Kotlin

Kotlin Multiplatform 跨平台开发的后起之秀

刘银龙

@kotlin | Developed by JetBrains

个人简介

美团 移动端开发工程师

一直从事餐饮收银软件的开发，涉及到 Android、iOS、Windows 等多种平台

GMTC北京202302：KMM 在美团餐饮 SaaS 中的探索与实践

Kotlin 炉边漫谈 第8期：阿里和美团的 Kotlin Multiplatform 应用案例

2023 KotlinConf Global 北京站：KMM 跨平台原理及实践

刘银龙

Kotlin Multiplatform
KMP

Kotlin Multiplatform
发展史

Kotlin： 1.2.0

2017

KMP： Experimental

Kotlin： 1.4.0

2020

KMM： Alpha

Kotlin： 1.7.20

2022

KMM： Beta

Kotlin： 1.9.20

2023

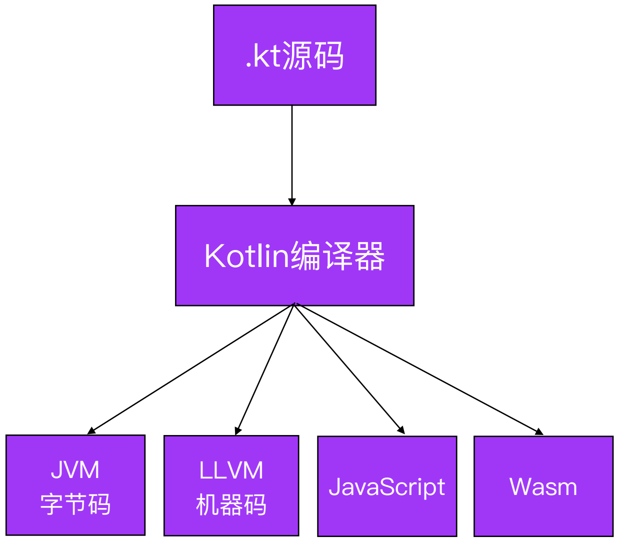
KMP： Stable

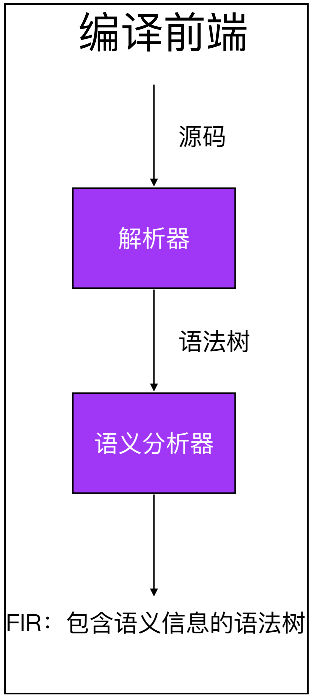
Kotlin Multiplatform
简介

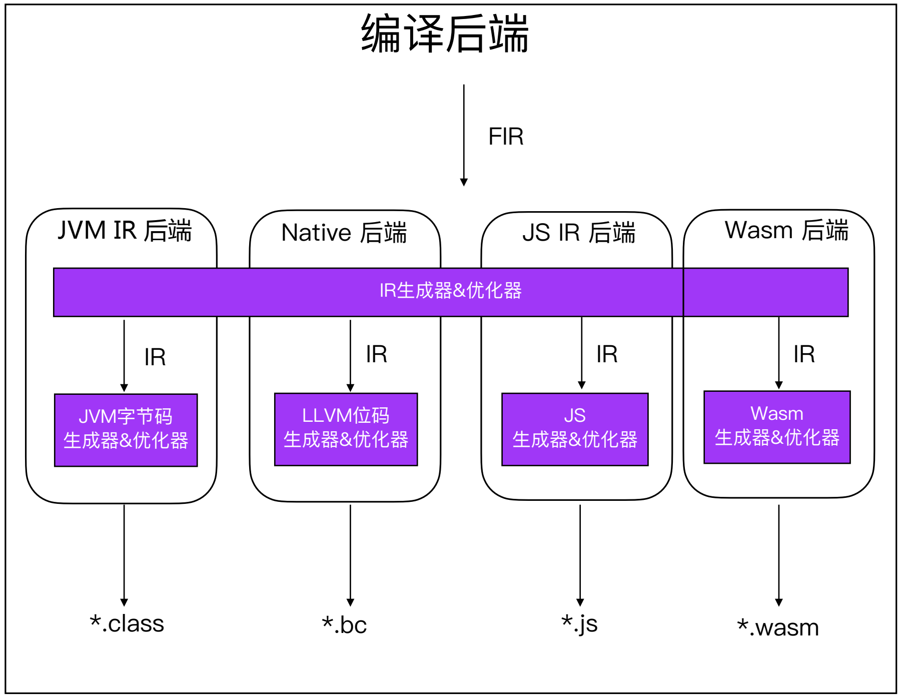
- Open-source technology by JetBrains for flexible multiplatform development
- Share code without compromising quality
- Suitable for all kinds of projects

