史上最详细的 Android Studio 系列教程一--下载和安装

背景

相信大家对 Android Studio 已经不陌生了,Android Studio 是 Google 于 2013 I/O 大会针对 Android 开发推出的新的开发工具,目前很多开源项目都已经在采用,Google 的更新速度也很快,明显能感觉到这是 Android 开发的未来,那么我们还有什么理由不去拥抱未来呢?

虽然推出了很久,但是国内貌似普及的程度并不高,鉴于很多朋友求 studio 的详细教程,那么今天我就手把手教大家下载、安装、使用,Studio 之路从这里开始。

Android Studio VS Eclipse

相信目前国内用 Eclipse 的还是大多数,那么首先就来说一下 Studio 的一些优点,比较才能更有说服力,才能说明为什么我们要从 Eclipse 迁移到 Studio。

•1、Google 推出的

毫无疑问,这个是它的最大优势,Android Stuido 是 Google 推出,专门为 Android"量身订做"的,是 Google 大力支持的一款基于 Intellij IDEA 改造的 IDE,这个应该能说明为什么它是 Android 的未来

●2、速度更快

Eclipse 的启动速度、响应速度、内存占用一直被诟病,相信大家这点应该深有体会,而且经常遇到卡死状态。Studio 不管哪一个方面都全面领先 Eclipse

●3、UI 更漂亮

I/O 上演示的那款黑色主题真是太棒了,极客范,Stuido 自带的 Darcula 主题的炫酷黑界面实在是高大上,相比而言 Eclipse 下的黑色主题太 low 了

●4、更加智能

提示补全对于开发来说意义重大, Studio 则更加智能,智能保存,从此再也不用每次都 Ctrl + S 了。熟悉 Studio 以后效率会大大提升。

•5、整合了 Gradle 构建工具

Gradle 是一个新的构建工具,自 Studio 亮相之处就支持 Gradle ,可以说 Gradle 集合了 Ant 和 Maven 的 优点,不管是配置、编译、打包都非常棒。

6、强大的 UI 编辑器

Android Studio 的编辑器非常的智能,除了吸收 Eclipse+ADT 的优点之外,还自带了多设备的实时预览,相对这对 Android 开发者来说简直是神器啊。

●7、内置终端

Studio 内置终端,这对于习惯命令行操作的人来说简直是福音啊,再也不用来回切换了,一个 Studio 全部搞定。

•8、更完善的插件系统

Studio 下支持各种插件,如 Git、Markdown、Gradle 等等,你想要什么插件,直接搜索下载。

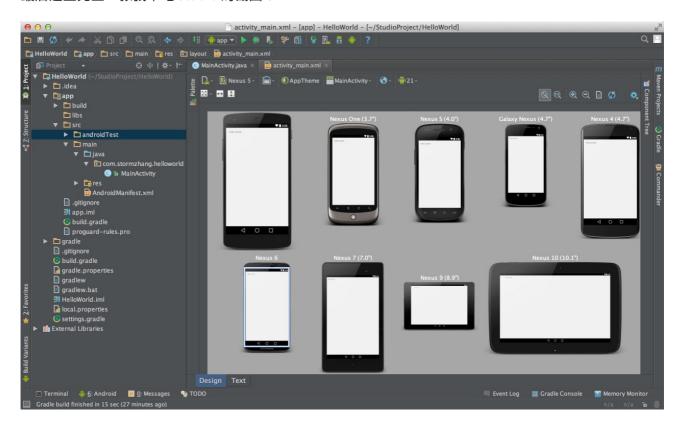
•9、完美整合版本控制系统

安装的时候就自带了如 GitHub, Git, SVN 等流行的版本控制系统,可以直接 check out 你的项目。

大家看完以上是不是很动心呢,优点是很多,但是大家学习的时候会遇到很多问题,如 Studio 和 Eclipse 的目录结构、快捷键等等完全不一样,需要适应一段时间,Gradle 同样增加了学习成本,虽然 Google 的更新速度已经相当快了,但是目前最新的是 1.0RC 版本,仍未推出正式版,说明可能会有一些小问题等, Studio 官方解释暂未支持 NDK,所以如果你的项目用到了 NDK 最好也不要使用 Studio。

但是相信 Google 会越来越完善的,学习成本与适应阶段是我们做技术一直要保持的心态,一旦上手相信你要离不开它了。

最后这里先上一张我本地 Studio 的截图:



准备

因为 Google Android 的一些官方网站在国内访问有限制,原因你懂得。所以在开始下载安装 Studio 之前,你需要自备梯子,关于如何翻墙有很多种方法,这里就不做过多介绍,私以为作为一个 Android 开发者,不懂翻墙基本没法做下去。所以这点投入是值得的,这里推荐大家直接购买 VPN 吧,因为我曾经折腾了很多翻墙的玩意,要么不稳定,要么速度慢,后来想通了,凡是花点钱能解决的问题都不是问题,这里推荐云梯 VPN,价格算是很便宜的了,别再问我速度、稳定性如何,我已经使用并续费快两年了。(通过这个链接购买的,你的账户可以优惠 10 元)

下载

官方下载有两个地方,均需要翻墙。

●Android Developer 官网

Android 开发者官网的网站,可直接下载,但是这个网站貌似只更新 Beta 和正式版,目前只更新到 Beta 0.8.14 版本。

•Android Tools Project Site

Android 开发工具的网站,上面链接是 Studio 的 canary 渠道,列出了 Studio 各种实时预览版等,目前最新的是 1.0RC 版本。

创建 HelloWorld 项目

至于安装就没什么可说的了,点击直接运行了。这里我以 Mac 系统的 1.0RC 版本为例,来创建第一个 HelloWorld 项目。其他平台基本上差不多,在这之前假设你已经配置好了 JDK 和 Android SDK 环境,并且你是第一次安装 Studio。

