**深 圳 大 学 实 验 报 告**

**课程名称： 移动设备交互应用**

**实验项目名称： 熟悉Android编程开发环境**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 解为成**

**报告人： 沈晨玙 学号：2019092121 班级： 19国际班**

**实验时间： 2020.9.23**

**实验报告提交时间： 2020.9.23**

**教务处制**

一、实验目的与内容：

**目的：**掌握面向Android编程的开发环境搭建。学习、掌握Android程序编写基本步骤，例如，Android Studio平台编写简单的一个 HelloWorld程序，掌握编译、运行等基本步骤和操作。

**内容要求：**

1. 搭建开发环境（基本要求，60分）：

（在自己电脑上）下载安装和配置Android Studio开发环境，并熟悉该开发环境中的常用操作，并 New -> New Project ->Base Activity一个项目，而且能运行成功。请在报告中给出搭建环境成功的截图和简要文字说明，简述自己所了解掌握的常用操作。

2. 新建的程序项目解析（提升要求，40分）

通过查阅相关资料，尝试对前面新建的Base Activity项目进行详细的解析，包括代码的结构、每一部分的功能、每行代码表示的意义等。总体要求是越完整越好，越详尽越好。

注意：

1. 实验报告中需要有实验结果的截屏图像。

二、实验过程和代码与结果

**1.开发环境的搭建 实验过程及结果**

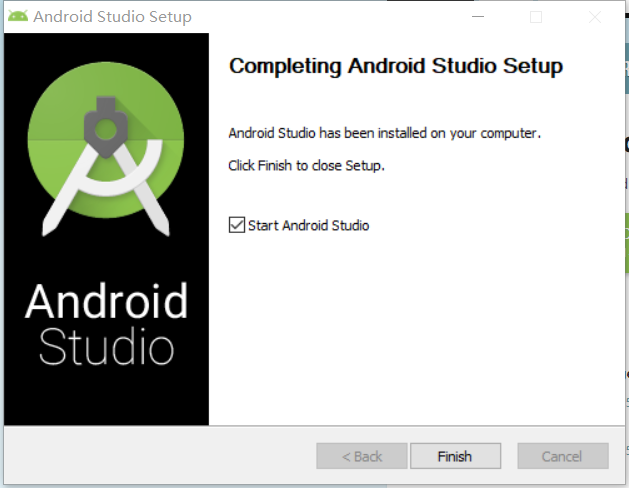
**1.1 Android Studio的安装及环境搭建**

**1.1.1** 前往Android Studio官网下载Android Studio 4.0安装包

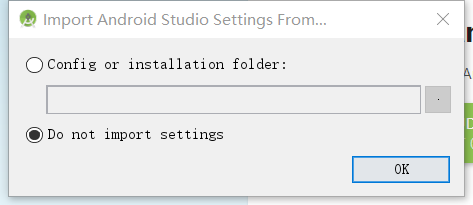
官网 http://www.android-studio.org/



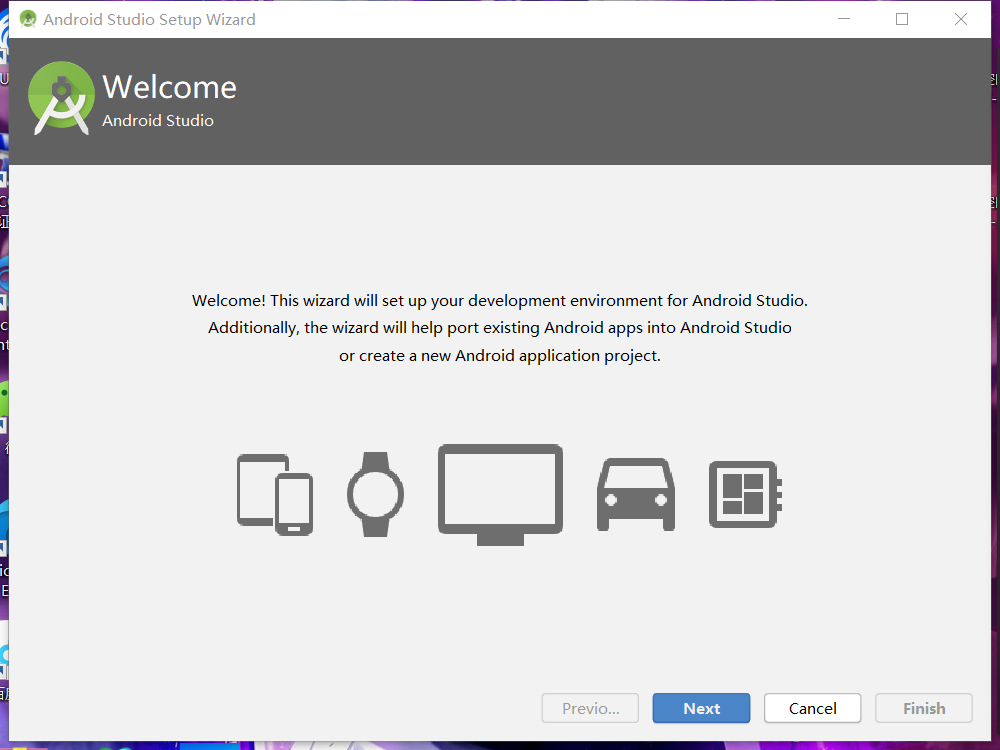
**1.1.2** 根据软件安装包的提示进行安装



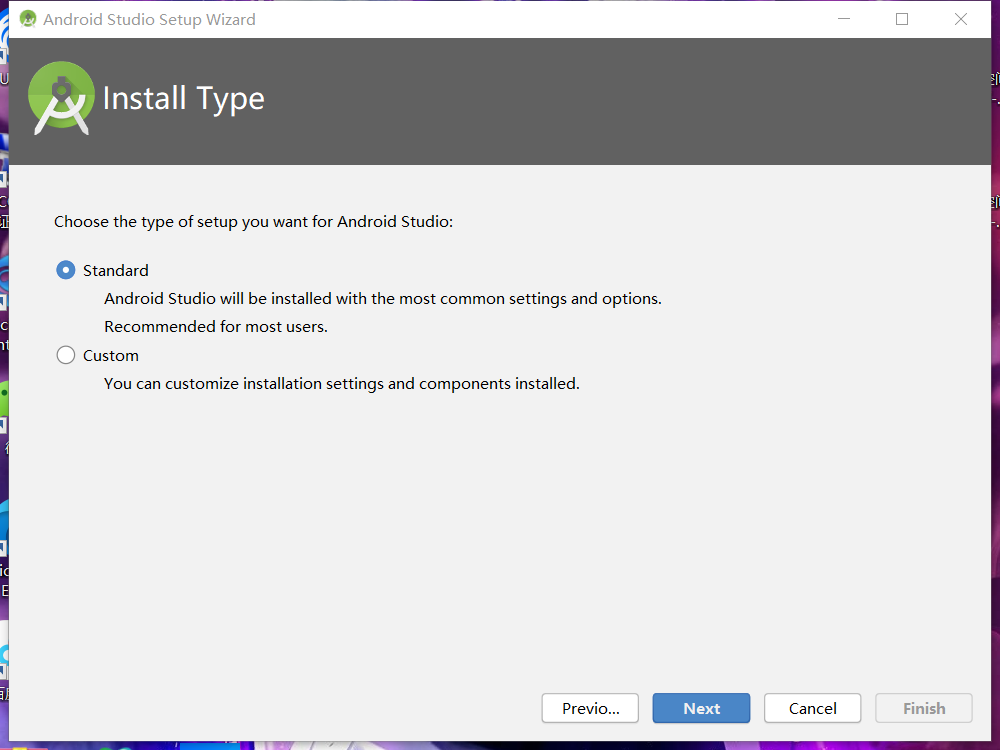
首次启动会让你选择是否导入之前Android Studio版本的配置，由于这是我们首次安装，选择不导入即可



