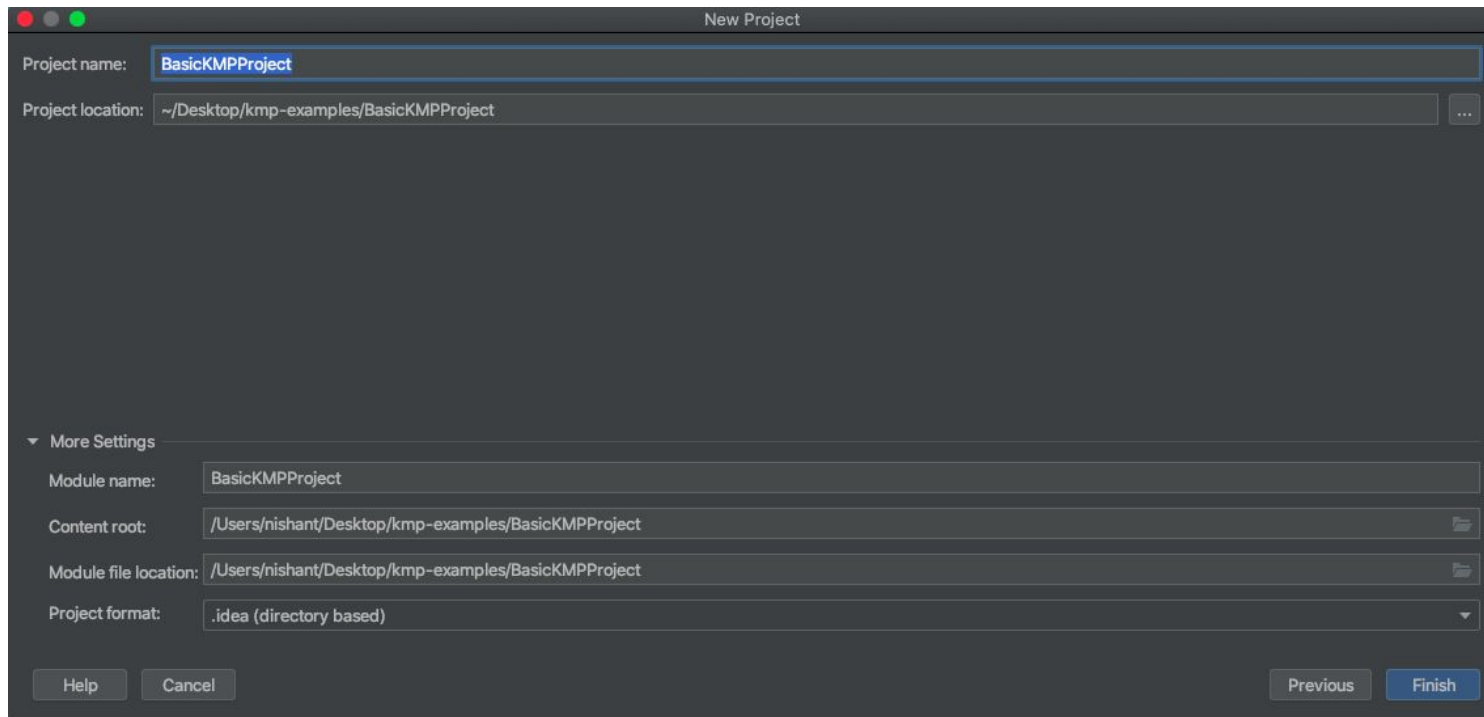
Kickstart KMP Development



Kickstart KMP Development



Kickstart KMP Development



Kickstart KMP Development

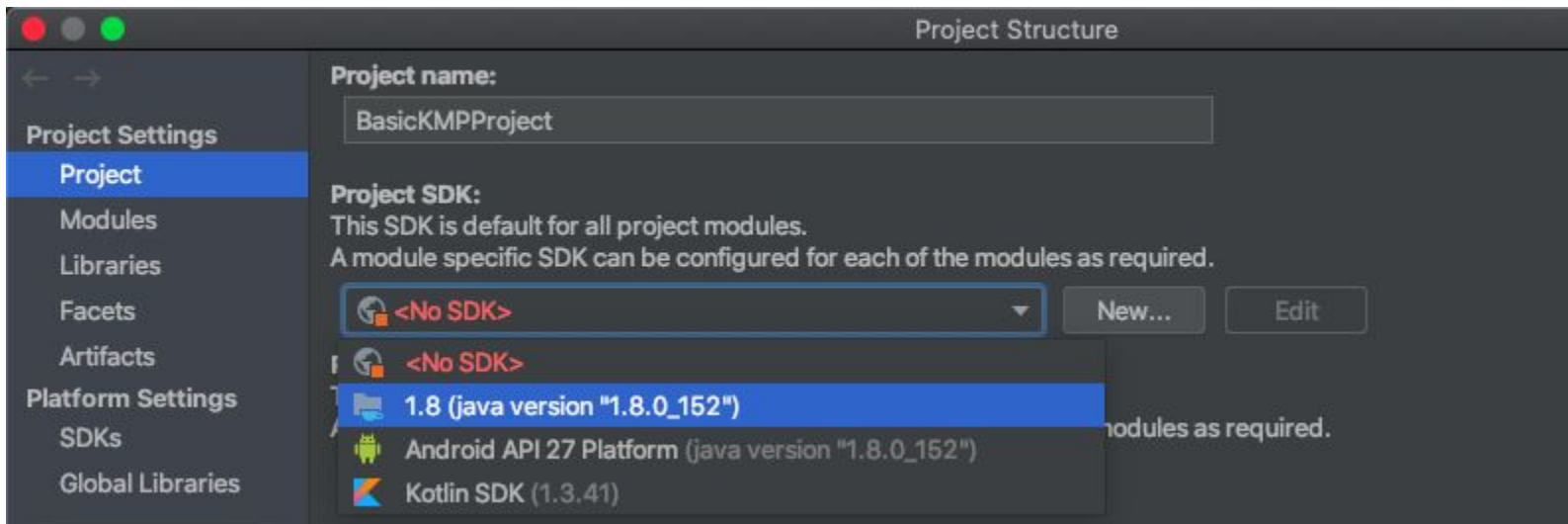
```
local.properties x
1  |## This file must *NOT* be checked into Version Control Systems,
2  |# as it contains information specific to your local configuration.
3  |#
4  |# Location of the SDK. This is only used by Gradle.
5  |# For customization when using a Version Control System, please read the
6  |# header note.
7  |sdk.dir=PleaseSpecifyAndroidSdkPathHere

A problem occurred configuring project ':app'.
> The SDK directory '/Users/nishant/Desktop/kmp-examples/BasicKMPProject/PleaseSpecifyAndroidSdkPathHere' does not exist.
```

Kickstart KMP Development

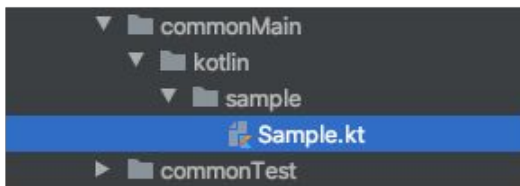
```
local.properties x
1  ## This file must *NOT* be checked into Version Control Systems,
2  # as it contains information specific to your local configuration.
3  #
4  # Location of the SDK. This is only used by Gradle.
5  # For customization when using a Version Control System, please read the
6  # header note.
7  #Sat Feb 01 23:19:28 CET 2020
8  sdk.dir=/Users/<username>/Library/Android/sdk
9
```