Kotlin Multiplatform

跨平台原理–K2编译器

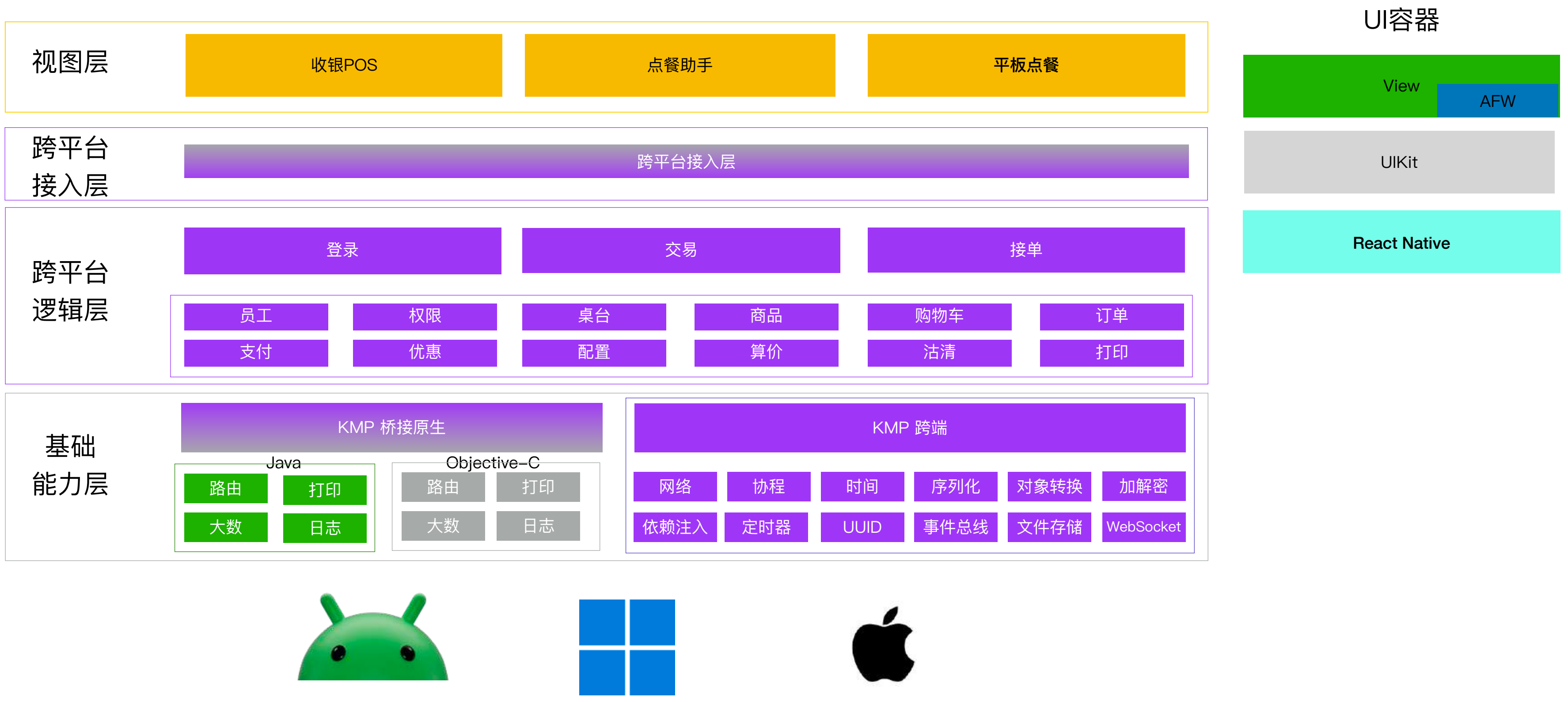






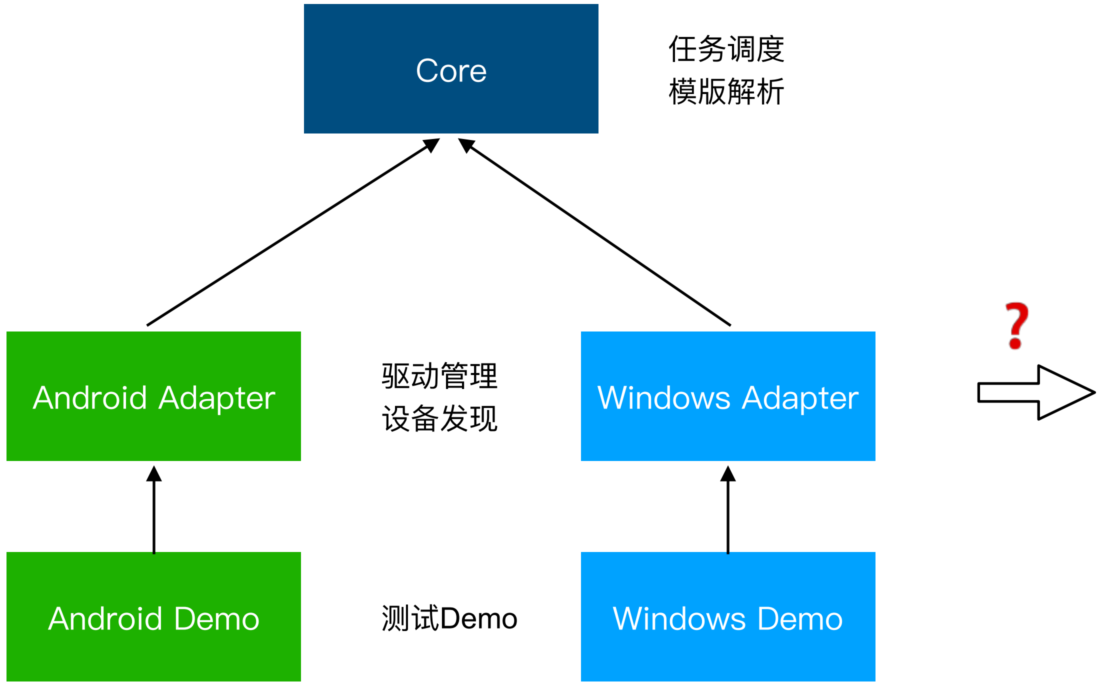
美团收银如何使用KMP做跨平台开发？

整体架构

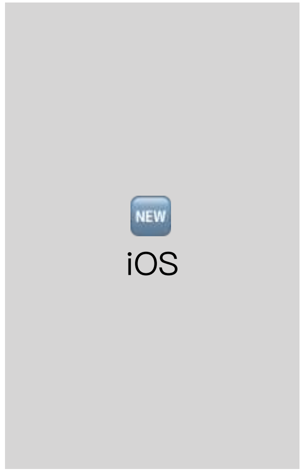

视图层
收银POS
点餐助手
平板点餐
跨平台接入层
跨平台接入层
跨平台逻辑层
登录
交易
接单
员工
权限
桌台
商品
购物车
订单
支付
优惠
配置
算价
沽清
打印
基础能力层
KMP 桥接原生
Java
路由
打印
大数
日志
Objective–C
路由
打印
大数
日志
KMP 跨端
网络
协程
时间
序列化
对象转换
加解密
依赖注入
定时器
UUID
事件总线
文件存储
WebSocket
UI容器
View
AFW
UIKit
React Native

