


Android 架构组件的最新进展 (上篇)



谷歌开发者 
已认证的官方帐号

已关注

23 人赞同了该文章

根据我们曾经做的调查，开发者们希望 Android 官方可以维护一些实用的组件库和架构实践，以降低中大型应用的开发门槛，这样开发团队就可以集中更多精力在实际业务的优化和改进上。

Jetpack 项目正是为了解决这些问题而诞生的，Jetpack 是一系列助力您更容易打造优秀 Android 应用的工具和组件，这些组件能帮助您遵循最佳实践、免除编写繁复的样板代码并简化复杂任务，从而使您可以专注于最核心的代码逻辑。其中 `androidx.*` 库与 Framework API 解耦，这能够提供向后兼容的同时，也能更频繁地更新。

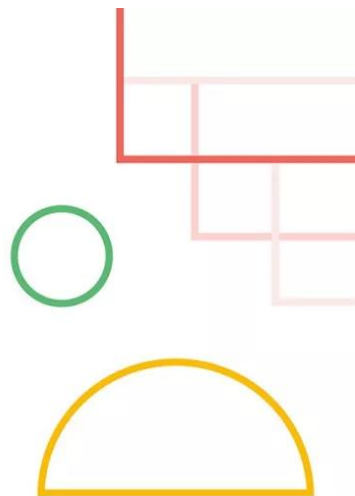
Android Jetpack 中的**架构组件**可帮助您设计稳健、可测试且易维护的应用。从最初发布的管理 Activity 和 Fragment 生命周期的 **Lifecycle 库**和访问 SQLite 数据库的**Room 库**，后来推出了分页 (**Paging**)、导航 (**Navigation**) 和管理后台任务的**WorkManager 库**。

根据 2019 年最新的开发者调查中，70% 以上的专业开发者用过这五个库当中的至少一个库进行应用开发，接下来我们将用上、下两篇文章为大家介绍 Android **架构组件**的最新更新：

Android 架构组件的最新进展

What's New in
Architecture Components

Part 1 of 2



数据绑定库

数据绑定 (Data Binding) 库是一种支持库，借助该库，您可以使用声明性格式 (而非程序化地) 将布局中的界面组件绑定到应用中的数据源。数据绑定可以理解为代码和 XML 标记语言之间的桥梁。

更快的编译速度

对开发者来说，处理界面中的数据绑定需要的时间成本不容忽视，我们现在将数据绑定注解处理的速度提高了 20%*。

* Google 内部实测结果。

如果您的工作涉及到协同开发，还有一个好消息，那就是我们增加了对分布式构建缓存的支持。

最后，数据绑定支持增量注解处理，能提升增量构建的性能。但这个功能还停留在测试阶段，所以请在 Gradle 配置文件中手动打开：

实时生成类代码

现在，给视图中的某个控件赋予 ID，它就会在绑定的类中成为一个可用的字段。或者直接在 XML 中设置一个变量，并在视图中访问，代码也能马上给出对应的提示。这些都即时可用，无需编译！



更好地支持重构

在 IDE 里使用重构的方式修改函数名称之后，XML 中会同步进行更新。



更好用的报错信息

数据绑定出错的信息可能一下子跳出来 1,000 条，这种尴尬将成为过去。现在在构建输出信息窗口中，数据绑定错误单独成组，这样开发者能更轻松地找到自己需要处理的错误信息。

有没有更好的视图访问方式？



视图访问方法当然不止一种，但正如上图所示的，在简明、编译安全和编译速度上，各个方法总有取舍。那有没有一种方法能一石 "三" 鸟呢？

即将到来！视图绑定 (View Binding)

给出 ID 即可自动生成绑定类代码且能保证编译安全，能做到一石 "三" 鸟的视图绑定可在 Android Studio 3.6 Canary 11 或更新版本中用得上。

```
<!-- profile.xml -->
<LinearLayout>
    <TextView android:id="@id/title"/>
    <ImageView android:id="@id/photo"/>
</LinearLayout>

class ProfileActivity: AppCompatActivity {
    override fun onCreate(savedInstanceState: Bundle?) {
        val binding = ProfileBinding.inflate(layoutInflater)
        setContentView(binding.root)
        // binding.title:TextView
        // binding.photo:ImageView
    }
}
```

△ 在生成的绑定类 inflate 之后，即可运行 setContentView，如果绑定的某个类型的控件不存在则无法编译。是时候告别 findViewById 了

所有的这些绑定类均由 Gradle 插件生成，如果开发者修改了某个布局文件，会报错的也会只有这个文件，100% 编译安全。

处理生命周期

"ViewModel 和 SavedState 一样吗？ViewModel 会破坏 SavedState 吗？"——很多开发者会这么问

基本上，开发者会通过 ViewModel 或着 SavedState 来保存自己的内容/状态，当应用配置发生变化时再从 ViewModel 或者 SavedState 中取回保存的内容/状态：