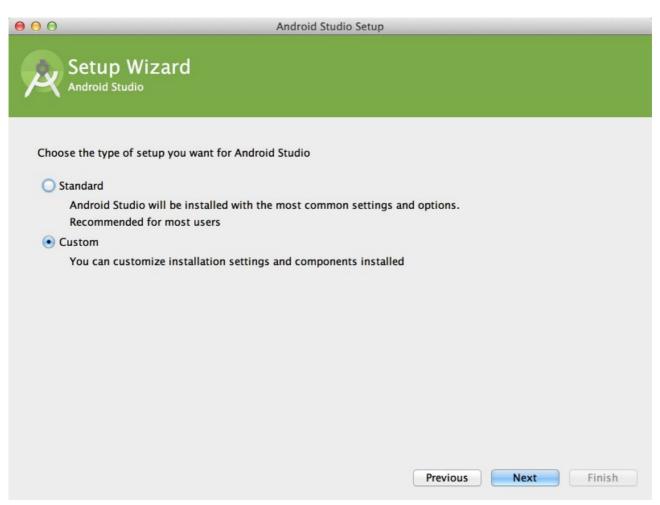
•1、首先运行时的欢迎画面



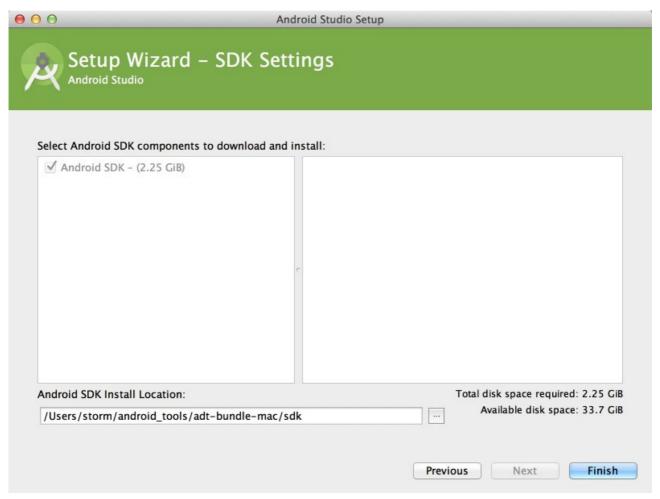
2、之后第一次的话 会进入到设置向导页



点击 Next 进入选择设置类型向导页



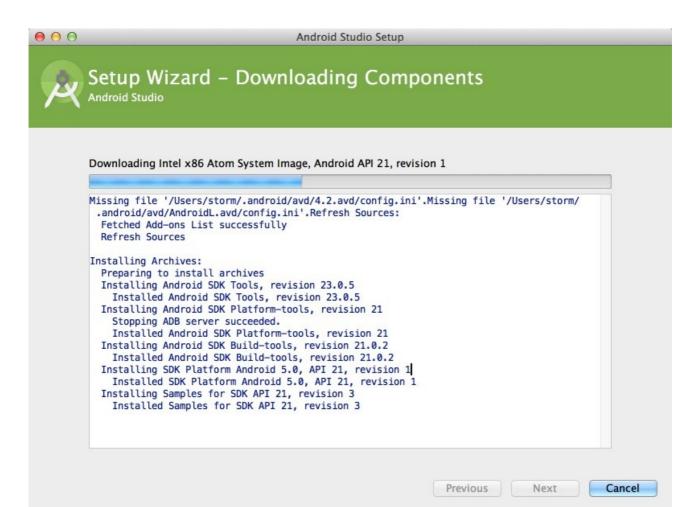
因为我本地已经下载 SDK 并配置好了环境变量,所以我选择"Custom",然后到下一步:



这一步选择你本地 SDK 的位置,可以看到有个 2.25GB 的 SDK 要下载,那是因为 Studio 1.0 默认要下载 5.0 的 SDK 以及一些 Tools 之类的,然后点击"Finish"按钮

(PS: 这个选择并下载 2.25G 的组件是 studio 的一个 bug , 评论里有人提醒 , 感谢这位同学。如果网速不行想跳过这步的可以在 bin 目录的 idea.properties 增加一行: disable.android.first.run=true 就行了 , mac 平台的右键安装包->Show Package Contents 就找到 bin 目录了。)

•3、下载依赖组件



之后便到下载组件页面,这个过程需要翻墙,而且依赖你的网速,时间有点久,大家耐心等待...





Setup Wizard - Downloading Components Android Studio

Missing file '/Users/storm/.android/avd/4.2.avd/config.ini'.Missing file '/Users/storm/ .android/avd/AndroidL.avd/config.ini'.Refresh Sources: Fetched Add-ons List successfully Refresh Sources Installing Archives: Preparing to install archives Installing Android SDK Tools, revision 23.0.5 Installed Android SDK Tools, revision 23.0.5 Installing Android SDK Platform-tools, revision 21 Stopping ADB server succeeded. Installed Android SDK Platform-tools, revision 21 Installing Android SDK Build-tools, revision 21.0.2 Installed Android SDK Build-tools, revision 21.0.2 Installing SDK Platform Android 5.0, API 21, revision 1 Installed SDK Platform Android 5.0, API 21, revision 1 Installing Samples for SDK API 21, revision 3 Installed Samples for SDK API 21, revision 3 Installing Intel x86 Atom System Image, Android API 21, revision 1 Installed Intel x86 Atom System Image, Android API 21, revision 1 Installing Sources for Android SDK, API 21, revision 1 Installed Sources for Android SDK, API 21, revision 1 Installing Android Support Repository, revision 9 Installed Android Support Repository, revision 9 Installing Google Repository, revision 13 Installed Google Repository, revision 13 Previous Next

下载完成后点击"Finish"按钮

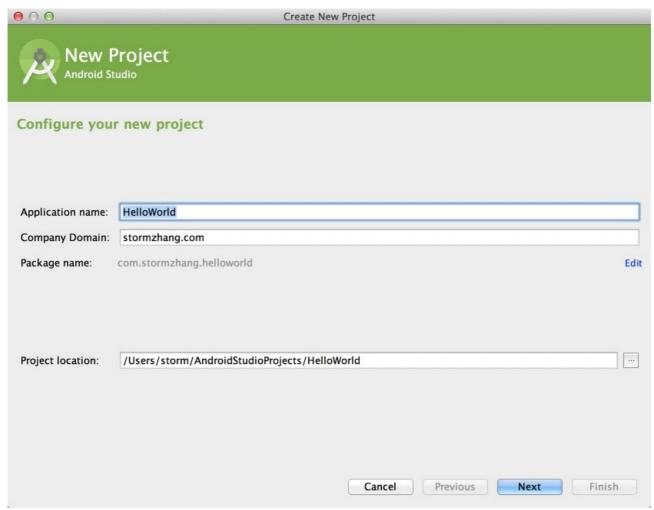
•4、新建项目



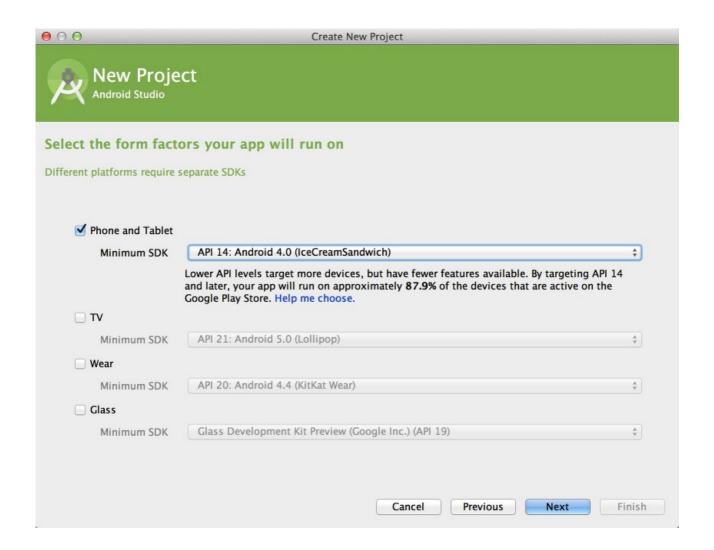
Android Studio 1.0 RC 1 Build 135.1598475. Check for updates now.

在这个页面我们可以新建项目,也可以导入项目本地或者 GitHub 上的项目等,左边可以查看最近打开的项目等,这里我直接新建项目

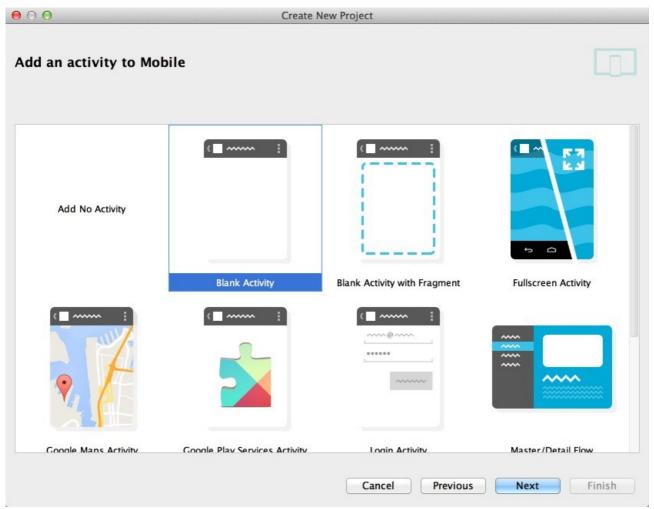
然后到如下界面



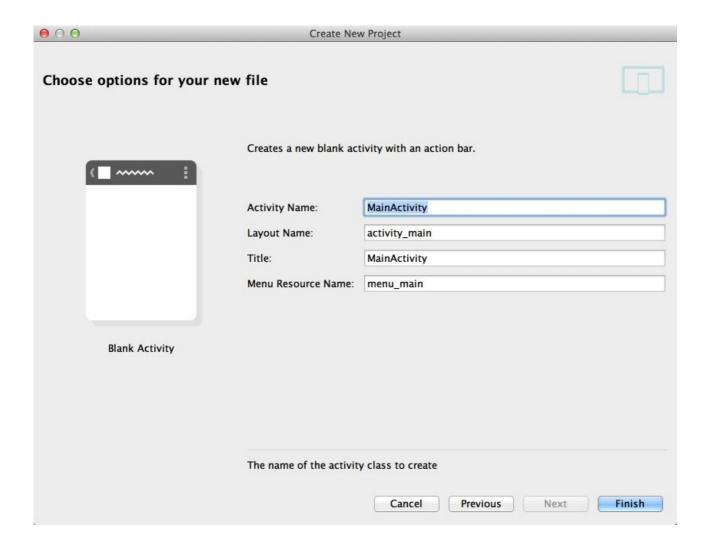
我们填上项目名称和报名以及项目路径等然后"Next"