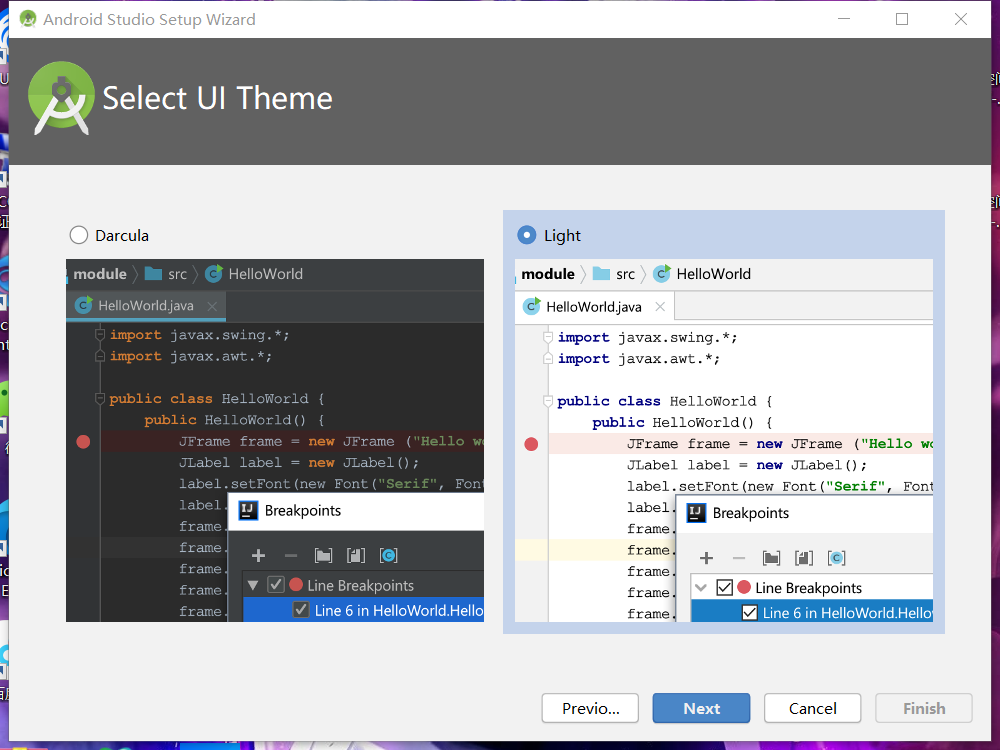
选择Next下一步



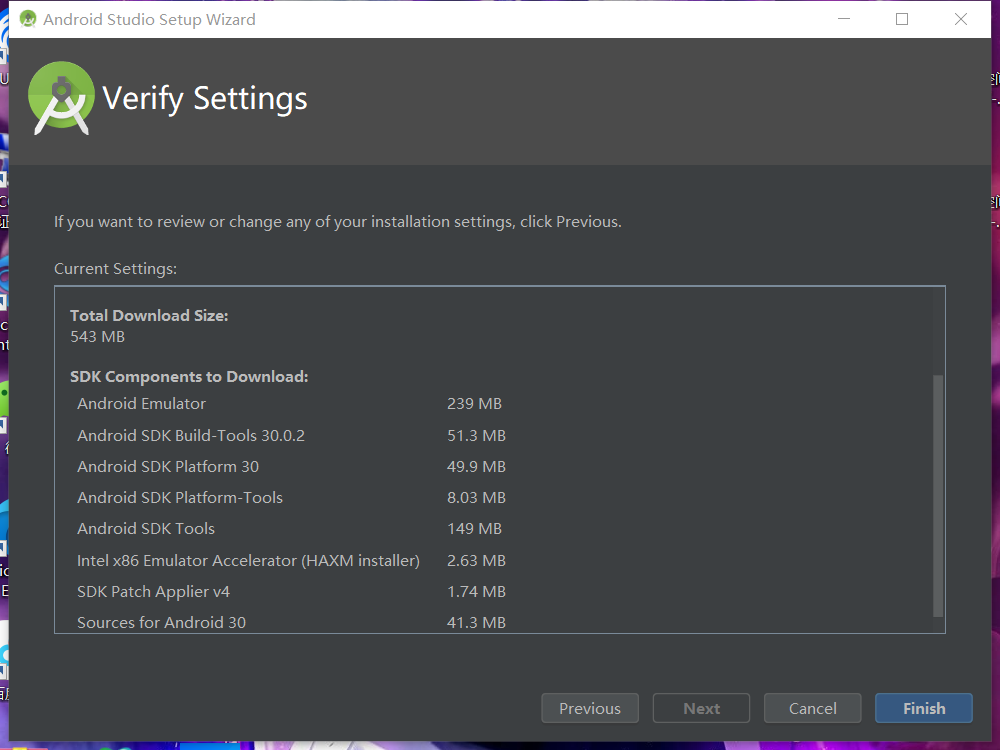