基础能力层实践
打印SDK跨平台改造

打印SDK
背景

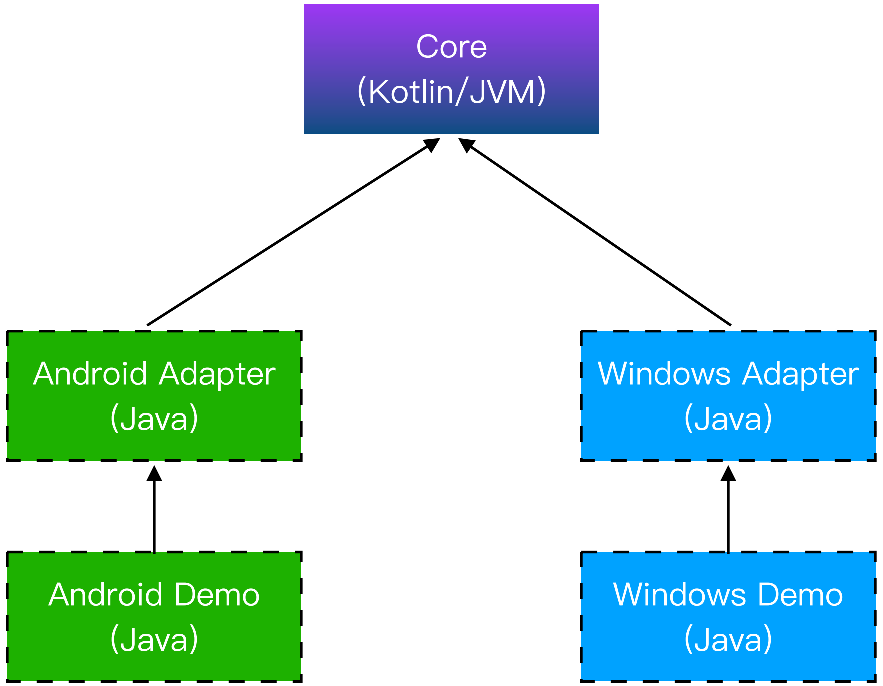


- "JVM" for iOS ❌
- ObjC/Swift 重写 ❌
- J2ObjC 转换 ❌
- KMP ✅



打印SDK

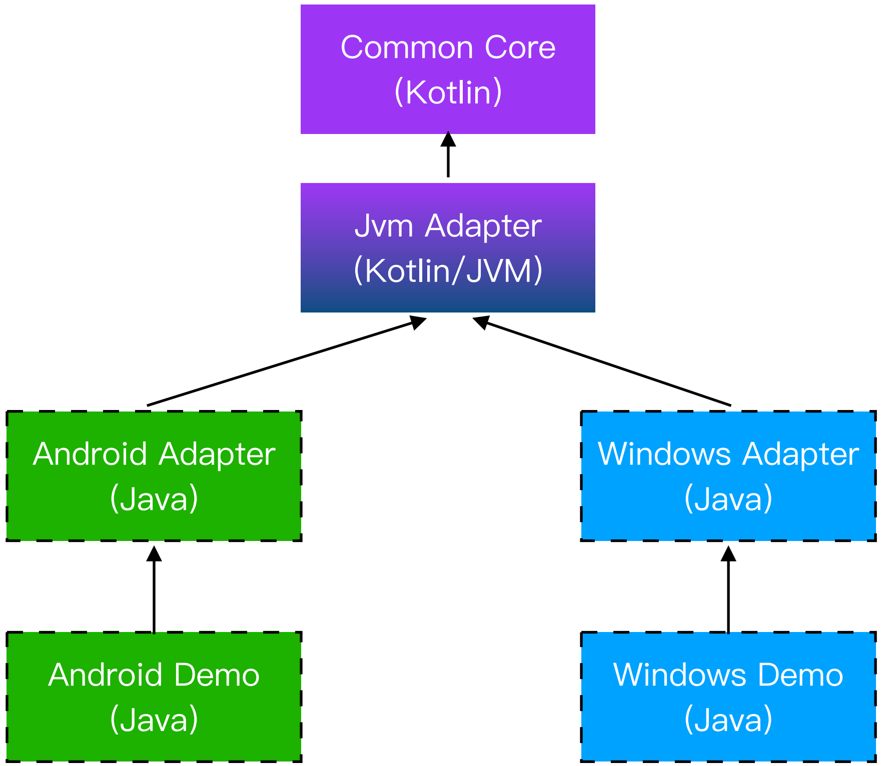
实施： Java –> Koltin/JVM



1. *.java –> *.kt
2. *.kt 错误修正
3. *.kt 编译通过
4. *.kt Demo验证

打印SDK

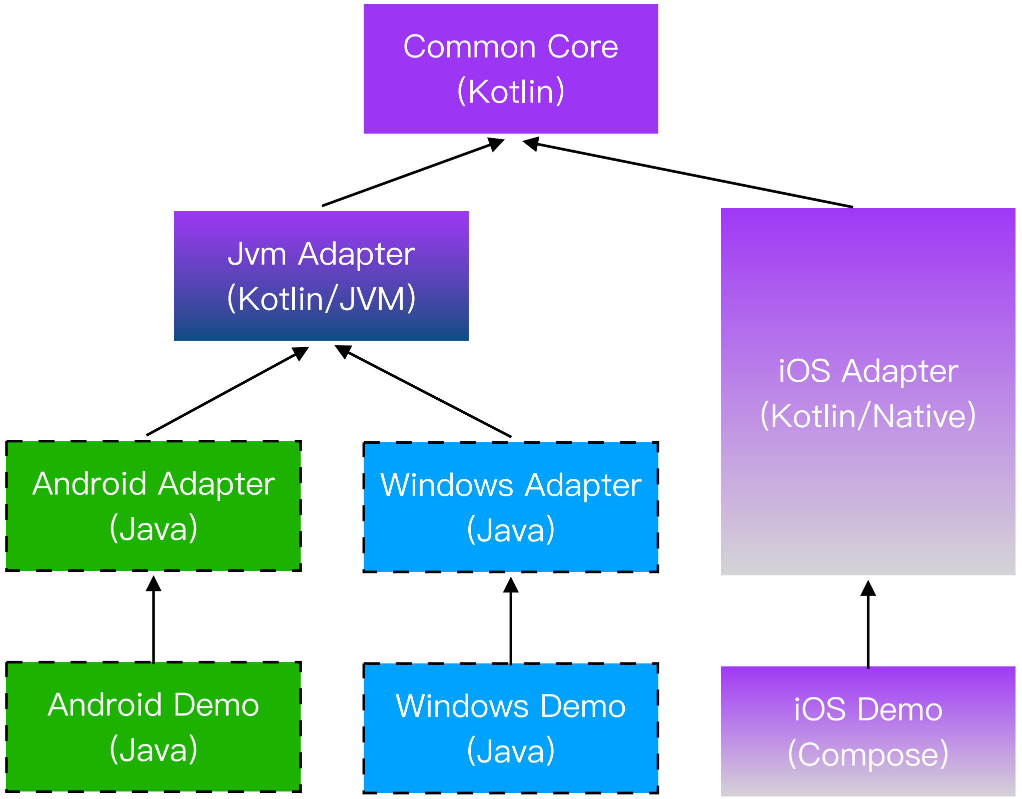
实施：Koltin/JVM –> Kotlin/Common



1. Common和Jvm边界确定
2. 改造方案调研设计
3. Gradle工程改造
4. 代码改造
5. 编译通过
6. Demo验证

打印SDK

实施：Kotlin/Native iOS 适配

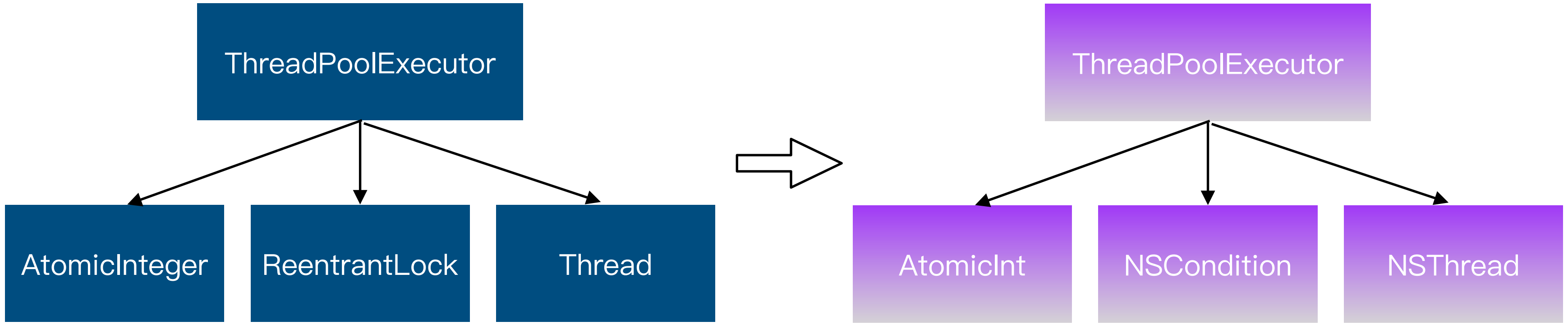


1. 适配方案调研设计
2. 核心功能适配(WiFi)
3. MVP Demo验证
4. 剩余功能适配（蓝牙）
5. 整体功能回归

打印SDK
实施：Kotlin/Native iOS 线程池适配

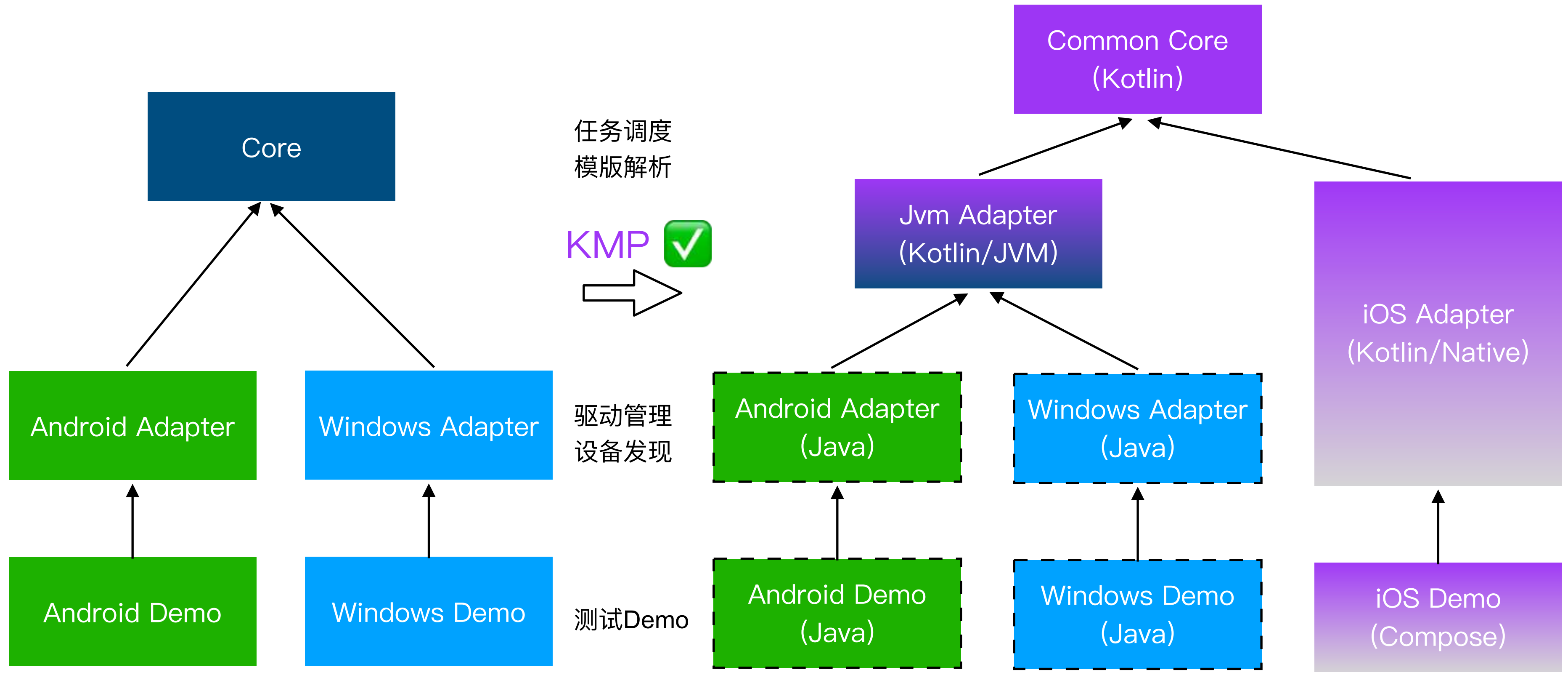
问题：Core层任务调度大量使用了线程池
方案：

- 替换为 Kotlin协程❌
- 抽象封装Jvm和iOS原生线程池❌
- “java.util.concurrent” for iOS ✅



打印SDK

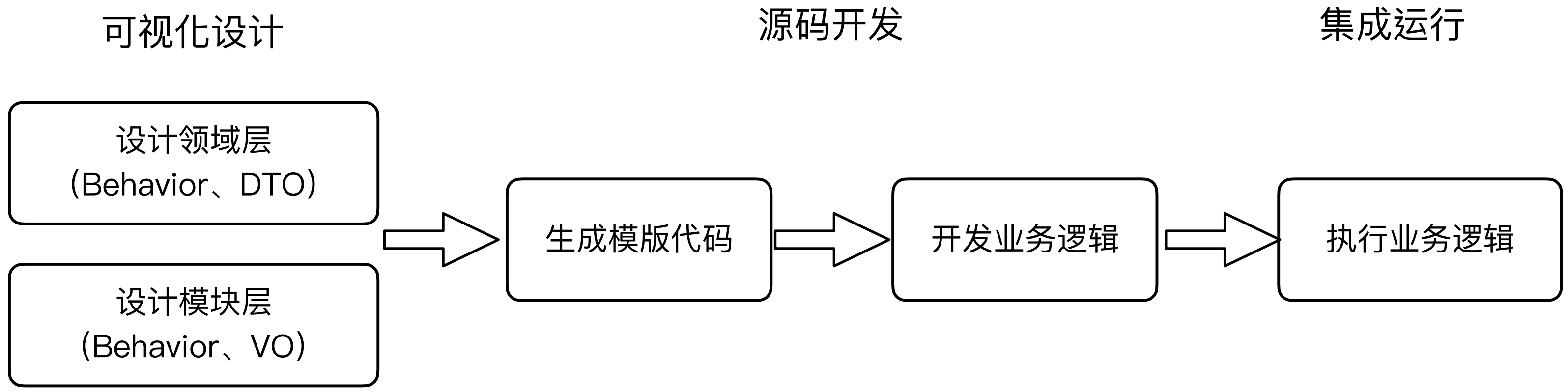
结果：跨平台基础能力



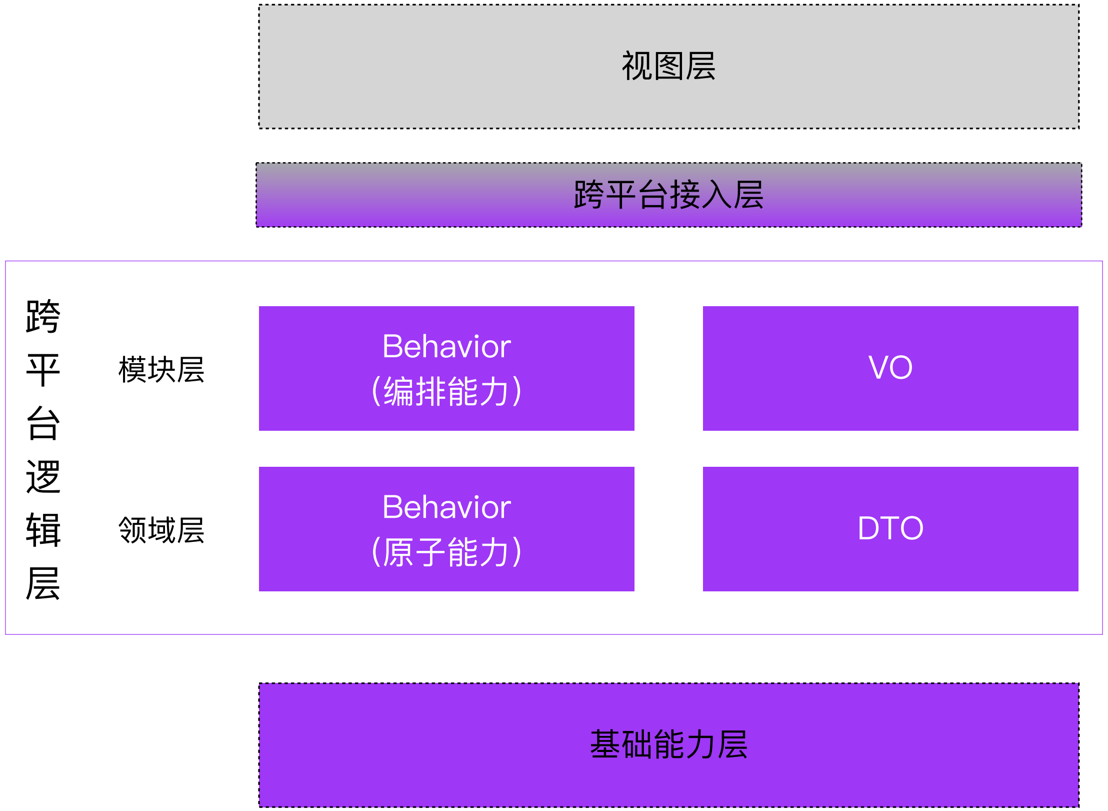
跨平台逻辑层实践

本地业务SDK

本地业务SDK
背景（搭建平台&脚手架）



本地业务SDK


视图层
跨平台接入层
跨平台逻辑层
模块层
Behavior (编排能力)
VO
领域层
Behavior (原子能力)
DTO
基础能力层

本地业务SDK

Behavior抽象

```kotlin
interface IBehavior<P, R> {
    @Throws(Exception::class)
    fun execute(param: P?): R

    fun getUri(): String
}
```

```kotlin
abstract class BaseBehavior<P, R> : IBehavior<P, R> {
   init {
        BehaviorHolder.regBehavior(this)
    }

    abstract fun beforeExecute(param: P?)

    @Throws(Exception::class)
    abstract fun doExecute(param: P?): R
    abstract fun afterExecute(param: P?)
    abstract fun finallyExecute(param: P?, res: R?)

    @Throws(Exception::class)
    override fun execute(param: P?): R {
        var res: R? = null
        try {
            beforeExecute(param)
            res = doExecute(param)
            afterExecute(param)
            return res
        } finally {
            finallyExecute(param, res)
        }
    }
}
```

