关于学习方式

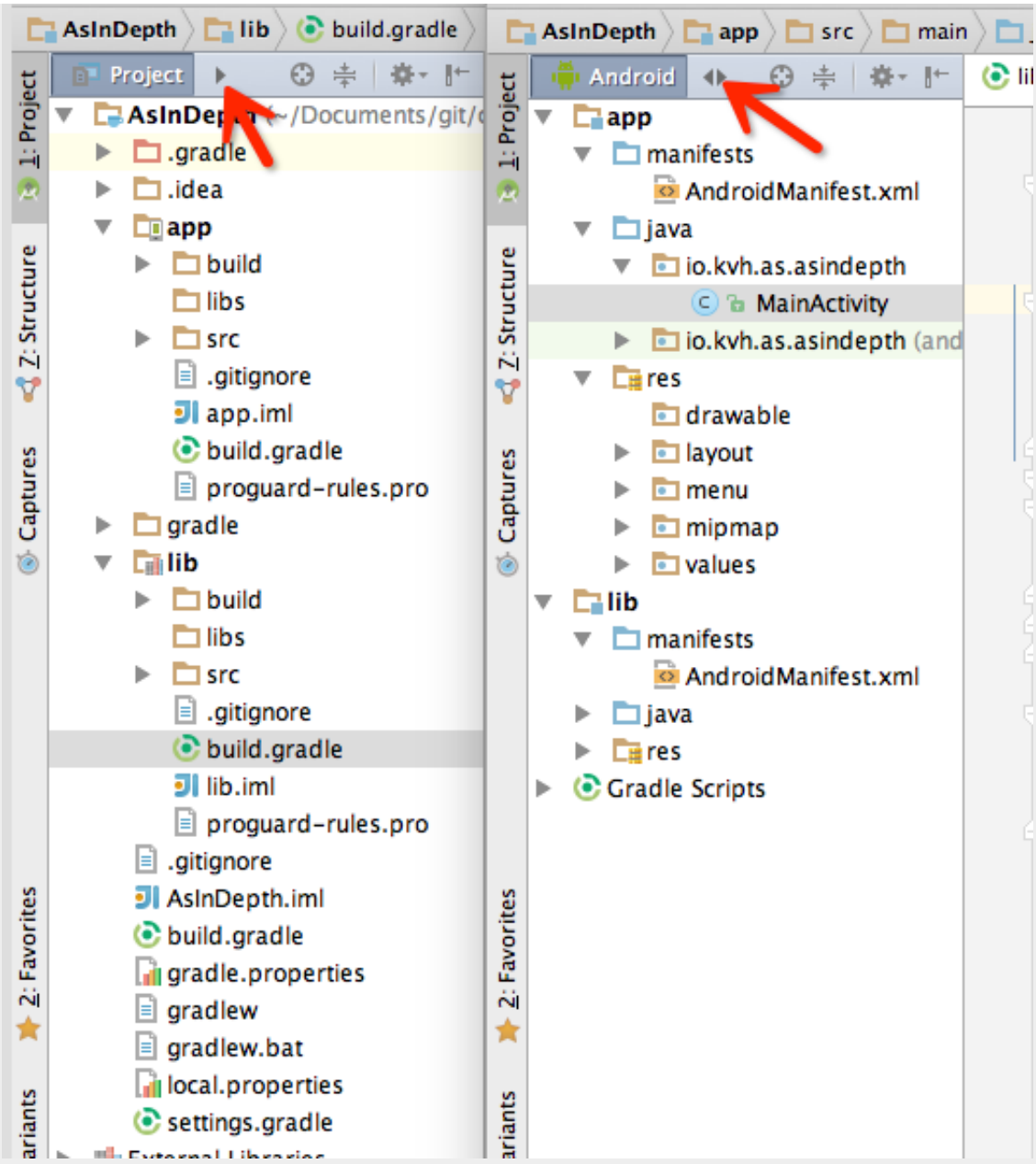
曾经跟朋友讨论过我们所接受过的大学工科教育，都是一上来先学基础理论，最后再来一个金工实习。一开始不知道为什么而学，学不进去，荒废了基础，等到金工实习的时候，又发现基础不牢，后悔不已。
考虑到传统教育方式的不足之处，笔者在组织本系列文章的时候是先讲入门实例，进而学习 Gradle 和 Groovy 基础原理，最后学习进阶实例。