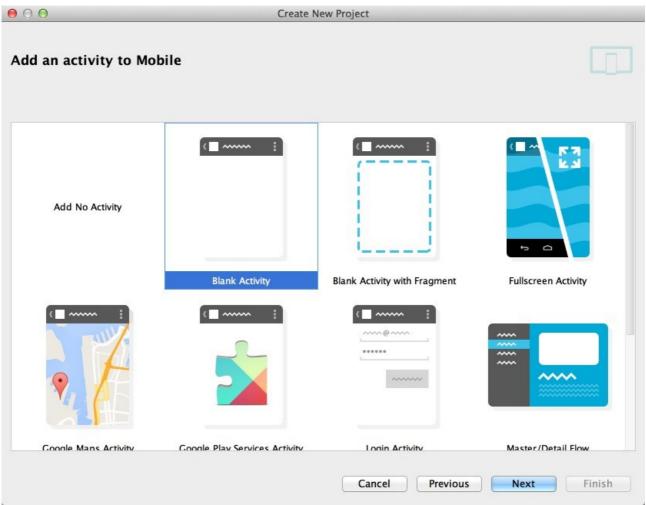


这个页面支持你适配 TV、Wear、Glass 等,我们只选择第一项就 ok,选好最小 SDK 然后"Next"

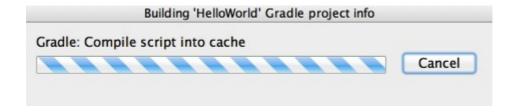


这个页面选择一个 Activity 模板,和 Eclipse 很像,我们直接选择一个 Blank Activity 好了

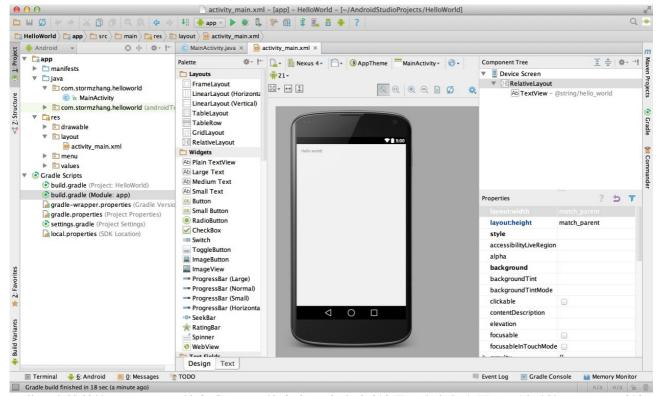




点击"Finish"后等一会出来如下一个进度条,很多人容易卡在这里,这里需要下载 Gradle,只第一次会下载,有点慢,需要翻墙,大家也耐心等待下

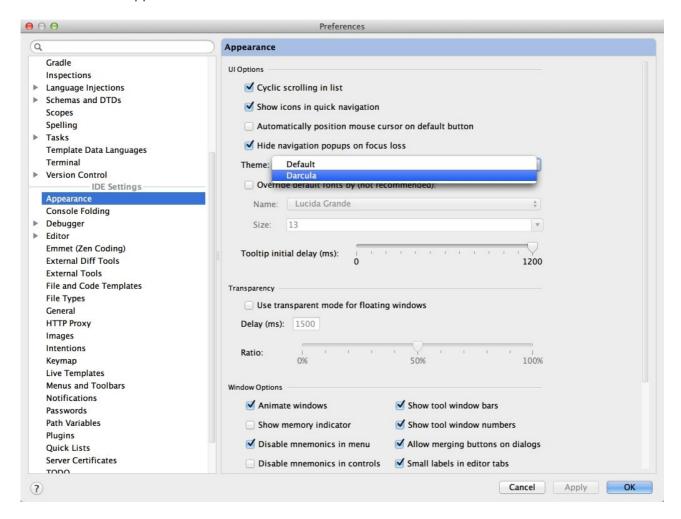


下载成功后变看到如下完整的项目界面

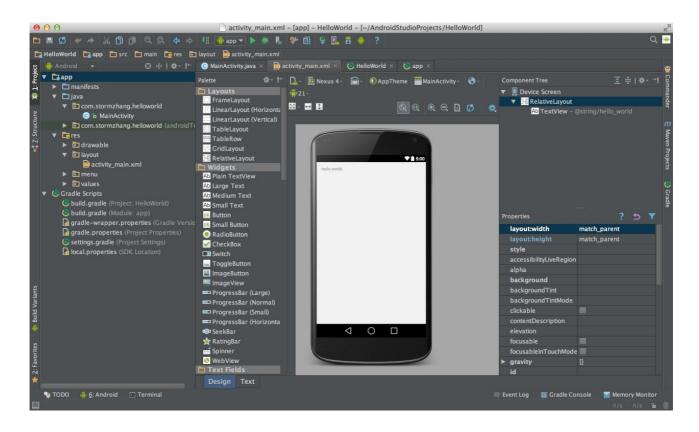


至此一个简单的 Studio 项目就完成了,图片中也可以看到默认是一个白色主题,不够酷炫? Studio 默认自带一款高大上的黑色主题,只需要简单修改下就 OK。

到 Preference -> Appearance 下更改主题到 Darcula



之后我们再来看一下更改后的主题



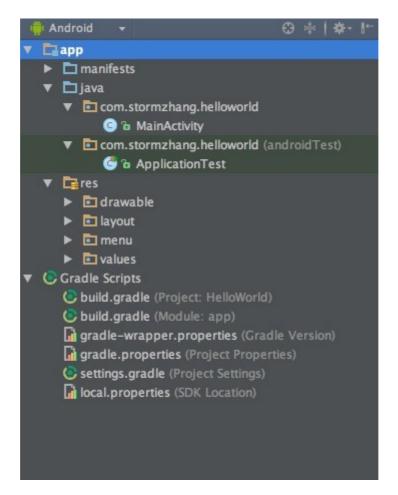
很酷炫有木有?至此第一期的 Studio 简单介绍就到这里,后面会介绍目录结构、快捷键、导入项目、集成 GitHub、Gradle 等等,敬请期待。

史上最详细的 Android Studio 系列教程二--基本设置与运行

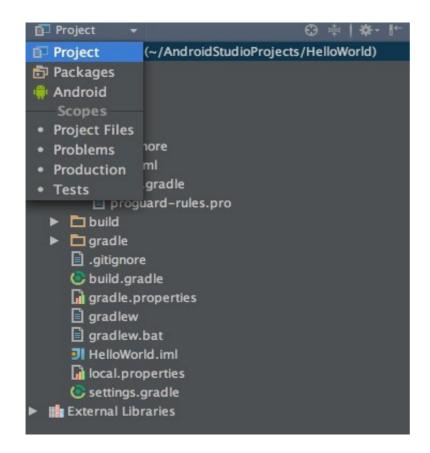
上面一篇博客,介绍了 Studio 的优点与 1.0 RC 的安装与上手体验,没想到 google 的更新速度这么快,已经出了 RC 2 版本,主要是修复一些 bug。那么今天就带大家预览下 Stduio 的界面与基本功能。