本地业务SDK

领域层代码生成

```
// 基类模板方法
package com.meituan.kmp.domain.order

abstract class ConfirmOrderBehavior : BaseBehavior<ConfirmOrderParamDTO, ConfirmOrderResDTO>() {
    override fun getUri(): String = com.meituan.kmp.domain.order.confirmOrder”

    override fun beforeExecute(param: ConfirmOrderParamDTO?) {}

    @Throws(Exception::class)
    override fun doExecute(param: ConfirmOrderParamDTO?): ConfirmOrderResDTO {
        return confirmOrder(param)
    }

    @Throws(Exception::class)
    abstract fun confirmOrder(param: ConfirmOrderParamDTO?): ConfirmOrderResDTO

    override fun afterExecute(param: ConfirmOrderParamDTO?) {}
    override fun finallyExecute(param: ConfirmOrderParamDTO?, res: ConfirmOrderResDTO?) {}
}
```

本地业务SDK
领域层代码生成

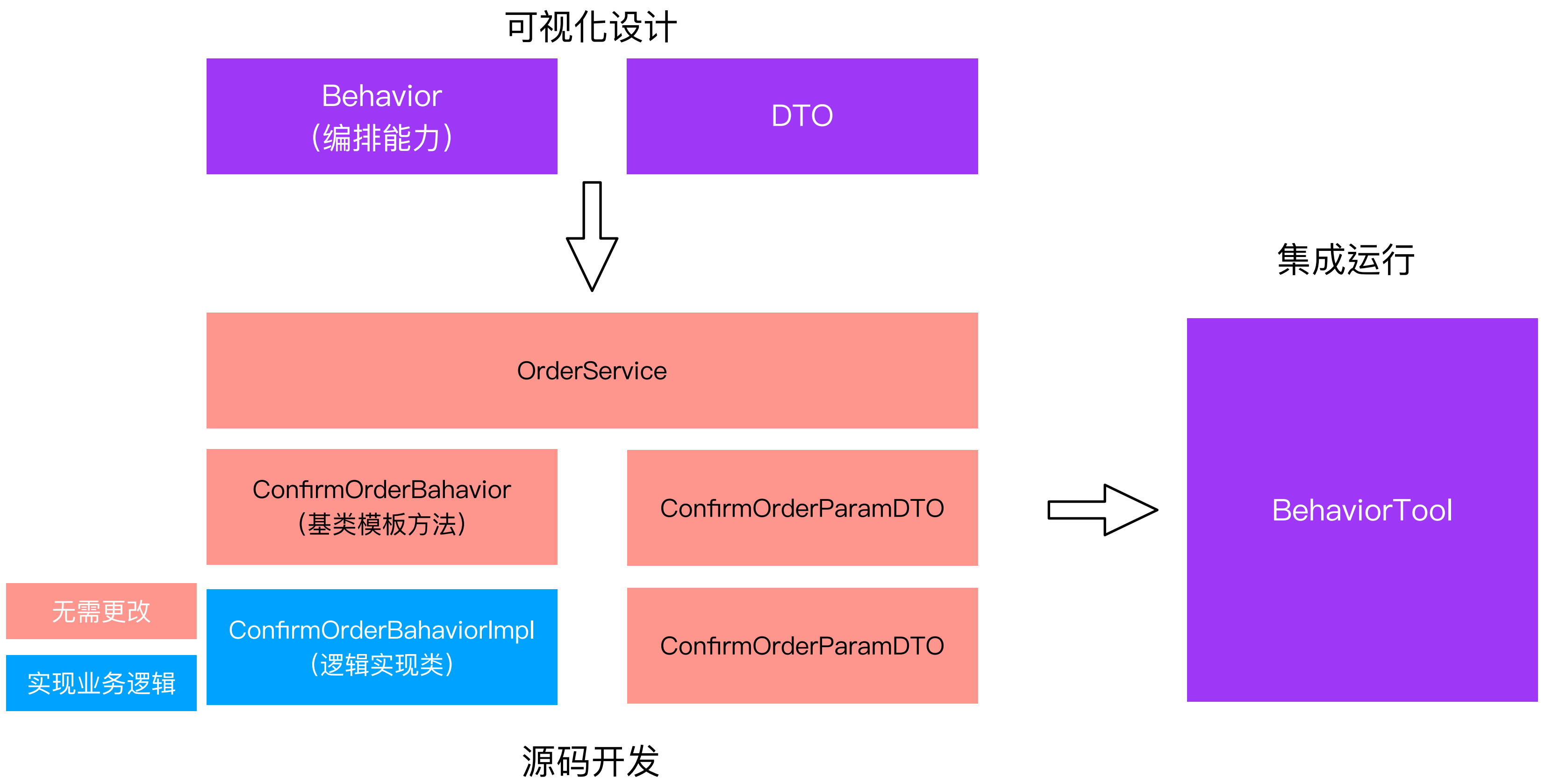
```
// 逻辑实现类
package com.meituan.kmp.domain.order

class ConfirmOrderBehaviorImpl : ConfirmOrderBehavior() {
    override fun confirmOrder(param: ConfirmOrderParamDTO?): ConfirmOrderResDTO {
        TODO("实现业务逻辑")
    }
}

// 逻辑调用门面类
package com.meituan.kmp.domain.order

object OrderService {
    private const val DOMAIN_URI = "com.meituan.kmp.domain.order"

    fun acceptOrder(param: ConfirmOrderParamDTO?): ConfirmOrderResDTO {
        return BehaviorTool.invoke("$DOMAIN_URI.acceptOrder", param)
    }
}
```

