Android提供了丰富的应用程序框架，它允许你建立创新的应用和游戏在Java语言环境中的移动设备。

Android应用提供多个入口点。App由多个可以单独调用的组件组成。比如：一个单独的Activity提供一个UserInterface，一个service在后台运行。在一个组件中通过intent启动另一个组件（可以是不同应用程序的组件）。这种模式对于单一的应用程序提供多个入口点，并默认允许其他app调用。

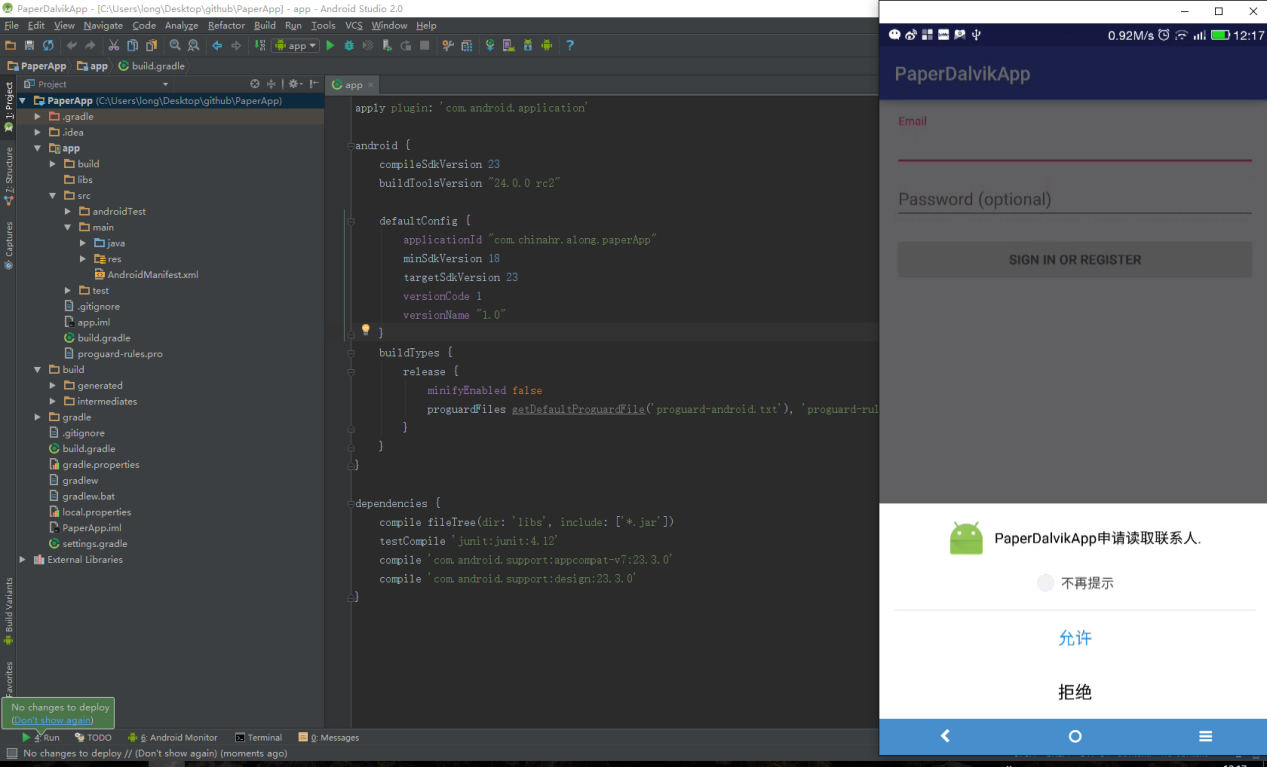
Apps可以适配不同的设备。Android提供了一个自适应的应用程​​序框架，它允许您为不同的设备配置提供独特的资源。例如，您可以创建不同的屏幕大小不同的XML布局文件和系统确定应用基于当前设备的屏幕尺寸，其布局。您可以在运行时查询设备功能的可用性，如果任何应用功能需要特定的硬件，如相机。如果需要，您还可以声明功能，您的应用程序需要这样的应用市场，如谷歌Play商店不允许在不支持该功能的设备的安装。

