

# Jetpack来了：走近Google标准应用架构



博文视点...  
已认证的官方帐号

开发应用程序就像搭积木。没有良好架构的应用程序，就像没有搭好底座的积木，随着项目复杂度的上升，维护起来会困难重重，工程师会不停地陷入技术债务之中——“积木的倒塌”只是时间问题。

如何把握模块的粒度，在保持模块独立性的同时，又不影响模块间的通信，是全世界优秀的Android工程师共同追求的目标。为了解决这一问题，各类架构模式层出不穷，比较著名的有MVC、MVP和MVVM。**Jetpack**正是在这一背景下诞生的。它由Google官方推出，用于方便工程师搭建符合MVVM规范的Android应用程序。

本文选自《**Android Jetpack应用指南**》一书，让我们跟随下文一同走近Google标准应用架构。

## 01 | Android应用程序架构设计标准的缺失

一个Android应用程序通常至少有一个Activity，当我们要开发一个小型Android应用程序时，通常会将大部分的代码写在Activity/Fragment中。这些代码包括业务逻辑、数据Model、UI控件等。当涉及网络数据获取或数据库CRUD（Create、Retrieve、Update、Delete，即增加、查询、更新、删除）操作时，还需要用到工作线程，进而，我们还不得不考虑Activity/Fragment的生命周期问题。

针对一个小型项目，将大部分代码写在Activity/Fragment中并没有什么问题，但对于中大型项目而言，随着时间的推移和业务复杂度的增加，Activity/Fragment中的代码会变得复杂且难以维护。因此，我们需要将代码按照功能或类型的不同进行分类，并放到不同的包或类文件中，但又不破坏彼此正常的功能和通信。

这在软件开发中叫作“**解耦**”。为了将代码解耦以应对日益膨胀的代码量，工程师在应用程序中引入了“**架构**”的概念。使之在不影响应用程序各模块组件间通信的同时，还能够保持模块的相对独立。这样不仅有利于后期维护，也有利于代码测试。

关于架构，相信大家或多或少都听说过MVC（Model View Controller）、MVP（Model View Presenter）和MVVM（Model View ViewModel）。

在Android应用程序开发中，一直以来都有用到MVC，将Activity/Fragment与布局文件分开就是一种最简单、最基本的MVC思想，只是它没有很好地解决我们的问题，所以才有了MVP和MVVM。

由于Google官方并没有推出关于Android应用程序架构设计的标准，因此，世界各地的工程师只能自己创造各种解决方案，但这些方案都面临着以下问题。

- **非Google官方解决方案：**

由于不是Google官方解决方案，所以工程师不敢轻易在自己的线上项目中使用这些方案，除了害怕引入未知问题，更重要的是担心这些解决方案后期是否有开发者持续跟进维护。

- **无法辨别最佳解决方案：**

Android的应用架构始终处于一个混乱的阶段，Android工程师很困惑，他们不确定自己使用的架构是否真的是最佳方案。这不仅增加了工程师的学习成本，还可能最终导致他们开发出的应用程序质量参差不齐。

Android工程师希望Google官方可以推出并维护一些关于架构的组件或指南，这样他们就可以将更多的精力放在自己的业务代码上了。Google也意识到了这个问题，这便有了**Jetpack**，Jetpack正是为了解决这些问题而诞生的。

## 02 | 什么是Jetpack

前面提到，Jetpack是Google为了解决Android架构问题而引入的，但实际上Jetpack能做的不止这些。

按照Google官方的说法：

Jetpack是一套库、工具和指南，可以帮助开发者更轻松地编写应用程序。Jetpack中的组件可以帮助开发者遵循最佳做法、摆脱编写样板代码的工作并简化复杂的任务，以便他们能将精力集中在业务所需的代码上。

Jetpack主要包括4个方面，如下图所示，分别是**架构**（Architecture）、**界面**（UI）、**行为**（Behavior）和**基础**（Foundation）。