项目结构

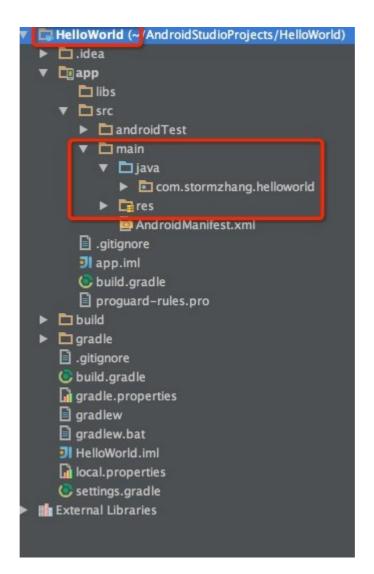
当我们新建一个项目的目录结构默认是这样的



可以看到和 Eclipse 的目录结构有很大区别,Studio 一个窗口只能有一个项目,而 Eclipse 则可以同时存在很多项目,如果你看着不习惯可以点击左上角进行切换



切换到"project"模式下的目录结构是这样的,我个人也更习惯这种格式

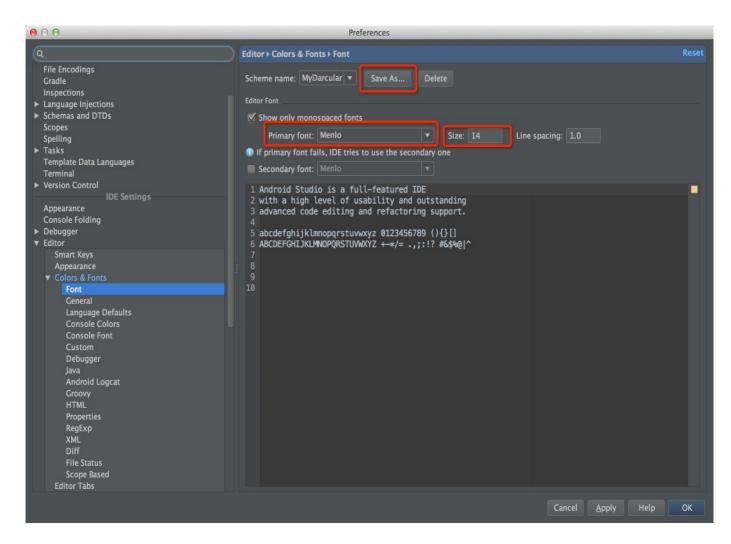


和 Eclipse 的区别有如下:

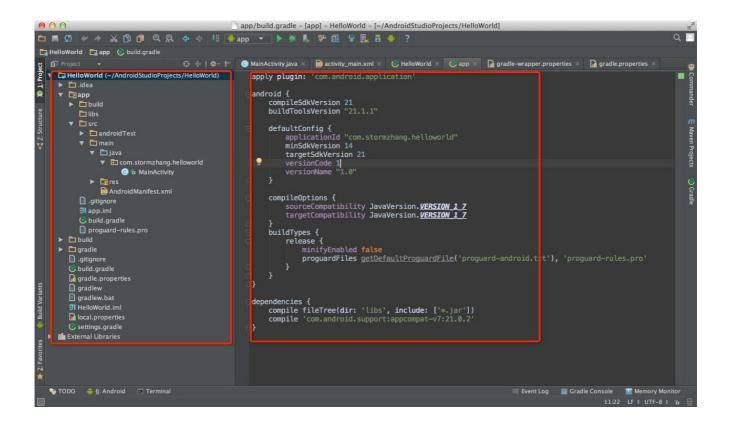
- ●1、Studio 中有 Project 和 Module 的概念,前面说到 Studio 中一个窗口只能有一个项目,即 Project, 代表一个 workspace,但是一个 Project 可以包含多个 Module,比如你项目引用的 Android Library, Java Library 等,这些都可以看做是一个 Module;
- •2、上述目录中将 java 代码和资源文件(图片、布局文件等)全部归结为 src,在 src 目录下有一个 main 的分组,同时划分出 java 和 res 两个文件夹,java 文件夹则相当于 Eclipse 下的 src 文件夹,res 目录结构则一样.

偏好设置

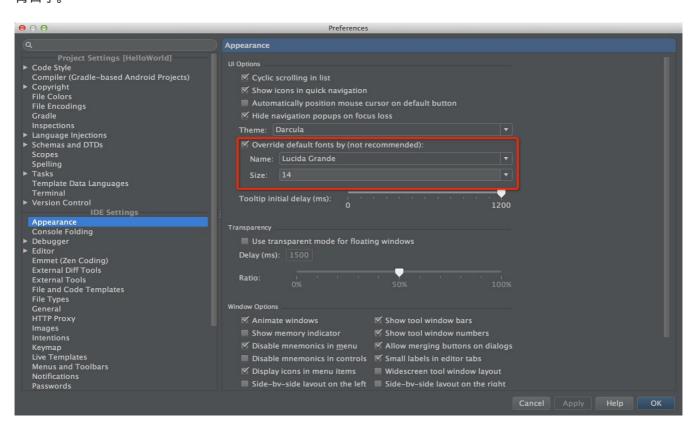
进入后你也许发现字体大小或者样式不符合你的习惯,比如我是觉得代码太小看起来伤眼,Darcular 主题默认的字体是 12,我个人更习惯 14 的字体大小。没关系,到 Preferences (设置)页面搜索 Font 找到 Colors&Fonts 下的 Font 选项,我们可以看到默认字体大小是 12,但是无法修改,需要先保存才可以修改,点击 Save as 输入一个名字,比如 MyDarcular,然后就可以修改字体大小和字体样式了.



点击确定之后再回到页面发现字体是变大了,但是 Studio 默认的一些字体大小如侧边栏等确没有变化,看起来很不协调,如下图



强迫症的你肯定无法忍受,没关系,这里也同样可以设置,到 Preferences -> Appearance 修改如图所示就 ok,这里同样不仅可以更改字体大小,也可以选择不同的字体,点击 OK,这次页面字体就完全对你胃口了。



调整之后再看下效果

```
app/build.gradle - [app] - HelloWorld - [~/AndroidStudioProjects/HelloWorld]
                                                                                                                                                                                             Q 🖪
 <u>□ ⊞ ∅ ؇ ở ኤ ὑ 햅 ◯ Ջ ♦ ⇨ Ⅵ ᡨapp▼ ▶ ※ № № ♀ ጪ ೪ № 苗 ቀ ?</u>
 HelloWorld app & build.gradle
D Project

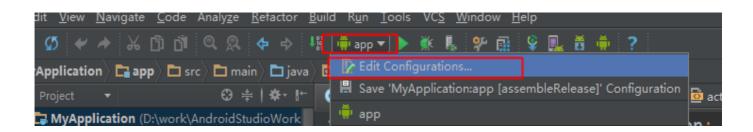
▼ □ Hello
                                           ⊕ 🖶 | 🌣 - 🎼 📵 MainActivity.java × 💆 activity_main.xml × 🚺 app ×
    ▼ 📮 HelloWorld (~/AndroidStudioProjects/HelloWo
                                                                                                                                                                                              ■ Con
                                                               apply plugin: 'com.android.application'
      ▶ 🗖 .idea
1:
      ▼ 🗓 app
▶ 🗀 build
                                                                     compileSdkVersion 21
buildToolsVersion "21.1.1"
                                                                    defaultConfig {
            ▶ □ androidTest
                                                                         applicationId "com.stormzhang.helloworld"
minSdkVersion 14
             ▼ 🖿 main
                                                                          targetSdkVersion 21
                       © <sup>™</sup> MainActivity
                  res
AndroidManifest.xml
                                                                 compileOptions {
    sourceCompatibility JavaVersion. VERSION 1 7
    targetCompatibility JavaVersion. VERSION 1 7
            app.iml
build.gradle
             proguard-rules.pro
                                                                    buildTypes {
    release {
         gradle
gitignore
                                                                              uninifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
         build.gradle
         gradle.properties
         gradlew
         gradlew.bat
                                                                dependencies {
                                                                    compile fileTree(dir: 'libs', include: ['*.jar'])
compile 'com.android.support:appcompat-v7:21.0.2'
         local.properties
      settings.gradle
     ▶ <u>4</u>: Run 🔏 TODO 🌞 <u>6</u>: Android 🗈 Terminal 🔲 <u>0</u>: Messages
                                                                                                                                           🔍 Event Log 🛘 🔟 Gradle Console 🔛 Memory Monitor
```