Kickstart KMP Development

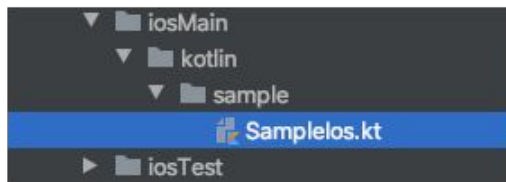


Kickstart KMP Development

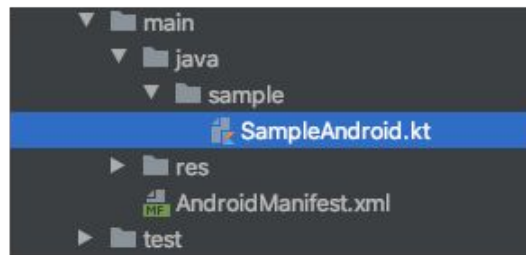
Common code



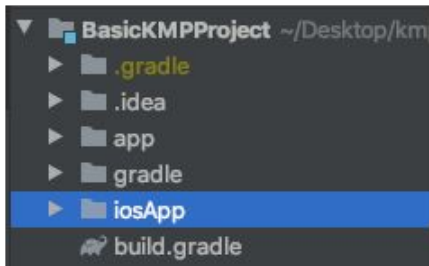
K/N iOS code



Android and K/JVM code

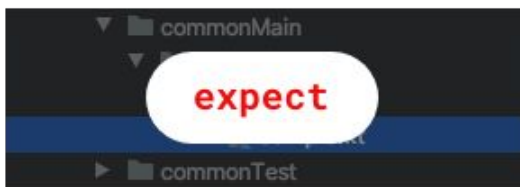


iOS specific code

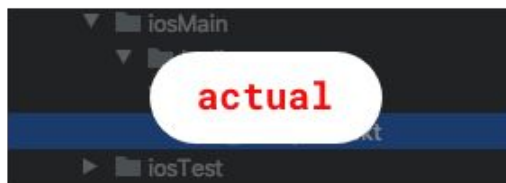


Kickstart KMP Development

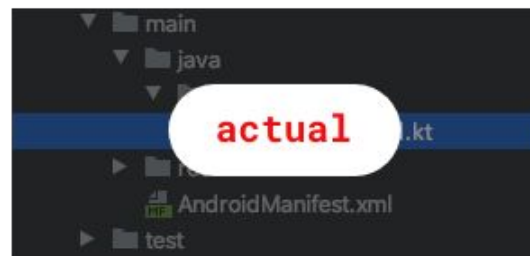
Common code



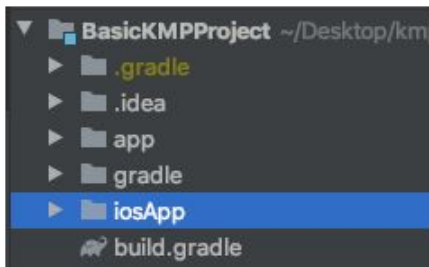
K/N iOS code



Android and K/JVM code



iOS specific code



Kickstart KMP Development

```
// app/build.gradle

plugins {
    id 'org.jetbrains.kotlin.multiplatform' version '1.3.61'
}

...
```

Kickstart KMP Development

```
// app/build.gradle

kotlin {
    android("android")
    // This is for iPhone emulator
    // Switch here to iosArm64 (or iosArm32) to build library for iPhone device
    iosX64("ios"){
        binaries {
            framework()
        }
    }

    sourceSets{...}
}
```