### 03 | Jetpack 与 AndroidX

在2018年的Google I/O大会上，Google宣布用AndroidX代替Android SupportLibrary，AndroidSupport Library在版本28之后就不再更新了，未来的更新会在AndroidX中进行。不仅如此，AAC（Android Architecture Component）中的组件也被并入AndroidX。所以，当使用Jetpack的组件时，经常会看到以“androidx”开头的包名。

下图从包名的变化，我们便可以看出，AndroidSupport Library与AAC中的各种组件已经迁移到了AndroidX中。

AAC/Android Support Library	AndroidX
...	...
android.arch.persistence.room:rxjava2	androidx.room:room-rxjava2:2.0.0-rc01
android.arch.persistence.room:testing	androidx.room:room-testing:2.0.0-rc01
android.arch.persistence.db	androidx.sqlite:sqlite:2.0.0-rc01
android.arch.persistence.db-framework	androidx.sqlite:sqlite-framework:2.0.0-rc01
com.android.support.constraint:constraint-layout	androidx.constraintlayout:constraintlayout:1.1.2
com.android.support.constraint:constraint-layout-solver	androidx.constraintlayout:constraintlayout-solver:1.1.2
...	...

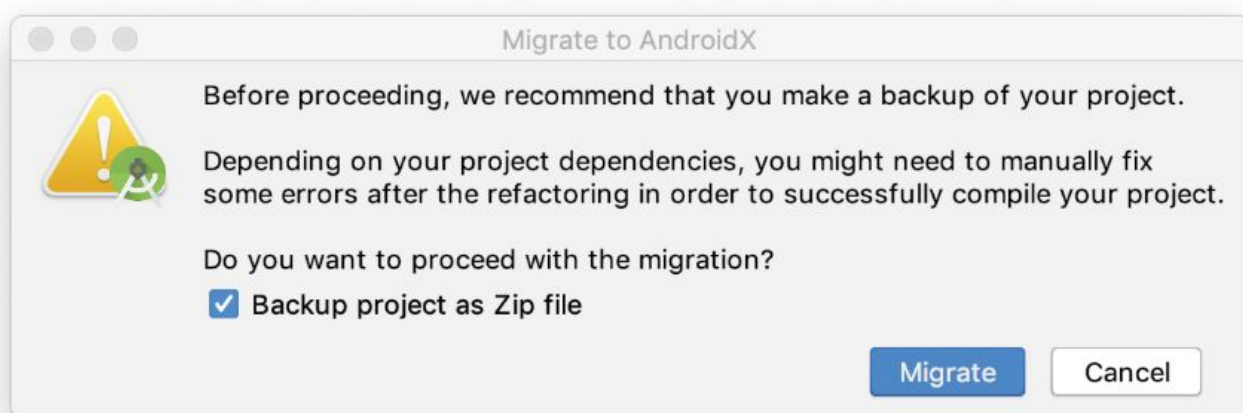
**为什么Jetpack组件需要以兼容包的形式存在，而不是成为Framework的一部分呢？**

很简单，这是为了提供向后兼容，使Jetpack组件能够应对更加频繁的更新。除了Android Support Library和AAC，其他一些需要频繁更新和迭代的特性也被并入了AndroidX，例如Emoji。

## 04 | 迁移至AndroidX

如果你从未在项目中使用过Jetpack组件，现在你希望将项目迁移至AndroidX，那么可以在菜单栏中选择 Refactor → Migrate to AndroidX... 选项，将你的项目迁移至AndroidX。

此时，会出现一个对话框，询问迁移之前是否需要以Zip文件的形式备份项目，如下图所示。这里建议备份一份，以防迁移出错。




## 05 | 新建项目默认支持AndroidX

如果你的Android Studio为最新版本，那么在新建一个项目时，应该能在创建过程中看到“**Useandroidx.\* artifacts**”这个选项。这表示，新创建的项目会默认配置对AndroidX的支持，如下图所示。

Create New Project

## Configure your project



Name

Package name

Save location

Language


Java

Minimum API level

API 14: Android 4.0 (IceCreamSandwich)

Empty Activity

Creates a new empty activity

 Your app will run on approximately **100%** of devices.