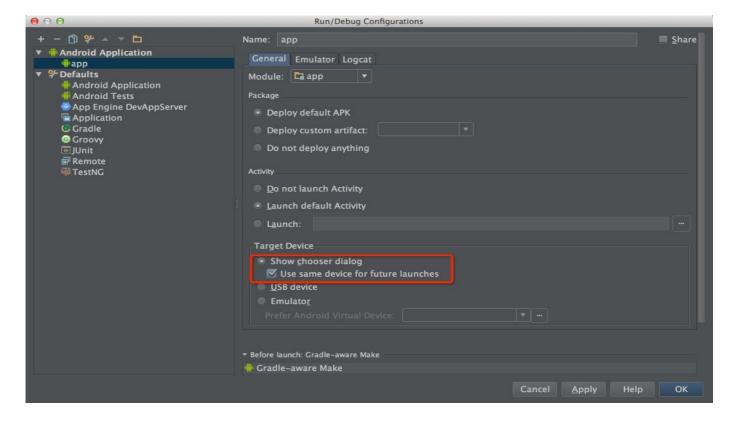
运行

接下来运行程序,运行和 Eclipse 中比较像,点击菜单栏的绿色箭头直接运行



Studio 默认安装会启动模拟器,如果想让安装到真机上可以配置一下。在下拉菜单中选择 Edit Configurations 选择提示或者是 USB 设备。





常用功能

在 Studio 菜单栏的右边有这样几个常用的功能,如图分别是 Gradle 同步、AVD Manager、SDK Manager、DDMS



Gradle 同步 在你项目运行或者更改 Gradle 配置的时候都要点击下这个按钮,会下载相应的依赖

AVD Manager 模拟器管理

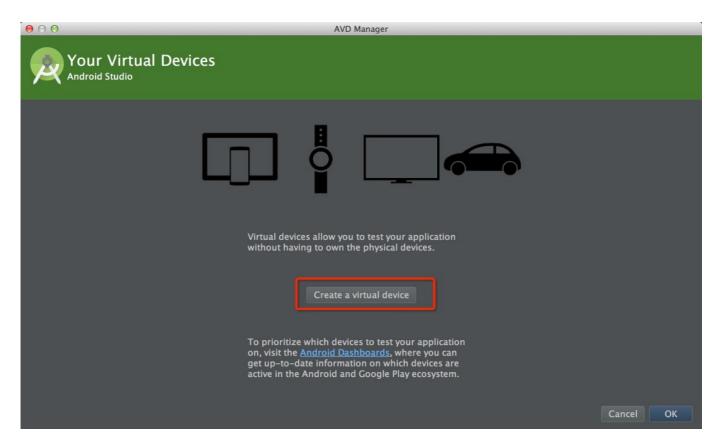
SDK Manager 就是管理你的 SDK 版本

DDMS 即 Dalvik Debug Monitor Service, Dalvik 调试监控服务。

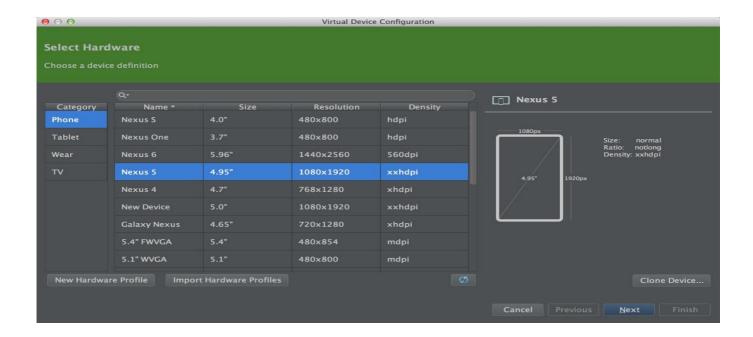
创建模拟器

建议在创建模拟器前把 SDK Manager 中的 Tools、Extras 都更新到最新。

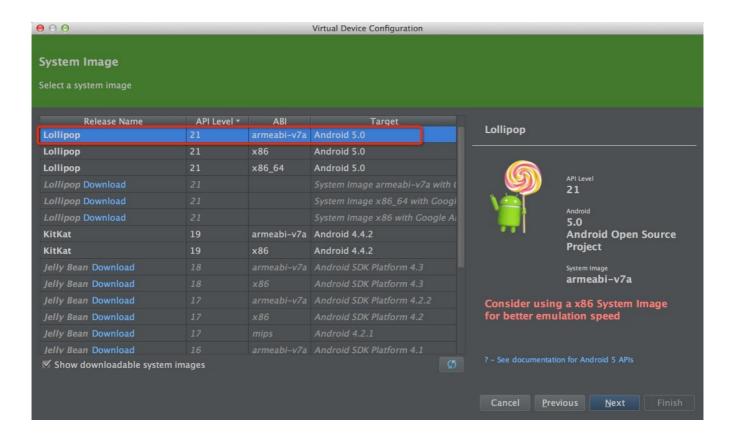
点击 AVD Manager 按钮



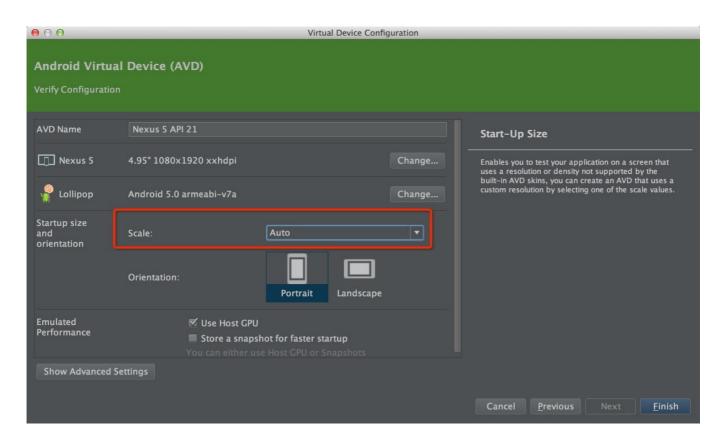
点击图中的创建按钮



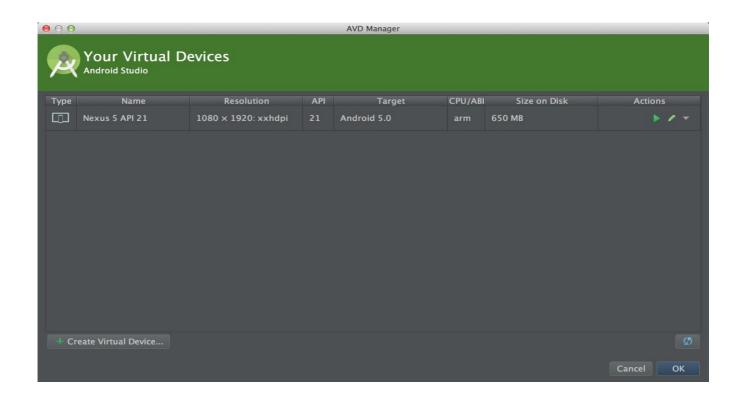
选择一个设备,这里我选择 Nexus 5,然后 Next



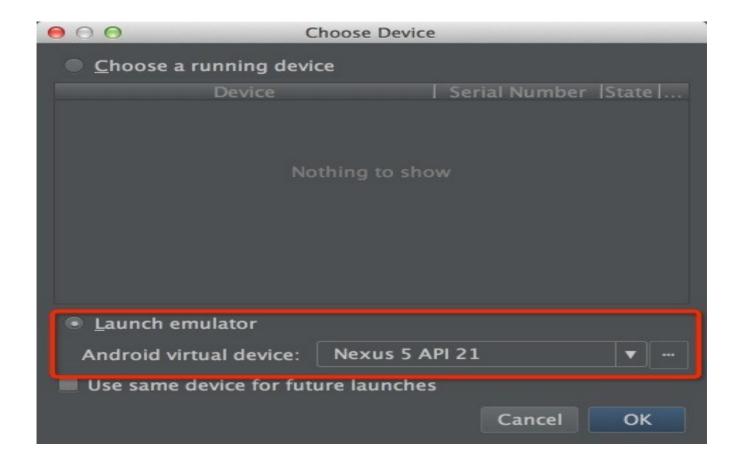
这里选择一个系统版本,这里以5.0为例,然后Next



由于各位的屏幕尺寸不一样,建议这里 Scale 一栏选择 Auto, 然后点击 Finish 接着可以看到我们已经创建好一个 5.0 的模拟器了



这次我们再运行,选择模拟器启动看下最终效果(模拟器的启动很慢,大家耐心等待)