选择标准安装

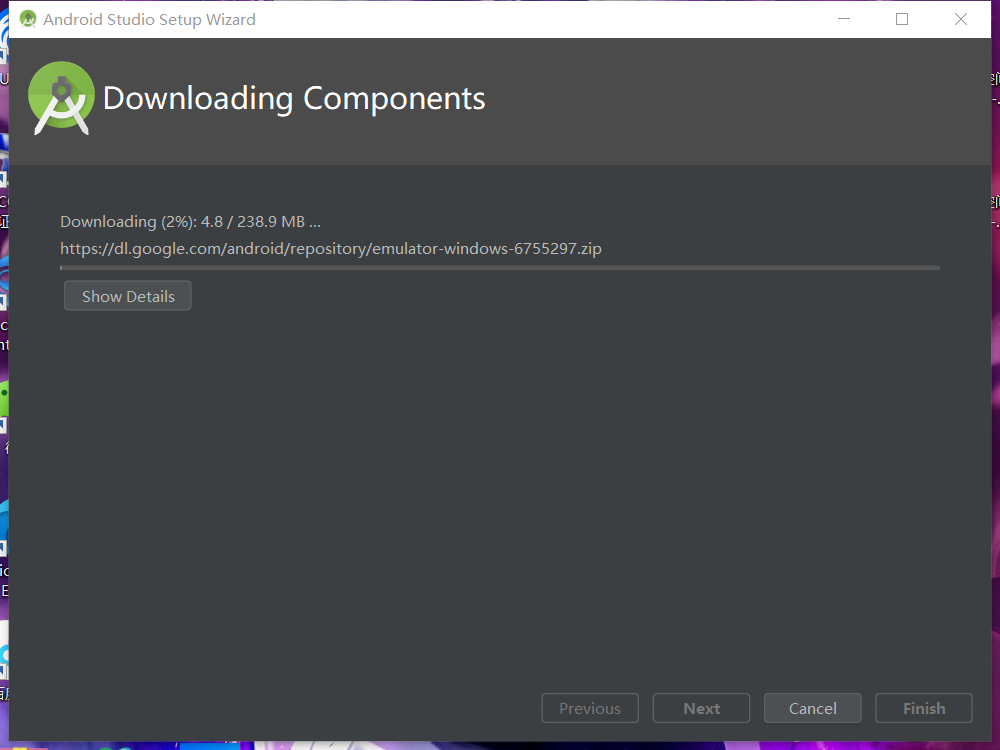


根据个人喜好选择主题风格



下载并安装所需要的插件