本地业务SDK

执行业务逻辑

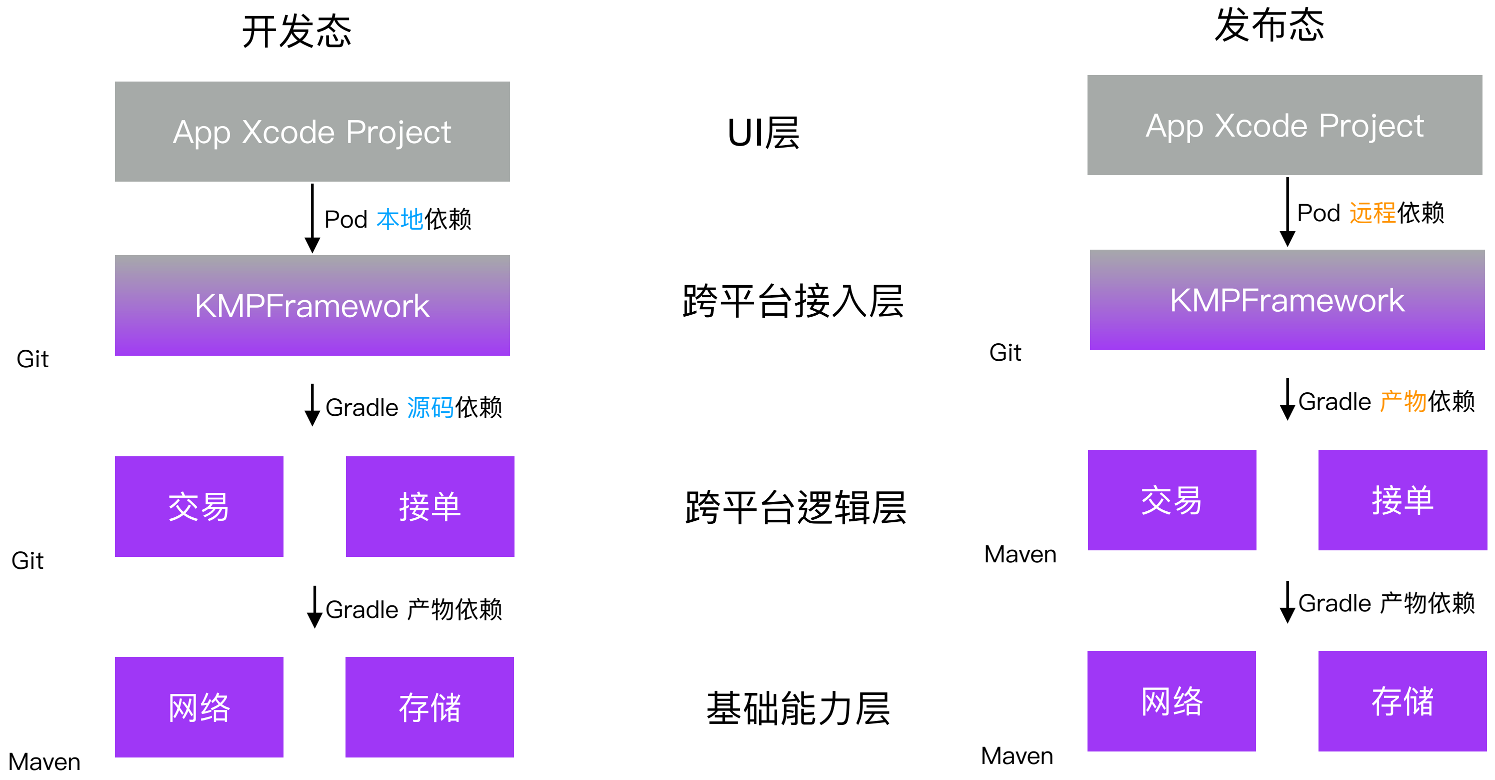
```
// 依赖注入管理
object BehaviorHolder {
    fun regBehavior(behavior: BaseBehavior<*, *>) {}

    fun <P, R> executeBehavior(behaviorRri: String, param: P?): R {}
}

// 执行逻辑
object BehaviorTool {
    @Throws(Exception::class)
    inline fun <reified P, reified R> invoke(behaviorUri: String, param: P?): R {
        return BehaviorHolder.executeBehavior<P, R>(behaviorUri, param)
    }
}
```

本地业务SDK


可视化设计
Behavior
(编排能力)
DTO
OrderService
ConfirmOrderBahavior
(基类模板方法)
ConfirmOrderParamDTO
无需更改
实现业务逻辑
ConfirmOrderBahaviorImpl
(逻辑实现类)
ConfirmOrderParamDTO
源码开发
集成运行
BehaviorTool

跨平台接入层实践
iOS集成

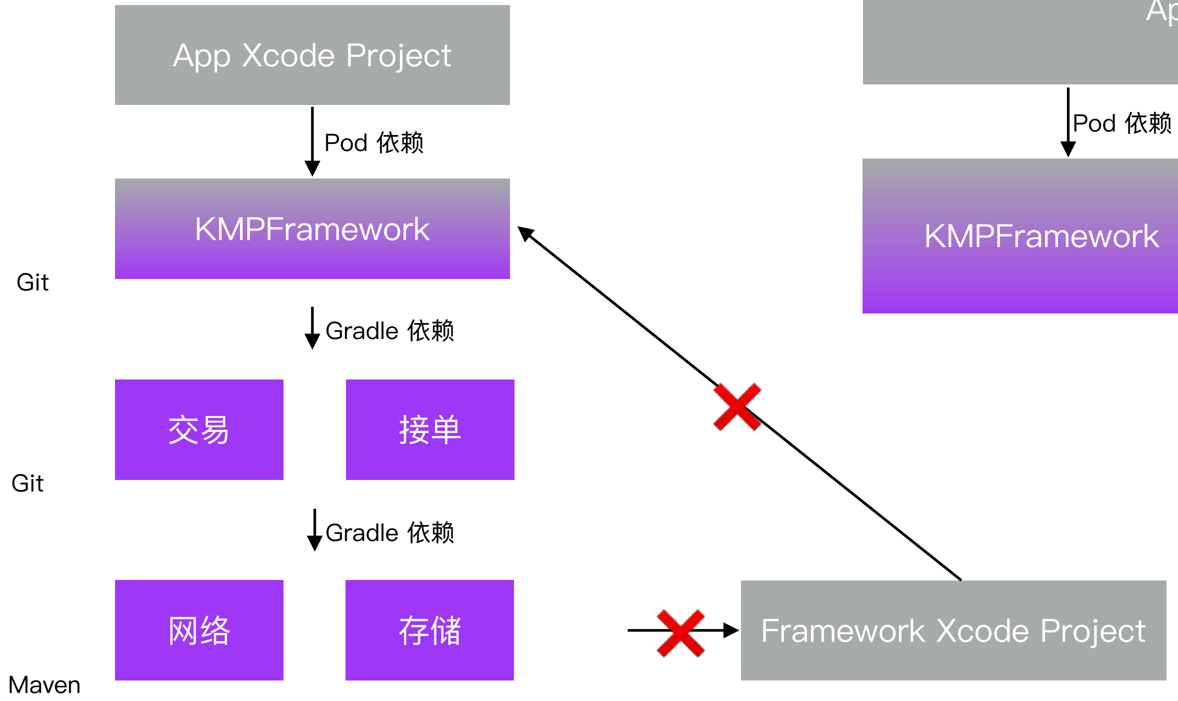
跨平台接入层



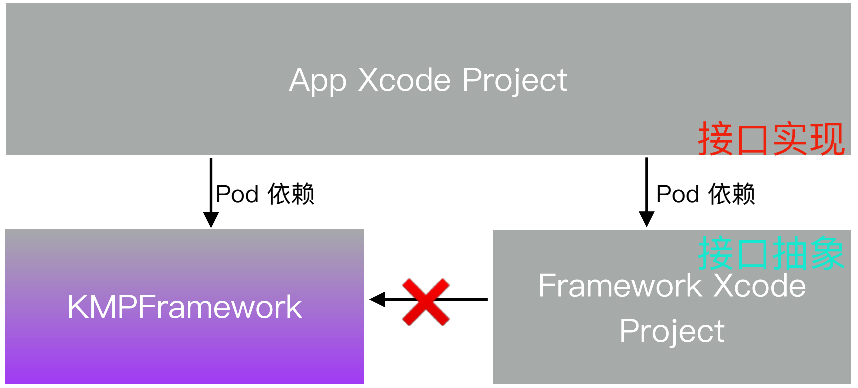
- Xcode + xcode-kotlin plugin
- Android Studio + KMM plugin

跨平台接入层

Objective-C 库调用 KMPFramework

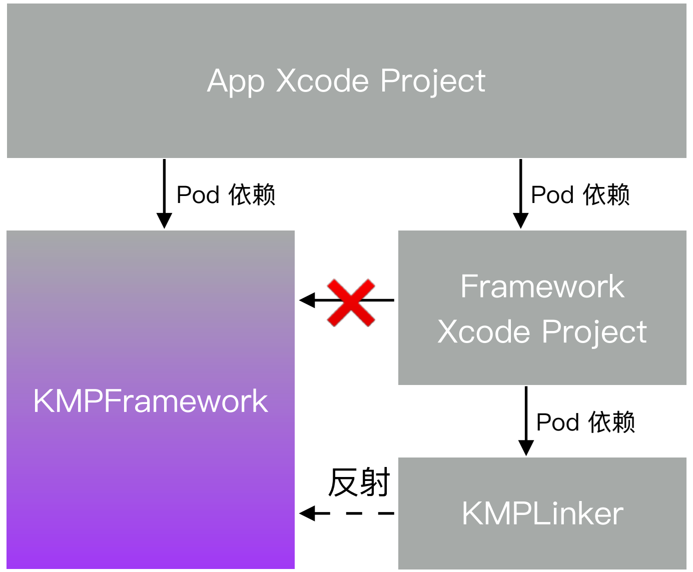


方案一：抽象接口



跨平台接入层
Objective-C 库调用 KMPFramework

方案二：反射



Objective-C

```
/// 通过反射获取[<className> shared]，其中className为类名
id findSharedInstance (NSString* className) {
    id sharedInstance = nil;
    Class aClass = NSClassFromString(className);
    if ([aClass respondsToSelector: @selector(shared)]) {
        sharedInstance = [aClass shared];
    }
    return sharedInstance;
}

// 强转为XXXXXProtocol协议的id
id<XXXXXProtocol> instance = findSharedInstance(@“KMPXXXXX")
[instance doXXXX];  // 从而调用doXXX方法
```