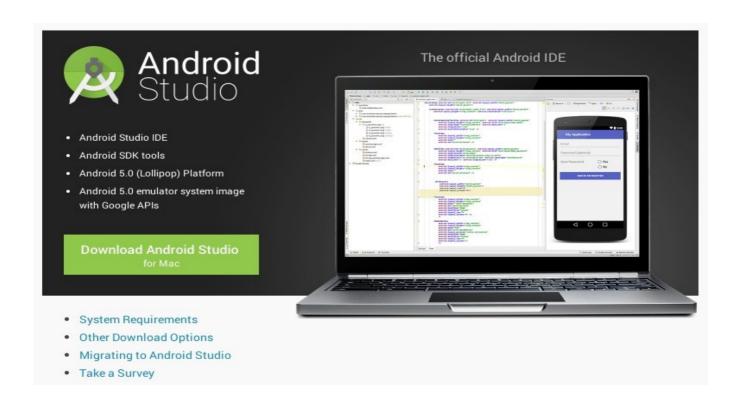




史上最详细的 Android Studio 系列教程三--快捷键

Android Studio 1.0 正式版发布啦

今天是个大日子,Android Studio 1.0 终于发布了正式版 ,这对于 Android 开发者来说简直是喜大普奔的大消息啊,那么就果断来下载使用。



官方下载地址: http://developer.android.com/sdk/index.html

如果你之前已经使用其他版本的 Studio,那么直接覆盖就好了,如果是第一次使用,那么参照 Android Studio 系列教程一进行安装配置。

于此同时一起更新的还有 SDK Tools 等,打开 SDK Manager 进行更新就 OK。如果之前有在使用 RC 版本的 Studio,更新覆盖之后应该会出错,那是因为 Stuido 1.0 默认使用 1.0 的 Gradle 插件,所以只需到项目根目录的 **build.gradle** 更新成如下代码然后同步下就 OK 了

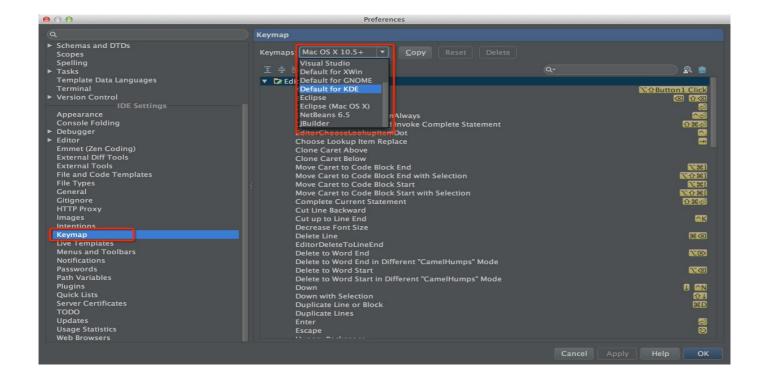
```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        Classpath 'com.android.tools.build:gradle:1.0.0'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
allprojects {
    repositories {
        jcenter()
    }
}
```

快捷键

对于很多 Eclipse 转过来的同学,不适应的最主要就是快捷键部分了,Studio 默认的快捷键和 Eclipse 差别很大,但是 Studio 强大的地方在于通过设置你可以一直沿用 Eclipse 风格的快捷键,这样对于那些 Eclipse 过渡来的同学省了很多学习成本。

Preferences -> Keymap 然后就可以选择你想要的快捷键,这里不止可以选择 Eclipse, 还可以选择 Emacs、Net Beans 等编辑器的快捷键



对于新手们或者对 Eclipse 没有好感的同学们,肯定想学习默认的 Studio 快捷键,下面整理了下常用的一些快捷键,更多的快捷键还需要大家自己去习惯适应。(我自己选择的是 Mac OS X 10.5+)

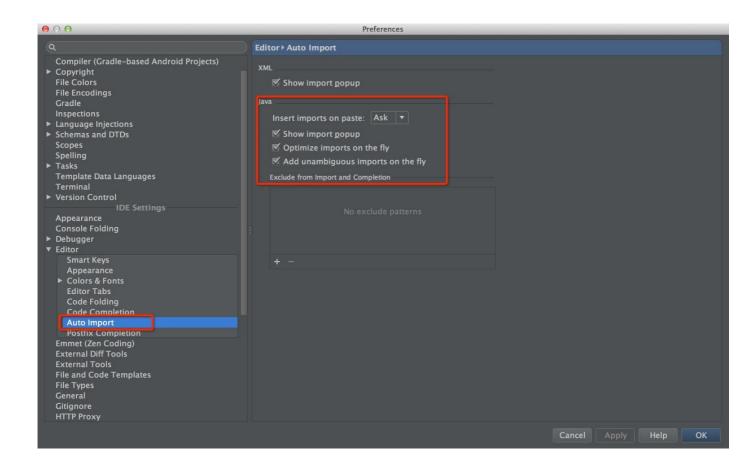
Action	Mac OSX	Win/Linux
注释代码(//)	Cmd + /	Ctrl + /
注释代码(/**/)	Cmd + Option + /	Ctrl + Alt + /
格式化代码	Cmd + Option + L	Ctrl + Alt + L
清除无效包引用	Option + Control + O	Alt + Ctrl + O
查找	Cmd + F	Ctrl + F
查找+替换	Cmd + R	Ctrl + R
上下移动代码	Option + Shift + Up/Down	Alt + Shift + Up/Down
删除行	Cmd + Delete	Ctrl + Y
扩大缩小选中范围	Option + Up/Down	Ctrl + W/Ctrl + Shift + W
快捷生成结构体	Cmd + Option + T	Ctrl + Alt + T
快捷覆写方法	Ctrl + O	Ctrl + O
快捷定位到行首/尾	Cmd + Left/Right	Ctrl + Left/Right
折叠展开代码块	Cmd + Plus,Minus	Ctrl + Plus/Minus
折叠展开全部代码 块	Cmd + Shift + Plus,Minus	Ctrl + Shift + Plus,Minus
文件方法结构	Cmd + F12	Ctrl + F12
查找调用的位置	Ctrl + Option + H	Ctrl + Alt + H
大小写转换	Cmd + Shift + U	Ctrl + Shift + U

(PS: 以上快捷键没有在 win 平台尝试确认,欢迎反馈错误与补充)

自动导包

最后再介绍一个最有用的设置,我们只有每次引用一些类的时候必须要导包,而 Studio 可以通过设置自动导包,简直太实用了.

到 Preferences -> Editor -> Auto Import -> Java 把以下选项勾上就 OK 了



从此世界清静了。。。

至此 Android Studio 的基本使用相信大家都已经会了,从下一系列开始介绍 Gradle 语法以及基本配置与多渠道打包等,敬请期待!

史上最详细的 Android Studio 系列教程四--Gradle 基础

其实很早之前也写了一篇 Gradle 的基础博客,但是时间很久了,现在 Gradle 已经更新了很多,所以暂且结合 Stduio 1.0 正式版与最新的 Gradle 语法来详细讲解下,小伙伴们直接跟我一步步来学习吧。

什么是 Gradle?