上篇文章介绍了从 ADT 迁移到 Android Studio，相信经过很短时间的使用之后，已经开始熟悉和爱上 Android Studio 了。基础的功能我就不讲了，下面列举一些较为深入又比较实用的功能。

Android Studio 相关功能介绍

文件夹组织视图

最常用的有 Project 和 Android 视图，前者按照项目文件树进行组织，后者是以 Gradle 构建文件作为核心进行组织：



Project 视图与 Android 视图

Gradle 相关文件结构

让我们来观察一下Android Studio 中 Gradle 相关的结构：

```
.
├── gradle
│   └── wrapper                          //所使用的 Gradle 包装器配置
├── .gradle                                      //所使用 Gradle 版本
│   └── 2.8
├── AsInDepth.iml
├── app                                          //app module
│   ├── app.iml
│   ├── build
│   ├── build.gradle                  //app module 的 build.gradle
```

```
11  │   ├── libs
12  │   ├── proguard-rules.pro
13  │   └── src
14  ├── build.gradle                  //项目 build.gradle，通常配置项目全局配置，如 r
15  ├── gradle.properties             //项目属性文件，通常可以放置一些常量
16  ├── gradlew                                  //Gradle 包装器可执行文件
17  ├── gradlew.bat                        //Gradle 包装器可执行文件(Windows)
18  ├── lib                                      //lib module
19  │   ├── build
20  │   ├── build.gradle              //lib module 的 build.gradle
21  │   ├── lib.iml
22  │   ├── libs
23  │   ├── proguard-rules.pro
24  │   └── src
25  ├── local.properties              //项目的本地属性，通常是 sdk 所在位置
26  └── settings.gradle                     //项目总体设置，通常是配置项目中所有的 m
```

Invalidate Cache

Android Studio 会出现索引的问题，那可以从删除 cache 重建索引，File->Invalidate Caches/Restart

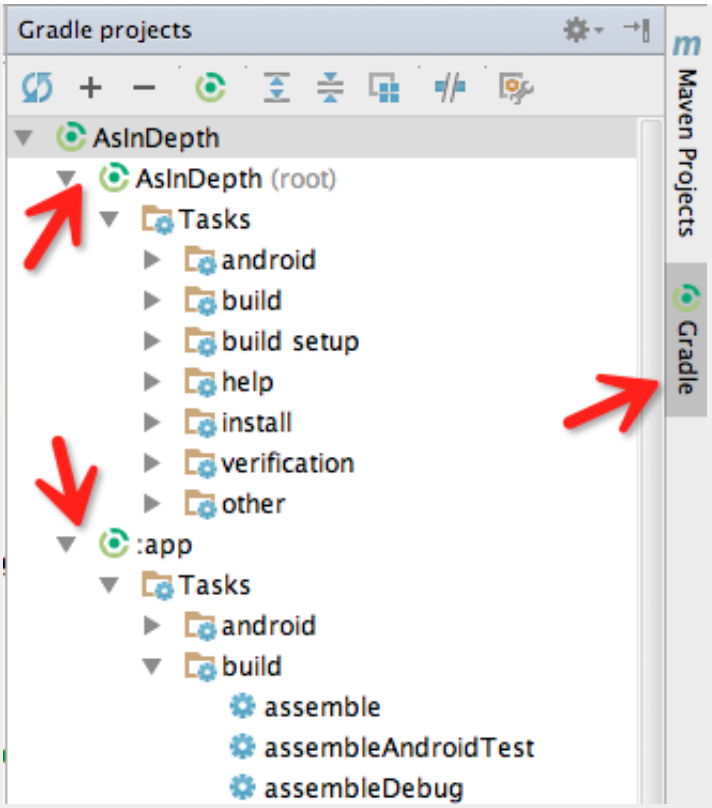
Multiple Language Editor

多语言文字可以通过右击文件 Open Translation Editor，可以同时进行编辑，但是我发现如果把 strings.xml 改了别的名字，这个功能就不 work 了。