如果只这样粗略地理解的话，ViewModel 和 SavedState 其实是一回事。然而并不是这样的。

SavedState 会经由 System Server (一个独立的进程) 保存内容 (序列化的数据)，也就是说，它会无视进程的限制。

而 ViewModel 则一直运行于进程内，即便应用配置发生变化，只要进程还在，ViewModel 保存的内容就不会消失。但只要进程消失，ViewModel 里的内容也会消失。

ViewModel 用于：

- 保留应用对网络、数据库的请求
- 当作大型对象的缓存

SavedState 用于：

- UI 的状态记录，比如选择区域和滚动距离等
- 导航状态键值记录

各取所长，联手打造流畅体验

```
// SavedStateHandle
class UserViewModel(val handle: SavedStateHandle) : ViewModel() {
}
```

现在用户的 ViewModel 会在构造函数中接收一个 SavedStateHandle，这样开发者就能在 ViewModel 中马上访问 SavedState。

而这个 SavedStateHandle 内部的逻辑也非常直白：一个 Map 类的键值结构。当然，也提供了 LiveData 供访问，只不过在这里使用的是 MutableLiveData (因为 SavedState 是可变的)。

```
// map-like object
val handle : SavedStateHandle

// read
val myValue : Int = handle.get("key")

// write
handle.put("key", newValue)

// or
val liveData : MutableLiveData<Int> = handle.getLiveData("key")

// observe as usual
liveData.observe (lifecycleOwner) { value -> }

// write
liveData.value = newValue
```

更 Kotlin 友好的代码

我们会持续确保 Kotlin 语言的首选开发语言地位。其中一个例子就是 liveData.observe 现在支持 lambda 表达式：

```
// lifecycle-livedata-ktx

liveData.observe(lifecycleOwner) { newValue ->
}
```



```
// lifecycle-livedata-ktx

// 以前
val mapped = Transformations.map(liveData) {
    user -> user.name
}

// 现在
val mapped = liveData.map { user -> user.name }
```

ViewModel 的初始化也大幅精简，以前您可能需要这么操作：

```
// ViewModels initialization

lateinit var userViewModel: UserViewModel

fun onCreate(bundle: Bundle?) {
    userViewModel = ViewModelProviders.of(this)
        .get(UserViewModel::class.java)
}
```

而现在只需要一行：

```
// ViewModels initialization

val userViewModel: UserViewModel by viewModels()
```

导航

导航 (Navigation) 是一套管理应用内 UI 流程的 Jetpack 代码库，现已发布了 2.1 的稳定版，与此同时下一个版本也已经出现在了不远的前方，接下来我们会：

- 在导航中提供成组 (Scoped) ViewModel，比如一套登录流程的界面集合就可以用一个 ViewModel 来管理
- 使用 URI 直接导航
- 对话框可以做为导航目标
- 更好地对动态功能做出支持

请大家保持对本次连载的关注，我们会在下篇中为大家介绍分页库、Room 持久性库和 WorkManager 的更新进展。您也可以观看 📺 下面的视频 📺 重温我们对架构组件进展的介绍。

- 腾讯视频链接

Android 架构组件的最新进展 (上篇)_
腾讯视频
v.qq.com



- Bilibili 视频链接

<https://www.bilibili.com/video/av71179658/>
www.bilibili.com



希望在了解完架构组件的最新进展后，大家能在其中找到适合自己应用的功能。如果对架构组件有任何疑问或者建议，欢迎在评论区和我们分享。



发布于 2019-10-14

[Android 开发](#)

[Android](#)

推荐阅读



Android Q 兼容那些事

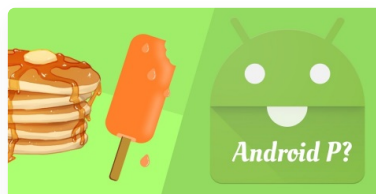
安卓小煜

Android绘制优化----系统显示原理

Android的显示过程可以概括为：Android应用程序把经过测量、布局、绘制后的surface缓存数据，通过SurfaceFlinger把数据渲染到屏幕上，通过Android的刷新机制来刷新数据。即应用层负责绘制，...

priva...

发表于Andro...



Google 可能会在 Android P 中更严格限制隐藏 API 的使用

Hevin



聊聊 Android 中的 @hide

Hevin

发表于极光日报

2 条评论

[切换为时间排序](#)

写下你的评论...



Kotlin

2019-10-14

Google大法好

👍 1



lasttears

2019-11-02

作为一个用户，什么时候能根治APP乱请求全写完，乱放文件就好了

👍 赞