Gradle 是一种依赖管理工具,基于 Groovy 语言,面向 Java 应用为主,它抛弃了基于 XML 的各种繁琐配置,取而代之的是一种基于 Groovy 的内部领域特定(DSL)语言。

安装 Gradle

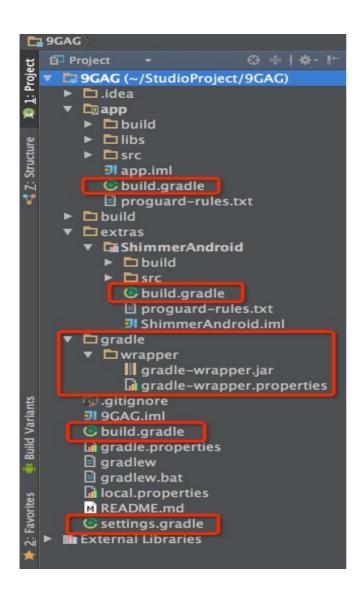
在 Android Studio 系列教程一--下载与安装中新建项目成功后会下载 Gradle,貌似这个过程不翻墙也是可以下载,但是访问特别慢,建议翻墙下载。那么下载的 Gradle 到什么地方呢?

- •Mac 上会默认下载到 /Users/<用户名>/.gradle/wrapper/dists 目录
- •Win 平台会默认下载到 C:\Documents and Settings\<用户名>.gradle\wrapper\dists 目录

你会看到这个目录下有个 gradle-x.xx-all 的文件夹, 如果下载实在太慢,但是又不想翻墙的话,可以自己手动到 Gradle 官网下载对应的版本,然后将下载的.zip 文件(也可以解压)复制到上述的 gradle-x.xx-all 文件夹下,不过还是建议让它直接下载的好。

Gradle 基本概念

下面就以我的开源项目 **9GAG** 来详细讲解下和 Gradle 相关的知识, 和 Gradle 相关的几个文件一般有如下几个:



红色标记部分从上到下咱们来一步步分析:

1. 9GAG/app/build.gradle

这个文件是 app 文件夹下这个 Module 的 gradle 配置文件,也可以算是整个项目最主要的 gradle 配置文件,我们来看下这个文件的内容:

// 声明是 Android 程序

```
apply plugin: 'com.android.application'
android {
 // 编译 SDK 的版本
 compileSdkVersion 21
 // build tools 的版本
 buildToolsVersion "21.1.1"
 defaultConfig {
  // 应用的包名
  applicationId "me.storm.ninegag"
  minSdkVersion 14
  targetSdkVersion 21
  versionCode 1
  versionName "1.0.0"
 // java 版本
 compileOptions {
  sourceCompatibility JavaVersion.VERSION 1 7
   targetCompatibility JavaVersion.VERSION 1 7
 buildTypes {
  debug {
   // debug 模式
  release {
    // 是否进行混淆
    minifyEnabled false
    // 混淆文件的位置
    proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proquard-rules.txt'
```

```
// 移除 lint 检查的 error
lintOptions {
  abortOnError false
}

dependencies {
  // 编译 libs 目录下的所有 jar 包
  compile fileTree(dir: 'libs', include: ['*.jar'])
  compile 'com.android.support:support-v4:21.0.2'
  compile 'com.etsy.android.grid:library:1.0.5'
  compile 'com.alexvasilkov:foldable-layout:1.0.1'
  // 编译 extras 目录下的 ShimmerAndroid 模块
  compile project(':extras:ShimmerAndroid')
}
```

这里需要说明几点:

- •文件开头 apply plugin 是最新 gradle 版本的写法,以前的写法是 apply plugin: 'android', 如果还是以前的写法,请改正过来。
- •buildToolsVersion 这个需要你本地安装该版本才行,很多人导入新的第三方库,失败的原因之一是build version 的版本不对,这个可以手动更改成你本地已有的版本或者打开 **SDK Manager** 去下载对应版本。
- •applicationId 代表应用的包名,也是最新的写法,这里就不在多说了。
- •android 5.0 开始默认安装 jdk1.7 才能编译,但是由于 mac 系统自带 jdk 的版本是 1.6,所以需要手动下载 jdk1.7 并配置下,具体可以见我这篇博客 <u>Mac 下安装和管理 Java</u>
- •minifyEnabled 也是最新的语法,很早之前是 runProguard,这个也需要更新下。
- •proguardFiles 这部分有两段,前一部分代表系统默认的 android 程序的混淆文件,该文件已经包含了基本的混淆声明,免去了我们很多事,这个文件的目录在 <sdk 目录>/tools/proguard/proguard-android.txt,后一部分是我们项目里的自定义的混淆文件,目录就在 app/proguard-rules.txt,如果你用 Studio 1.0 创建的新项目默认生成的文件名是 proguard-rules.pro,这个名字没关系,在这个文件里你可以声明一些第三方依赖的一些混淆规则,由于是开源项目,9GAG 里并未进行混淆,具体混淆的语法也不是本篇博客讨论的范围。最终混淆的结果是这两部分文件共同作用的。
- •compile project(':extras:ShimmerAndroid')这一行是因为 9GAG 中存在其他 Module , 不知道 Module 的 概念可以看下这篇博客 Android Studio 系列教程二--基本设置与运行, 总之你可以理解成 Android Library , 由于 Gradle 的普及以及远程仓库的完善,这种依赖渐渐的会变得非常不常见,但是你需要知道有这种依赖的。
- •以上文件里的内容只是基本配置,其实还有很多自定义部分,如自动打包 debug,release,beta 等环境,签名,多渠道打包等,后续会单独拿出来讲解。

2. 9GAG/extras/ShimmerAndroid/build.gradle

每一个 Module 都需要有一个 gradle 配置文件,语法都是一样,唯一不同的是开头声明的是 apply plugin: 'com.android.library'

3. 9GAG/gradle

这个目录下有个 wrapper 文件夹,里面可以看到有两个文件,我们主要看下 gradle-wrapper.properties 这个文件的内容:

```
#Thu Dec 18 16:02:24 CST 2014
```

distributionBase=GRADLE_USER_HOME

distributionPath=wrapper/dists

zipStoreBase=GRADLE USER HOME

zipStorePath=wrapper/dists

distributionUrl=https\://services.gradle.org/distributions/gradle-2.2.1-all.zip

可以看到里面声明了 gradle 的目录与下载路径以及当前项目使用的 gradle 版本,这些默认的路径我们一般不会更改的,这个文件里指明的 gradle 版本不对也是很多导包不成功的原因之一。

4. 9GAG/build.gradle

这个文件是整个项目的 gradle 基础配置文件,我们来看看这里面的内容

// Top-level build file where you can add configuration options common
to all sub-projects/modules.

```
buildscript {
  repositories {
```

```
jcenter()
}
dependencies {
  classpath 'com.android.tools.build:gradle:1.0.0'
}

allprojects {
  repositories {
   jcenter()
  }
}
```

内容主要包含了两个方面:一个是声明仓库的源,这里可以看到是指明的 jcenter(), 之前版本则是 mavenCentral(), jcenter 可以理解成是一个新的中央远程仓库,兼容 maven 中心仓库,而且性能更优。 另一个是声明了 android gradle plugin 的版本,android studio 1.0 正式版必须要求支持 gradle plugin 1.0 的版本。

5. 9GAG/settings.gradle

这个文件是全局的项目配置文件,里面主要声明一些需要加入 gradle 的 module , 我们来看看 9GAG 该文件的内容:

```
include ':app', ':extras:ShimmerAndroid'
```

文件中的 app, extras:ShimmerAndroid 都是 module , 如果还有其他 module 都需要按照如上格式加进去。

总结

关于 gradle 的基础知识就介绍到这里,接下来会介绍一种我常用的快速方便的编译查看第三方开源项目的方法,如何导入 Android Studio,Gradle 常用基本命令,多渠道打包配置等。有疑问或者发现错误欢迎大家直接博客留言。