Gradle 相关功能介绍

Gradle View

点击红色三角运行按钮，其实是执行了 Gradle 的 一些列任务，如果你想分别执行一些任务，则可以从 Gradle View 里面查看：



gradle view

命令行

工作区下方，有一个叫做 Terminal 的 tab，点击之后，会自动 cd 到当前 project 根目录下，可以输入如下命令来尝试下：

```
./gradlew build
```

> Windows 下应该是 gradlew.bat build，下面均以 Mac 为例，不再赘述

可使用 help 参数来查看有哪些选项：

```
./gradlew --help
```

下面介绍一些重要的选项：

- 查看运行 log

有些时候，一个任务运行失败，只给出一个错误，没有给具体原因，你就需要查看更多信息，可以使用参数 –info 或者 –stacktrace：

```
./gradlew build --info
```

- 指定 module 或者 build.gradle

Gradle 默认是当前目录下寻找 build.gradle 文件执行任务，这样执行 build 会使得整个 project 所有的 module 的 build 任务都会运行，浪费不必要的时间，可以指定 module (-p) 或者 build.gradle (-b)文件以缩小作用范围：

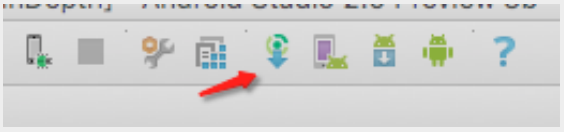
```
./gradlew -p app build
```

sync

正常情况下，修改了 build.gradle 文件，文件上方就会有一个 sync 的按钮，点击之后会重新构建整个 build.gradle。但是某些特殊情况，这个同步可能会失败。那就需要一个额外的触发。

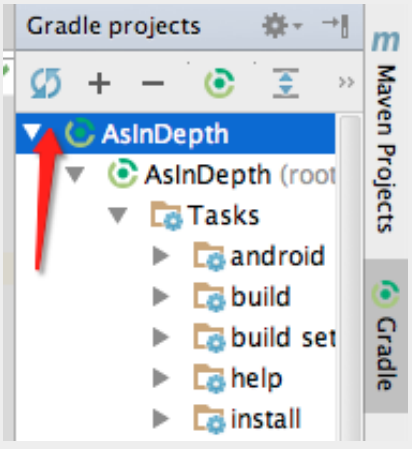
方法有四：

- 再修改一下文件，便会再次出现 sync 按钮
- 点击上方工作区的按钮



sync button top

- 点击 Gradle View 中的同步按钮



sync button top

- 命令行执行一次 build

Build Variant

首先要了解两个概念:

Build Type

分为 debug 和 release，这个概念容易懂

Product Flavor

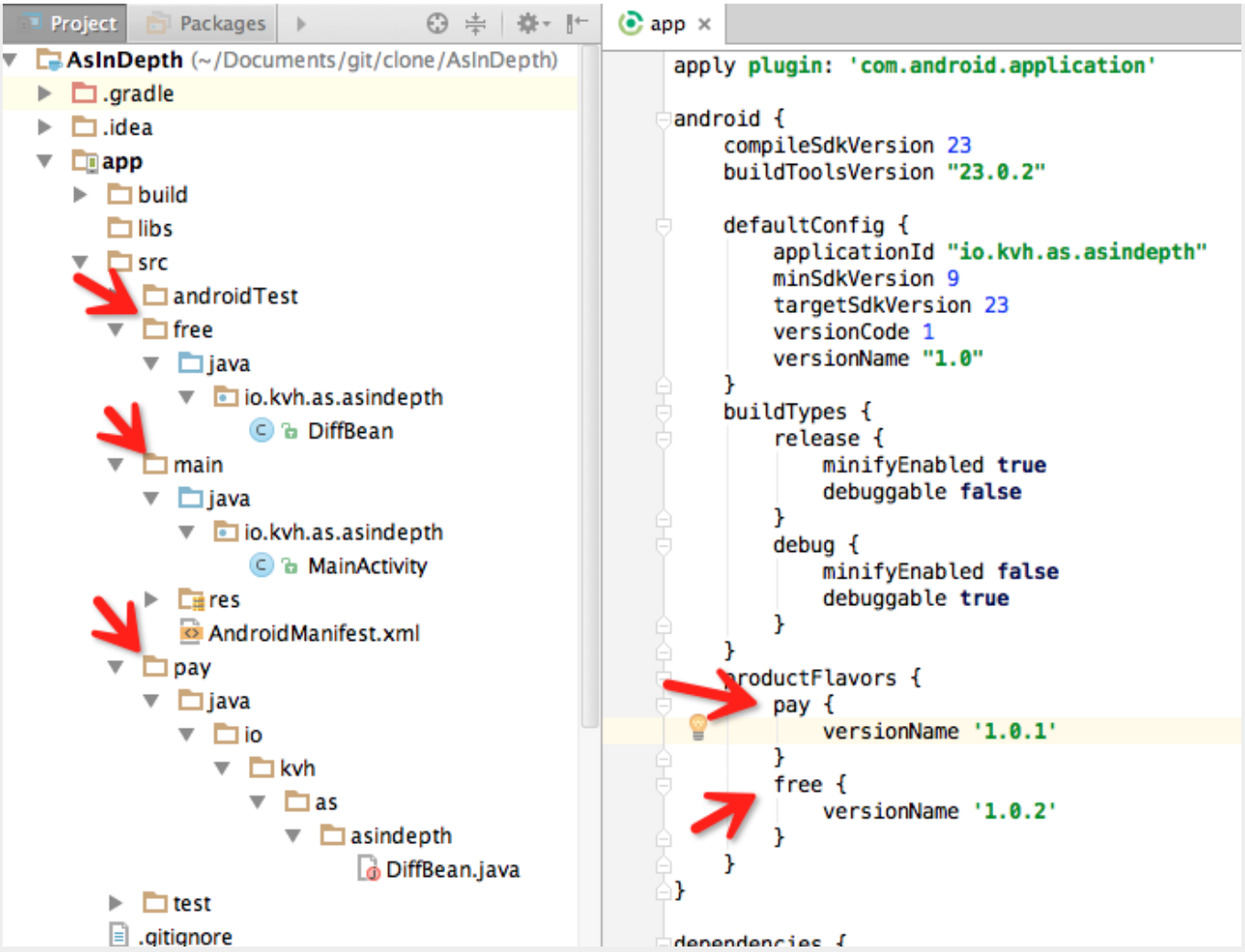
这个概念主要是为了满足如下需求：同一份代码要打多个包，例如收费 pay 和免费 free，逻辑上有一些小区别，又不想通过逻辑判断这种丑陋的方式。或者你要实现所谓多渠道打包。

Build Variant = Build Type x Product Flavor

配置好了Build Type 和 Product Flavor 之后，Gradle 会生成若干个包，分别为：

```
payDebug
payRelease
freeDebug
freeRelease
```

配置实例



build variant

假设这两个版本的 app，有一个类 DiffBean 需要做大量的逻辑判断，则可以通过在 build.gradle 中配置 product flavor，在代码中添加两个与 main 平齐的文件夹，
把 DiffBean 从 main 中抽出来，分别放在两个文件夹中，只关注对应的逻辑即可。

关于 Product Flavor 中都能定义哪些属性，请参考 Android Gradle DSL。

Gradle 使用的仓库

要使用远程依赖，就得有个库的仓库，Gradle 支持 maven 仓库。这些库可以是公用的，例如 mavenCentral 或者 jcenter，也可以使用私有库。