[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.\* artifacts

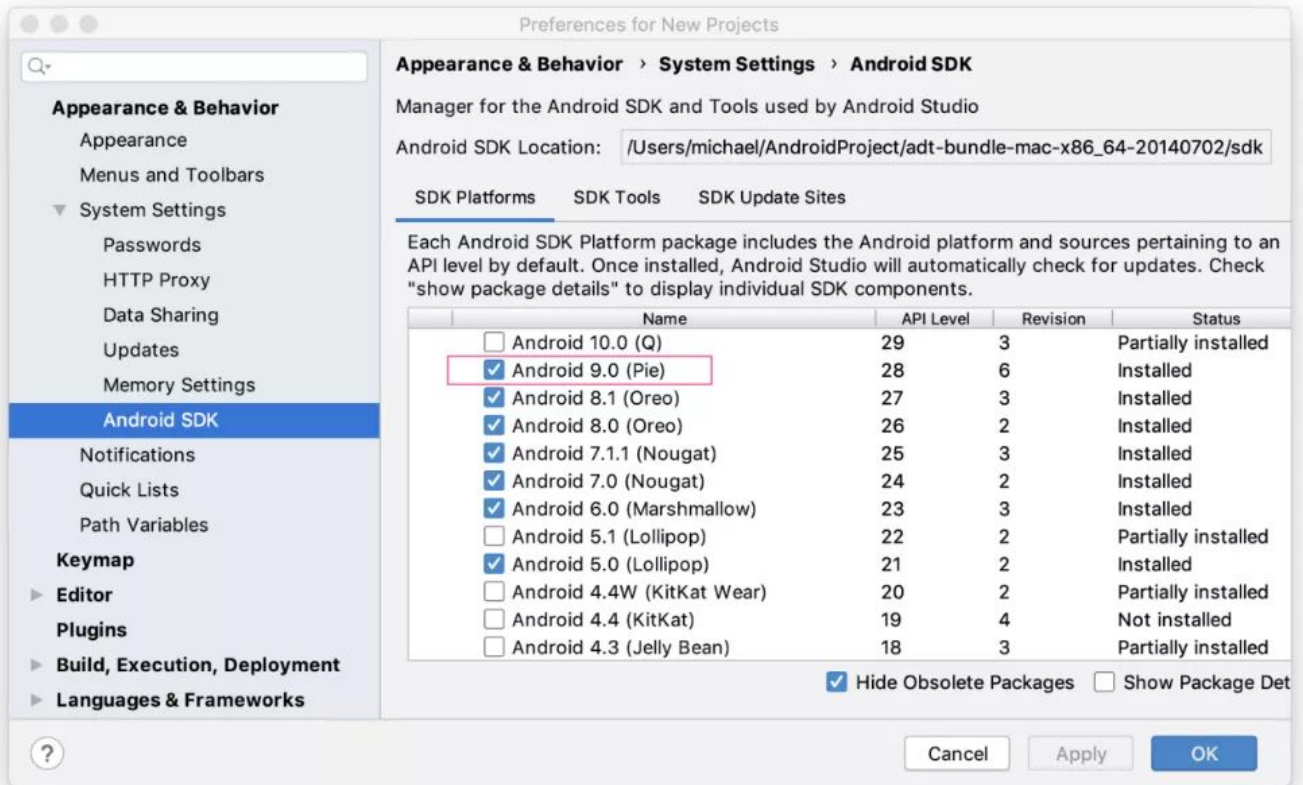
Cancel

Previous

Next

Finish

如果没有看见此选项，那么请检查你的SDK配置。通过 Tools → SDK Manager 打开配置界面，确保你已经安装了Android 9.0及以上版本的SDK。



本文选自博文视点新书《Android Jetpack应用指南》。

发布于 2020-07-14

Android 开发

谷歌 (Google)

架构