Kickstart KMP Development

```
// app/build.gradle

kotlin {
    android("android")
    // This is for iPhone emulator
    // Switch here to iosArm64 (or iosArm32) to build library for iPhone device
    tvosX64("tvos"){
        binaries {
            framework()
        }
    }

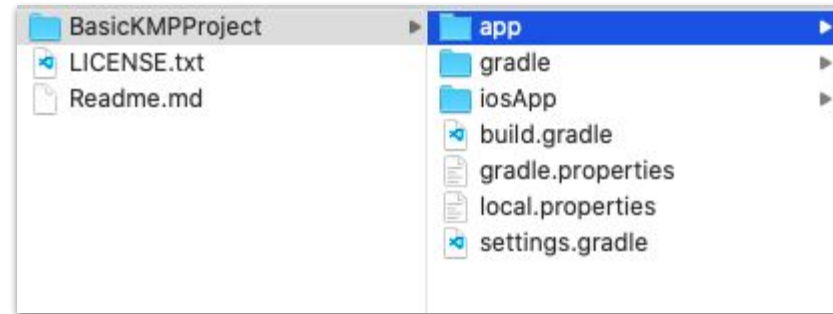
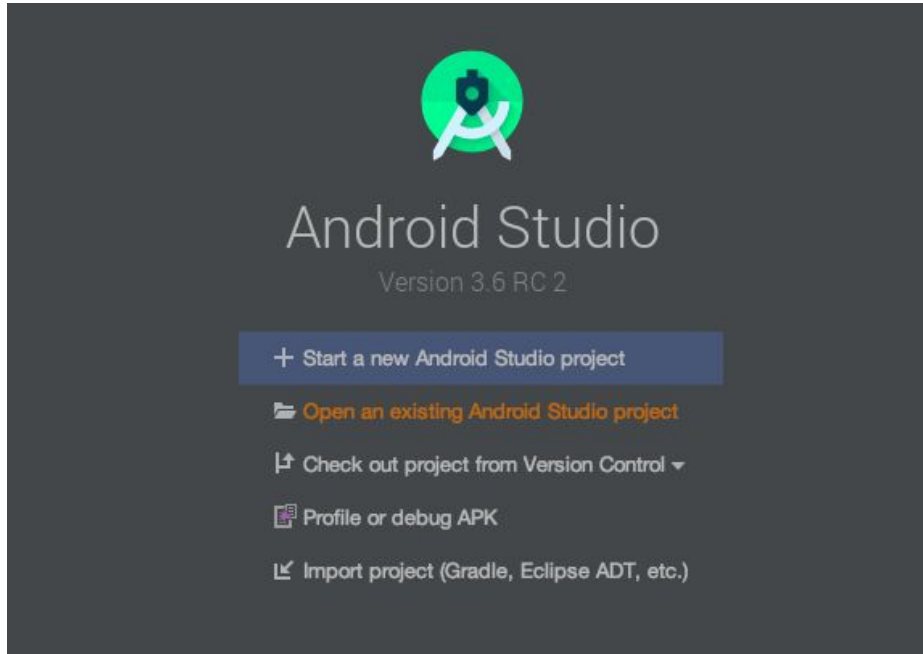
    sourceSets{...}
}
```

Kickstart KMP Development

```
// app/build.gradle

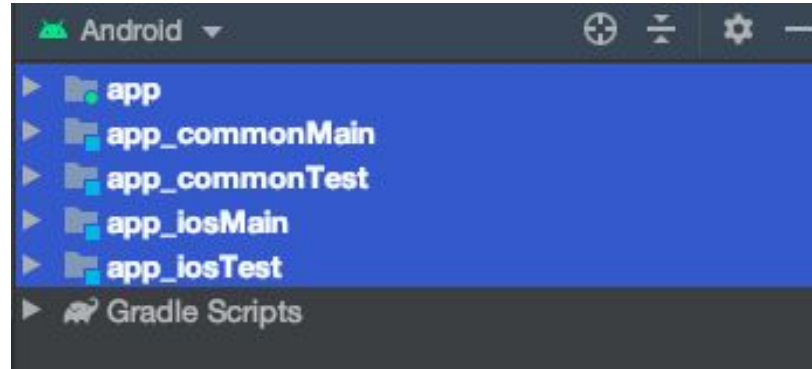
kotlin {
    ..
    sourceSets {
        commonMain.dependencies {
            implementation kotlin('stdlib-common')
        }
        commonTest {...}
        androidMain.dependencies {
            implementation kotlin('stdlib')
        }
        androidTest {...}
        iosMain {...}
        iosTest {...}
    }
}
```