笔者曾经上传过公用库到 mavenCentral 和 jcenter，前者的繁琐与难以管理，让人深恶痛绝。
Android Studio 在0.8.0版本以后，将 jcenter 作为默认的 maven 库公用库来源。

私有库可以托管在任何一个能访问到的地方，可以托管在 bintray 上的私有空间，也可以是内网服务器上，甚至可以是本机磁盘上。

下面是本人使用的仓库的一个例子：

```
allprojects {
    repositories {
        jcenter()
        mavenCentral()
        maven {
            url 'file:///Users/myusername/repo/'
        }
    }
}
```

关于库，会在本系列的四篇会有更详细的叙述。

遇到的坑

笔者所在的项目的包已经发布到了 mavenCentral 和 jcenter，很多用户的在集成的时候，遇到问题，其中最经典的问题，便是下面的三个。

下载依赖库失败，报 **peer not found**

连接 jcenter 库默认使用 https 协议，出现这个错，多数情况下都是因为连接失败了，原因嘛，你懂的。
可以尝试将 jcenter 改成默认使用 http 连接：

```
jcenter {
            url "http://jcenter.bintray.com/"
        }
```

下载不到特定的版本

Maven 和 Gradle 都会有一个缓存库，默认安装的情况下，是在用户根目录下的 .m2 或者 .gradle 文件夹中。
可以尝试删除里面的文件。例如将 Gradle 的 cache 文件删除：

```
rm -rf ~/.gradle/caches/
```

> 注意这个操作可能会导致所有的远程库都需要重新下载，要三思后行。

库重复冲突

笔者的包使用了 com.android.support:support-v4:19.0.0 包，可能用户的 app 也使用了这个包，就可能在编译进行代码合并到时候出类似的错：

```
UNEXPECTED TOP-LEVEL EXCEPTION
multiple dex files define Landroid/support/annotation/AniRes
```

这个时候，在 module 的 build.gradle 的 dependency 添加依赖时，加入 exclude 规则：

```
compile ('com.bugtags.library:bugtags-lib:latest.integration') {
        exclude group: 'com.android.support', module: 'support-v4'
}
```

后续引言

讲到这里，有些读者可能会遇到跟我当时开始使用 Gradle 一样的问题：遇到问题就 stackoverflow，找到 workaround 了但不知道为什么，gradle 版本更新了，发现不 work 了，甚为惆怅。

究其原因，都是不理解基础。

首先 Gradle 是一个构建平台，它使用的是 Groovy 语言。

Groovy是一种基于 Java 的语言，提供了更好的动态特性，可以使用闭包使得编程更灵活，很适合做脚本语言。

上面提到的 settings.gradle 和 build.gradle 在 Gradle 平台中，其实都是一个 Groovy 对象。

Gradle 通过插件（plugin）的方式来支持构建。插件是很多任务（task）的集合，task 中又包含了许多 action。

而例如 productFlavors 都是一个所谓的 DSL，插件都定义了很多的 DSL，我理解的所谓的 DSL 就是让脚本看起来像脚本。Android 的插件的 DSL 文档在 Android Gradle DSL 有说明。

理解了以上基础之后，你就会知其然，知其所以然了。

以上知识，下一篇将会详细介绍。敬请留意。

> 有问题？在文章下留言或者加 qq 群：453503476，希望能帮到你。
>
> 本文所使用的 demo 已经上传到了 github 中，可以参阅 embrace-android-studio-demo

参考文献

Android Studio Overview

Android Studio Installation

Android Gradle DSL

Android New Build System

系列导读

本文是笔者《拥抱 Android Studio》系列第二篇，其他篇请点击：

拥抱 Android Studio 之一：从 ADT 到 Android Studio

拥抱 Android Studio 之二：Android Studio 与 Gradle 深入

拥抱 Android Studio 之三：溯源，Groovy 与 Gradle 基础

拥抱 Android Studio 之四：Maven 库，本地库与发布到 bintray 或者 mavenCentral

拥抱 Android Studio 之五：Gradle 插件使用与开发