Android Studio 系列教程五--Gradle 命令详解与导入第三方包

Android Studio + Gradle 的组合用起来非常方便,很多第三方开源项目也早都迁移到了 Studio , 为此今天就来介绍下查看、编译并导入第三方开源项目的方法。

Sublime + Terminal 编译并查看源码

首先来给大家介绍一种简便并且个人最喜欢的一种办法。很多时候我们在 GitHub 上看到一个不错的开源项目,一般有两种需求,阅读源码和查看运行效果,如果是单纯的查看源码我更喜欢用一些轻量级编辑器,如 vim,sublime等,vim 不是很熟练,所以个人一种都习惯用 sublime 来查看并阅读源码(实际开发还是 Android Studio、Eclipse 等 IDE 更有效率);至于查看运行效果首先得需要 apk,对于一些开源项目已提供 apk 下载,那就好办,直接安装到手机即可。 而对于一些没有提供 apk 下载的,则需要自己手动编译打包。

下面以9GAG为例来讲解下命令行Gradle编译的过程。

•1、切换到 9GAG 项目的根目录,执行 ./gradlew -v 来查看下项目所用的 Gradle 版本

如果你是第一次执行会去下载 Gradle,这个过程如果不翻墙非常慢,建议翻墙

StormMBP:9GAG storm\$./gradlew -v Downloading https://services.gradle.org/distributions/gradle-2.2.1-all.zip
12-14-java-lang-exception 8 ———————————————————————————————————
紧接着下载成功会看到如下信息:

Gradle 2.2.1

Build time: 2014-11-24 09:45:35 UTC

Build number: none

Revision: 6fcb59c06f43a4e6b1bcb401f7686a8601a1fb4a

Groovy: 2.3.6

Ant: Apache Ant(TM) version 1.9.3 compiled on December 23 2013

JVM: 1.7.0_60 (Oracle Corporation 24.60-b09)

OS: Mac OS X 10.9.5 x86_64

•2、接着执行 ./gradlew clean

执行这个命令会去下载 Gradle 的一些依赖,下载成功并编译通过时会看到如下信息:

:app:clean UP-TO-DATE

:extras:ShimmerAndroid:clean UP-TO-DATE

BUILD SUCCESSFUL

•3、最后执行 ./gradlew build

这个命令会直接编译并生成相应的 apk 文件,如果看到如下字样就代表 build 成功了

BUILD SUCCESSFUL

Total time: 31.456 secs

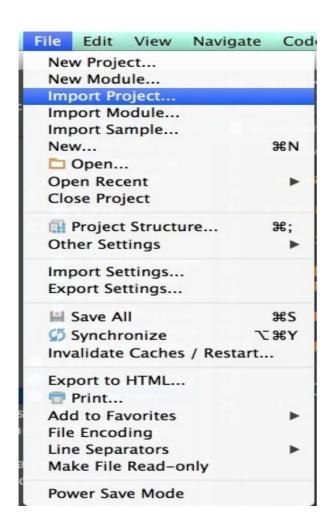
紧接着在 **9GAG/app/build/outputs/apk** 目录下会看到类似于 app-debug-unaligned.apk, app-release-unsigned.apk 等,看名字应该能理解意思,unaligned 代表没有进行 zip 优化的,unsigned 代表没有签名的。然后就可以直接安装 apk 查看运行效果了。

以上是我个人习惯的一种阅读第三方源码的习惯,关于上面提到的一些 gradle 命令大家应该还不理解,后面会一一进行说明的。

导入 Android Studio

但是如果你还是想导入 Android Studio 的话,下面就来简单介绍下导入 Studio 的方法以及一些注意事项。

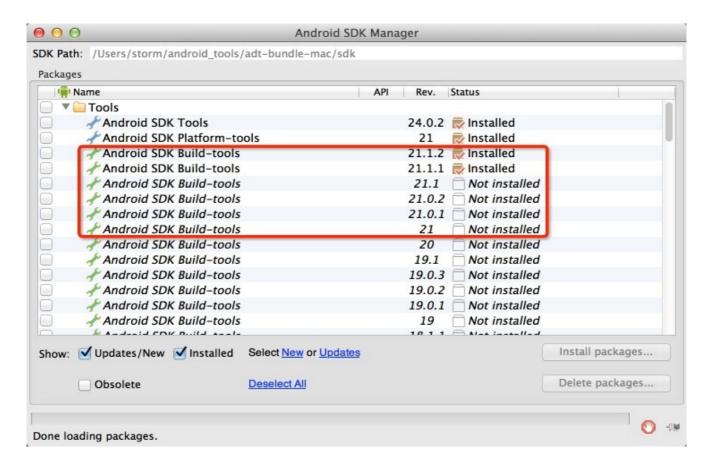
•1、选择 File->Import Project, 选择本地 9GAG 项目的目录



•2、第一次依然会下载 Gradle,其实自己在命令行已经下载过了,但是这次依然还要下载一次(依然要翻墙),可能是个 bug



- •3、导入之后你需要注意以下几个地方
- 1.每个 Module 下的 **build.gradle** 下的 buildToolsVersion,即 **9GAG/app/build.gradle** 和 **9GAG/extras/ShimmerAndroid/build.gradle**,可以打开 **SDK Manager** 查看本地你安装的 **SDK Build-tools**,如果相应版本没有安装请先下载。



1.项目根目录下的 build.gradle 下 gradle 插件的版本,如 9GAG/build.gradle 的内容:

classpath 'com.android.tools.build:gradle:1.0.0'

Android Studio 1.0 必须指定 gradle 插件 1.0 的版本

1.Gradle Wrapper 的版本,在 9GAG/gradle/wrapper/gralde-wrapper.properties 文件里有如下内容:

Thu Dec 18 16:02:24 CST 2014

distributionBase=GRADLE_USER_HOME distributionPath=wrapper/dists zipStoreBase=GRADLE_USER_HOME zipStorePath=wrapper/dists distributionUrl=https://services.gradle.org/distributions/gradle-2.2.1-all.zip

上述内容制定了 gradle 的版本使用 2.2.1,在 Studio 中如果你手动改了 gradle 插件的版本会有提示要不要更新 Gradle Wrapper 的版本。

如果以上几个地方确定版本是 ok 的,那么导入 Android Studio 应该没有问题,有些老的项目可能 gradle 版本比较老,仍然在使用一些老的语法,如果编译还通不过参考 Android Studio 系列教程四--Gradle 基础更改成最新的语法。

Gradle 常用命令

上面大家接触了一些命令如 ./gradlew -v ./gradlew clean ./gradlew build, 这里注意是./gradlew, ./代表当前目录,gradlew 代表 gradle wrapper,意思是 gradle 的一层包装,大家可以理解为在这个项目本地就封装了 gradle,即 gradle wrapper,在 9GAG/gradle/wrapper/gralde-wrapper.properties 文件中声明了它指向的目录和版本。只要下载成功即可用 grdlew wrapper 的命令代替全局的 gradle 命令。

理解了 gradle wrapper 的概念,下面一些常用命令也就容易理解了。

- ./gradlew -v 版本号
- ./gradlew clean 清除 9GAG/app 目录下的 build 文件夹
- ./gradlew build 检查依赖并编译打包

这里注意的是 ./gradlew build 命令把 debug、release 环境的包都打出来,如果正式发布只需要打Release 的包,该怎么办呢,下面介绍一个很有用的命令 assemble, 如

- ./gradlew assembleDebug 编译并打 Debug 包
- ./gradlew assembleRelease 编译并打 Release 的包

除此之外, assemble 还可以和 productFlavors 结合使用, 具体在下一篇多渠道打包进一步解释。

- ./gradlew installRelease Release 模式打包并安装
- ./gradlew uninstallRelease 卸载 Release 模式包

下一篇来介绍用 Gradle 进行方便的多渠道打包,以及项目中完整的 Gradle 配置。有问题或建议欢迎大家直接博客留言。