Kickstart KMP Development

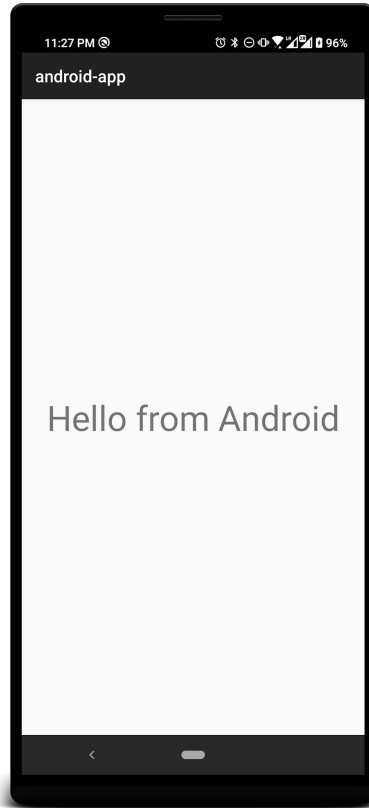


android

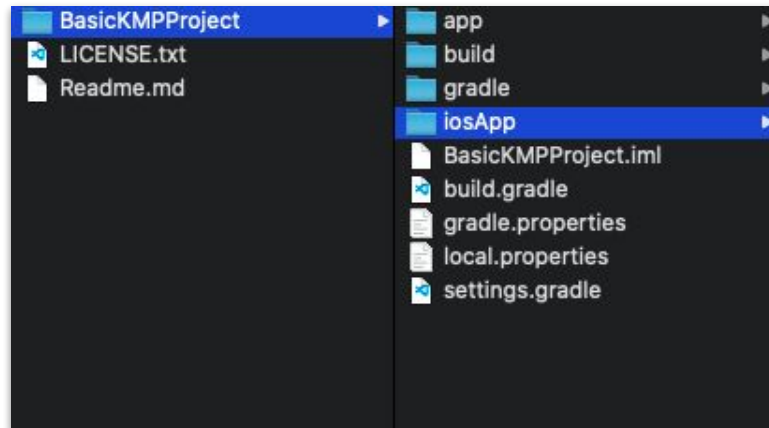
Kickstart KMP Development



Kickstart KMP Development



Kickstart KMP Development



Kickstart KMP Development

```
app 1 issue
  ! Uncategorized
    ! Command PhaseScriptExecution failed with a nonzero exit code
  iosApp 3 issues
    ! Warning
      ! Mapping architecture arm64 to x86_64. Ensure that this target's Architectures and Valid Architectur...
      ! Mapping architecture armv7 to i386. Ensure that this target's Architectures and Valid Architectur...
      ! Capabilities for Signing & Capabilities may not function correctly because its entitlements...

export TOOLCHAIN_DIR=/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain
export TREAT_MISSING_BASELINES_AS_TEST_FAILURES=NO
export TeamIdentifierPrefix=FAKETEAMID.
export UID=501
export UNLOCALIZED_RESOURCES_FOLDER_PATH=app.framework
export UNSTRIPPED_PRODUCT=NO
export USER=nishant
export USER_APPS_DIR=/Users/nishant/Applications
export USER_LIBRARY_DIR=/Users/nishant/Library
export USE_DYNAMIC_NO_PIC=YES
export USE_HEADERMAP=YES
export USE_HEADER_SYMLINKS=NO
export USE_LLVM_TARGET_TRIPLES=YES
export USE_LLVM_TARGET_TRIPLES_FOR_CLANG=YES
export USE_LLVM_TARGET_TRIPLES_FOR_LD=YES
export USE_LLVM_TARGET_TRIPLES_FOR_TAPI=YES
export VALIDATE_DEVELOPMENT_ASSET_PATHS=YES_ERROR
export VALIDATE_PRODUCT=NO
export VALIDATE_WORKSPACE=NO
export VALID_ARCHS="i386 x86_64"
export VERBOSE_PBXCP=NO
export VERSIONING_SYSTEM=apple-generic
export VERSIONLIST_PATH=app.framework/version.plist
export VERSIONS_FOLDER_PATH=app.framework
export VERSION_INFO_BUILDER=nishant
export VERSION_INFO_FILE=app_vers.c
export VERSION_INFO_STRING="@(#)PROGRAM:app PROJECT:iosApp-1""
export WRAPPER_EXTENSION=framework
export WRAPPER_NAME=app.framework
export WRAPPER_SUFFIX=.framework
export WRAP_ASSET_PACKS_IN_SEPARATE_DIRECTORIES=NO
export XCODE_APP_SUPPORT_DIR=/Applications/Xcode.app/Contents/Developer/Library/Xcode
export XCODE_PRODUCT_BUILD_VERSION=11C504
export XCODE_VERSION_ACTUAL=1130
export XCODE_VERSION_MAJOR=1100
export XCODE_VERSION_MINOR=1130
export XPCSERVICES_FOLDER_PATH=app.framework/XPCServices
export YACC=yacc
export arch=undefined_arch