- NSClassFromString、conformsToProtocol
- NSSelectorFromString、respondsToSelector、performSelector

视图层(iOS)
函数重载

Kotlin

```
class Money @JvmOverloads constructor(
    val amount: Long,
    val currency: Currency = DEFAULT_CURRENCY,
) {
    constructor(
        amount: String,
        currency: Currency = DEFAULT_CURRENCY,
    ) : this(
        amount.toLongCent(), currency
    )
}
```

函数重载：amount参数类型不同

Objective-C

```
@interface KMPMoney : KMPBase

- (instancetype)initWithAmount:(int64_t)amount
currency:(KMPCurrency *)currency
;

- (instancetype)initWithAmount:(NSString *)amount
currency:(KMPCurrency *)currency_
;

@end
```

编译器会自动给最后一个参数名添加下划线，
但每次添加规则有可能不一致

视图层(iOS)
函数重载(@ObjCName)

Kotlin

```kotlin
class Money @JvmOverloads constructor(
    @ObjCName("longAmount") val amount: Long,
    val currency: Currency = DEFAULT_CURRENCY,
) {
    constructor(
        @ObjCName("stringAmount") amount: String,
        currency: Currency = DEFAULT_CURRENCY,
    ) : this(
        amount.toLongCent(), currency
    )
}
```

@ObjCName 自定义amount参数名

Objective-C

```objc
@interface KMPMoney : KMPBase

- (instancetype)initWithLongAmount:(int64_t)longAmount
currency:(KMPCurrency *)currency
;

- (instancetype)initWithStringAmount:(NSString *)stringAmount
currency:(KMPCurrency *)currency
;

@end
```

生成不同签名的“重载”函数

视图层(iOS)
默认值参数(SKIE)

Kotlin

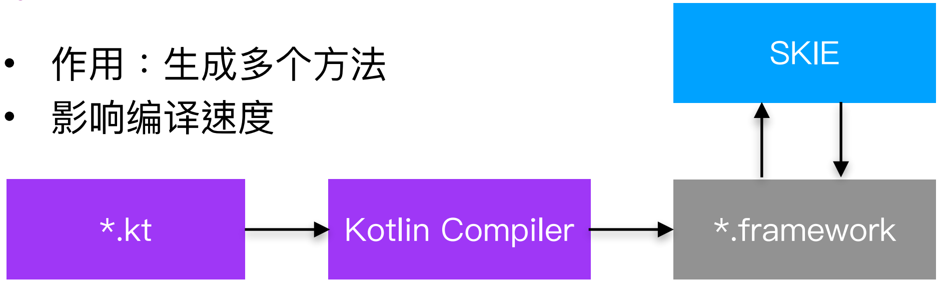
```
class Money
@JvmOverloads
@DefaultArgumentInterop.Enabled
constructor(
    @ObjCName("longAmount") val amount: Long,
    val currency: Currency = DEFAULT_CURRENCY,
) {
    constructor(
        @ObjCName("stringAmount") amount: String,
        currency: Currency = DEFAULT_CURRENCY,
    ) : this(
        amount.toLongCent(), currency
    )
}
```

SKIE: @DefaultArgumentInterop.Enabled

Objective-C

```
@interface KMPMoney : KMPBase

- (instancetype)initWithLongAmount:(int64_t)longAmount
currency:(KMPCurrency *)currency
;

- (instancetype)initWithLongAmount:(NSString *)longAmount
;

// …

@end
```

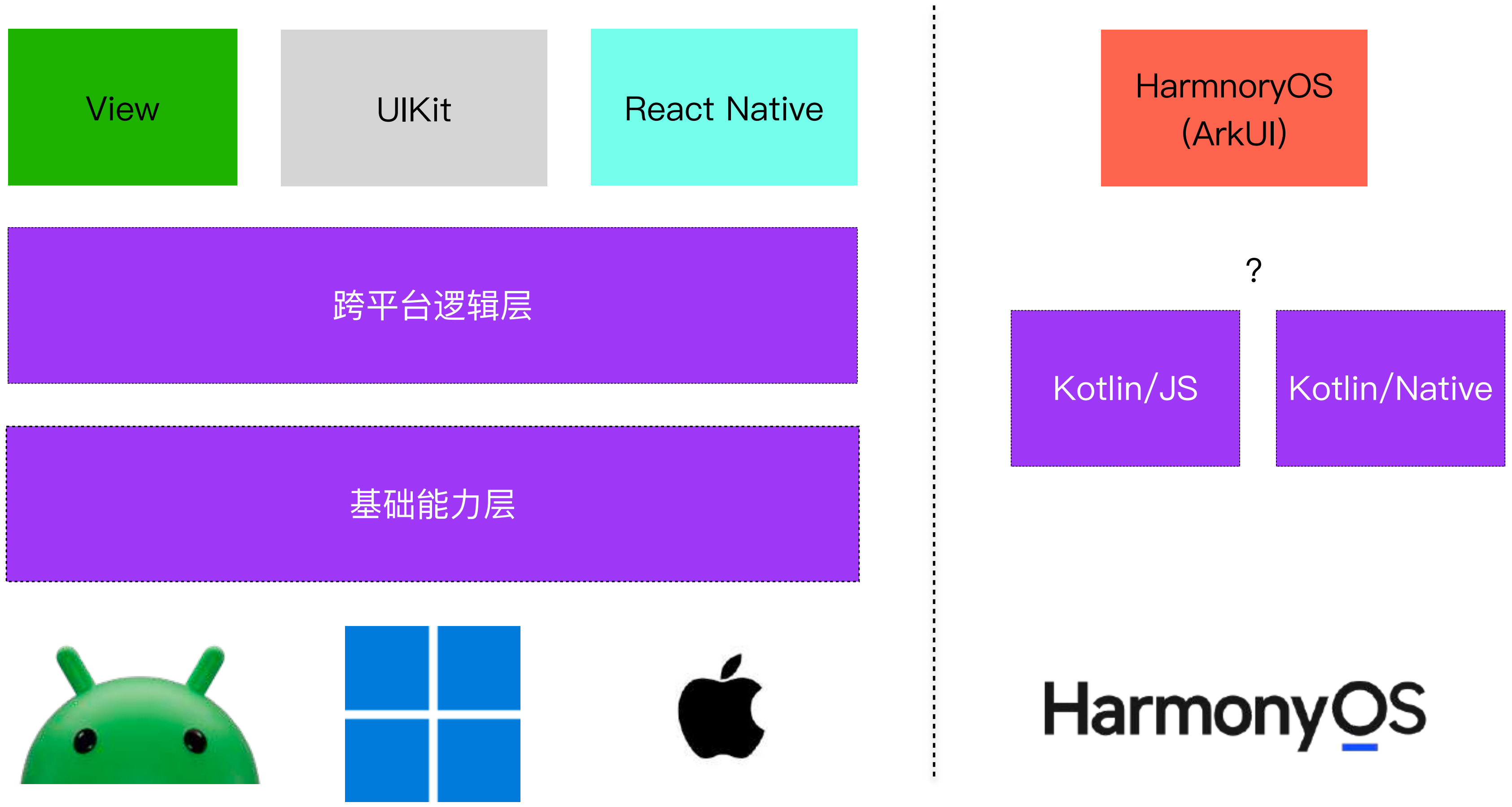
- 作用：生成多个方法
- 影响编译速度



KMP未来展望

跨平台UI展望

进一步统一技术栈？



跨平台UI展望

Compose Multiplatform

Jetpack Compose

Alpha	Beta	1.0	1.1	1.2	1.3	1.4	1.5
2020/8	2021/2	2021/7	2022/2	2022/7	2022/10	2023/1	2023/8