正常的工作流是这样的：创建，编译，测试，打包。

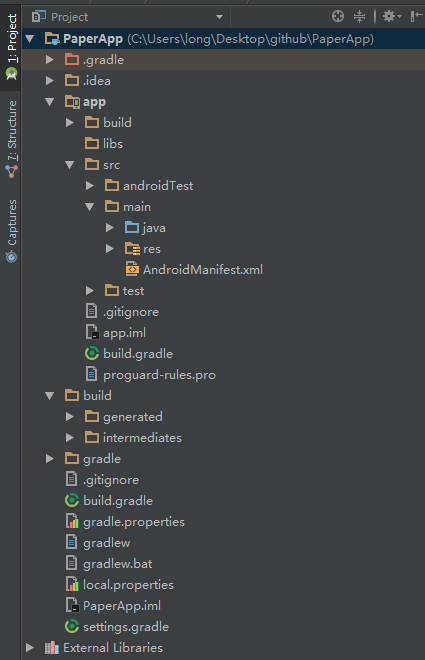
# 开始工作

## 创建应用

我们先创建一个app如下：



项目结构如下：



分为3块：

1. 编译系统：

.gradle 版本由gradle中的wrapper控制,通过gradlew执行

1. 配置文件：

Gradle.properties 与gradle相关属性的配置

Build.gradle 与gradle编译相关的配置

Local.properties 与本地环境配置相关

1. 应用模块：

build 编译后的文件存在的位置（包括最终生成的apk也在这里面）

libs 依赖的库所在的位置（jar和aar)

src 源代码所在的目录

src/main 主要代码所在位置（src/androidTest)就是测试代码所在位置了

src/main/assets android中附带的一些文件

src/main/java 最最重要的，我们的java代码所在的位置

src/main/jniLibs jni的一些动态库所在的默认位置(.so文件)

src/main/res android资源文件所在位置

src/main/AndroidManifest.xml AndroidManifest不用介绍了吧~

build.gradle 和这个项目有关的gradle配置，相当于这个项目的Makefile，一些项目的依赖就写在这里面

proguard.pro 代码混淆配置文件

## 编译，打包应用

构建过程使用许多工具，生成很多中间文件。在Android Studio中，运行gradle的编译任务会完成这一切的构建。构建过程非常灵活可配置。下图描述了参与编译的工具和流程：

标准的构建系统会合并配置在product flavors, build types, dependencies上的资源。过程如下：

1. Android Asset打包工具（AAPT）编译应用程序资源文件，如 AndroidManifest.xml文件，activity的布局XML文件。之后生成R.java，这样就可以从Java代码中引用资源。
2. aidl工具将.aidl文件转化成java的接口。
3. 编译java源码生成class文件。
4. Dex工具将第三方library和cllass文件转化成dex 文件。
5. Apkbuilder将编译的资源，没有编译的资源，dex文件打包成apk文件。
6. Apk文件创建成功后需要签名才能安装到设备上面。最后对齐apk，使应用程序运行时占用更少的内存。如果应用程序达到方法个数65k的限制，需要进行分包处理。
7. 编译后生成的apk输出在build文件中。

## 安装应用

首先，将模拟器执行起来，直到正式进入系统。(注意最好不要同时开着多个模拟器，不然安装时会因不知装那个而出错），如果是手机的 话，插上数据线（注意要确保手机驱动已经安装，不然会找不到改设备）

接着，打开一个cmd窗口，路径切换到模拟器目录下的tools目录下（模拟器在Android sdk内有，也可自己另外下载）

输入：adb devices

(用于搜索你的设备 或 模拟器，如果搜索到就可以对它进行安装。手机设备有可能搜索不到，那是你驱动没装好！）

输入：adb install --(apk完整路径）

例如：adb install paperApp.apk

（1） 拷贝apk文件到指定目录

在Android系统中，apk安装文件是会被保存起来的，默认情况下，用户安装的apk首先会被拷贝到 /data/app 目录下。

/data/app目录是用户有权限访问的目录，在安装apk的时候会自动选择该目录存放用户安装的文件，而系统出厂的apk文件则被放到了 /system 分区下,包括 /system/app，/system/vendor/app，以及 /system/priv-app 等等，该分区只有Root权限的用户才能访问，这也就是为什么在没有Root手机之前，我们无法删除系统出厂的app的原因了。

（2） 解压apk，拷贝文件，创建应用的数据目录

为了加快app的启动速度，apk在安装的时候，会首先将app的可执行文件（dex）拷贝到 /data/dalvik-cache 目录，缓存起来。

然后，在/data/data/目录下创建应用程序的数据目录（以应用的包名命名），存放应用的相关数据，如数据库、xml文件、cache、二进制的so动态库等等。

（3） 解析apk的AndroidManifinest.xml文件

Android系统中，也有一个类似注册表的东西，用来记录当前所有安装的应用的基本信息，每次系统安装或者卸载了任何apk文件，都会更新这个文件。这个文件位于如下目录：

/data/system/packages.xml

系统在安装apk的过程中，会解析apk的AndroidManifinest.xml文件，提取出这个apk的重要信息写入到packages.xml文件中，这些信息包括：权限、应用包名、APK的安装位置、版本、userID等等。

由此，我们就知道了为啥一些应用市场和软件管理类的app能够很清楚地知道当前手机所安装的所有的app，以及这些app的详细信息了。

（4） 其他操作

与windows应用安装类似，部分apk的安装也会向Launcher应用申请添加创建快捷方式。

# 运行时