export variant=normal
/bin/sh -c /Users/nishant/Library/Developer/Xcode/DerivedData/iosApp-dkftuslvqyreelbdsvignulguk/Build/Intermediates.noindex/iosApp.build/Debug-iphonesimulator/app.build/Script-F861D812207FA4320085E80D.sh
/Users/nishant/Library/Developer/Xcode/DerivedData/iosApp-dkftuslvqyreelbdsvignulguk/Build/Intermediates.noindex/iosApp.build/Debug-iphonesimulator/app.build/Script-F861D812207FA4320085E80D.sh: line 2: /Users/nishant/Desktop/kmp-examples/BasicKMPProject/iosApp/../gradlew: No such file or directory
Command PhaseScriptExecution failed with a nonzero exit code
```



Kickstart KMP Development

```
// Error
```

```
line 2: /Users/nishant/Desktop/kmp-examples/BasicKMPProject/iosApp/../../gradlew:
```

```
No such file or directory
```

```
Command PhaseScriptExecution failed with a nonzero exit code
```



Kickstart KMP Development

```
~/D/k/BasicKMPProject  master + ● ↑3
└─ ls
BasicKMPProject.iml  build          gradle.properties  local.properties
app                 build.gradle   iosApp             settings.gradle

~/D/k/BasicKMPProject  master + ● ↑3
└─ gradle wrapper

> Configure project :app
Kotlin Multiplatform Projects are an experimental feature.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
```

Kickstart KMP Development



Examples in the Wild



Apps in Production by

- ❑ [PlanGrid](#)
- ❑ [CashApp](#)
- ❑ [Careem](#)
- ❑ [VMware](#)
- ❑ [Quizlet](#)
- ❑ [Target](#)
- ❑ [SuperAwesome](#)

Apps in Production by

- ❑ [Infinut](#)
- ❑ [Touchlab/DroidconKotlin](#)
- ❑ [Jetbrains/Kotlinconf](#)
- ❑ [Jetbrain/Spaces](#)
- ❑ [Walpy](#)

Kotlin MultiPlatform Libraries

- ❑ Find a list here:

<https://github.com/AAkira/Kotlin-Multiplatform-Libraries>

Note:

- ❑ The ecosystem is still evolving
- ❑ Not many libraries at disposal
- ❑ Not all platforms supported in all libraries

Resources

- ❑ <https://play.kotlinlang.org/hands-on/Targeting%20iOS%20and%20Android%20with%20Kotlin%20Multiplatform/>
- ❑ <https://github.com/nisrulz/kmp-examples>

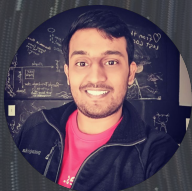


Thank You



twitter.com/nisrulz
github.com/nisrulz
www.nisrulz.com

Kotlin MP: Into the Multi-Verse



Nishant Srivastava
@nisrulz

twitter.com/nisrulz
github.com/nisrulz
www.nisrulz.com