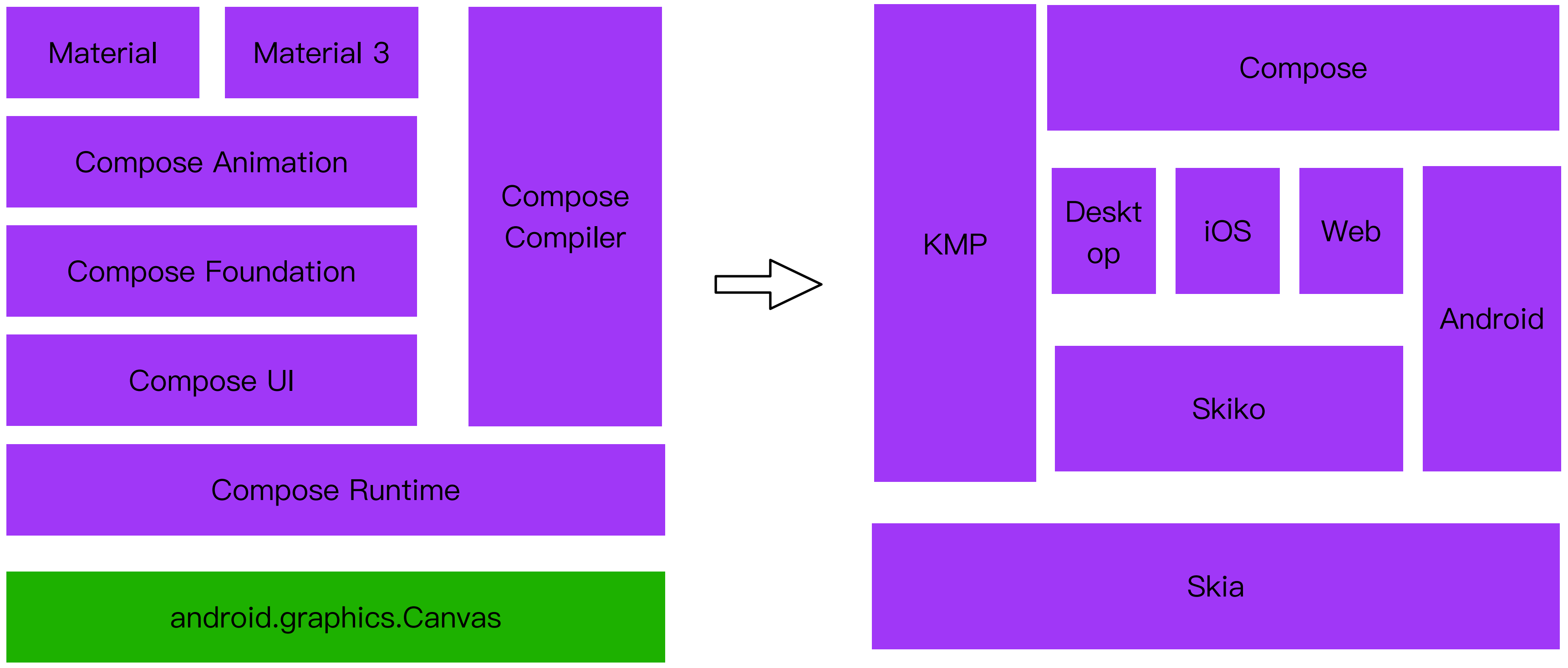
2020/11	2021/5	2021/12	2022/2	2022/10	2023/1	2023/4	2023/9	2023/11
Desktop M1	Web Preview	Desktop 1.0	Desktop 1.1	Desktop 1.2	Desktop 1.3	MP 1.4 iOS Alpha Web/Wasm Exp	MP 1.5	MP 1.5.10

Compose Multiplatform

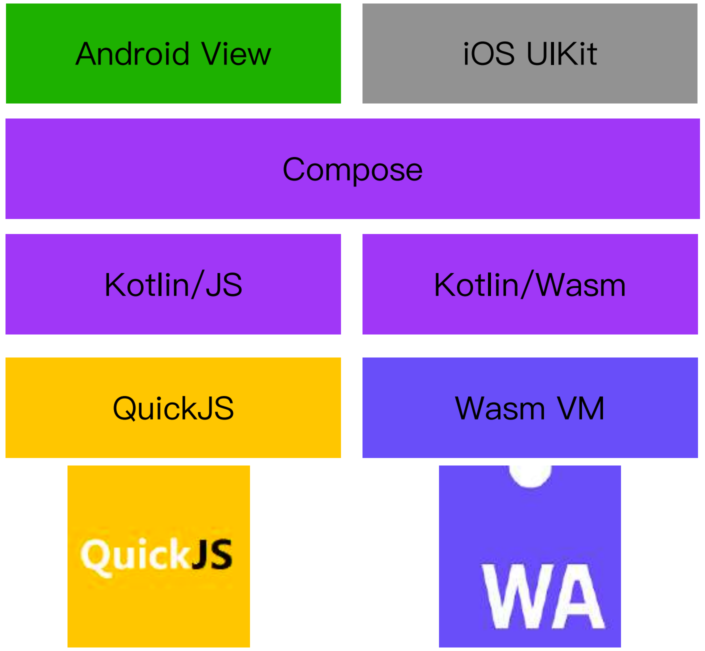
MP: Multiplatform
Exp：Experimental

跨平台UI展望

Compose Multiplatform



动态化展望



Redwood

- 原生渲染
- 复用已有组件
- 原生开发语言和工具
- 支持逐步迁移

Zipline

- QuickJS: 轻量且高效
- AOT: 预编译为字节码
- 首屏优化：模块化、异步下载、缓存、预置包

2024官方路线图

- Compose Multiplatform
  - for iOS to Beta
  - for Web to Alpha
- Tooling
  - Fleet
- Multiplatform core
  - direct Kotlin-to-Swift export
- Library ecosystem

常用学习资料

官方文档：Get started with Kotlin Multiplatform

源码：https://github.com/JetBrains/kotlin

Kotlin blog：https://blog.jetbrains.com/kotlin

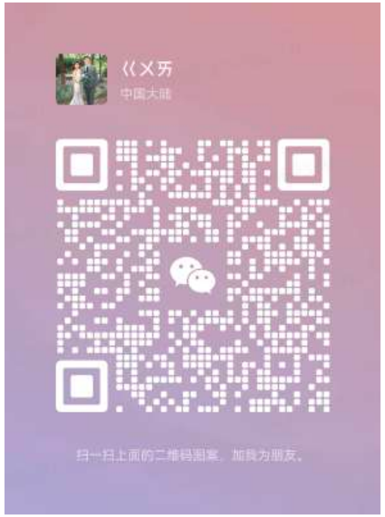
Kotlin Slack #multiplatform channel：https://kotlinlang.slack.com/archives/C3PQML5NU

YouTrack：https://youtrack.jetbrains.com/issues/KT

Kotlin Weekly：http://www.kotlinweekly.net

开源社区：https://github.com/AAkira/Kotlin-Multiplatform-Libraries
https://github.com/terrakok/kmm-awesome

微信群：国内KMM技术交流群、北京 Kotlin 用户组交流群



Thanks!
Starting. Fun. Love Kotlin

@kotlin I Developed by JetBrains