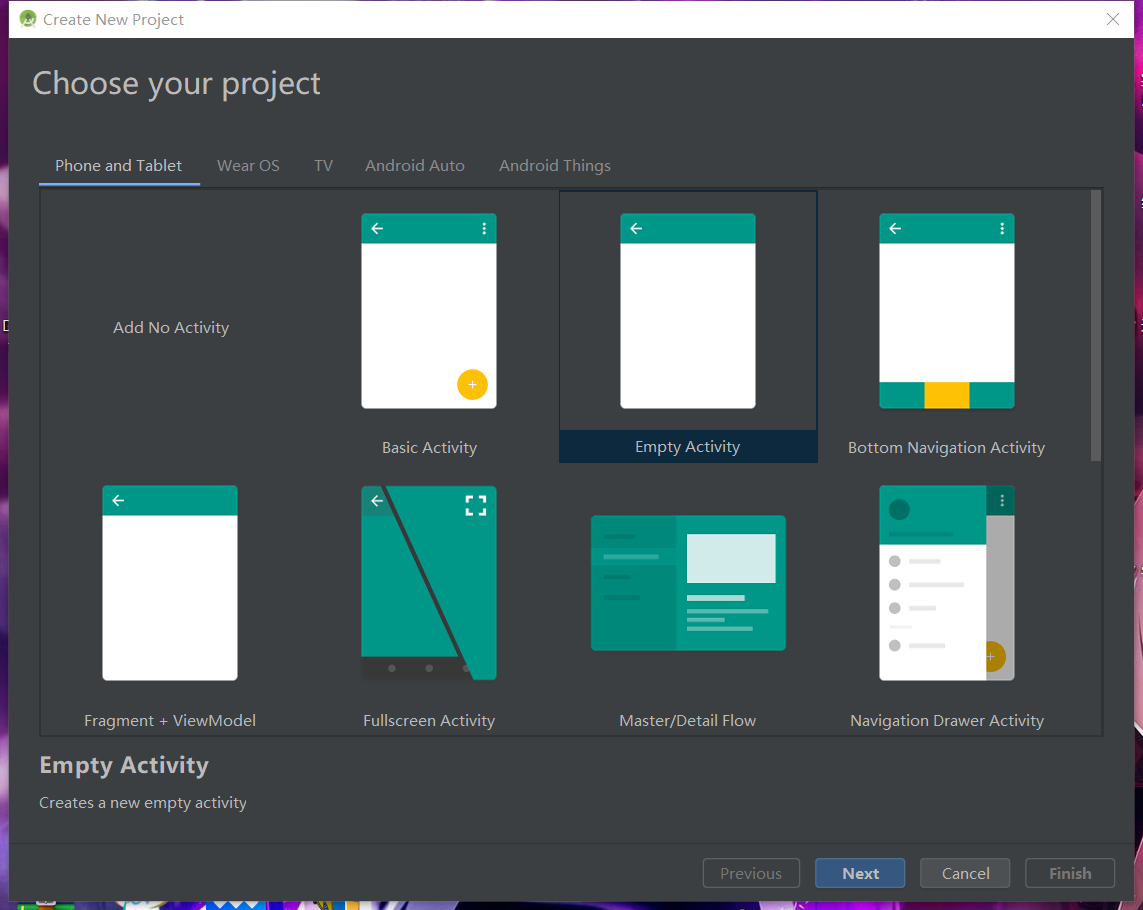
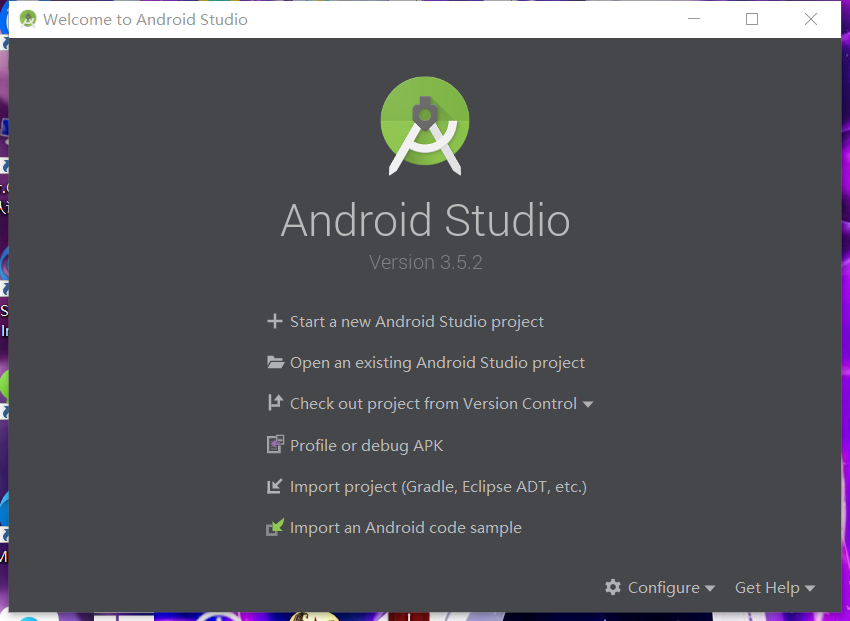
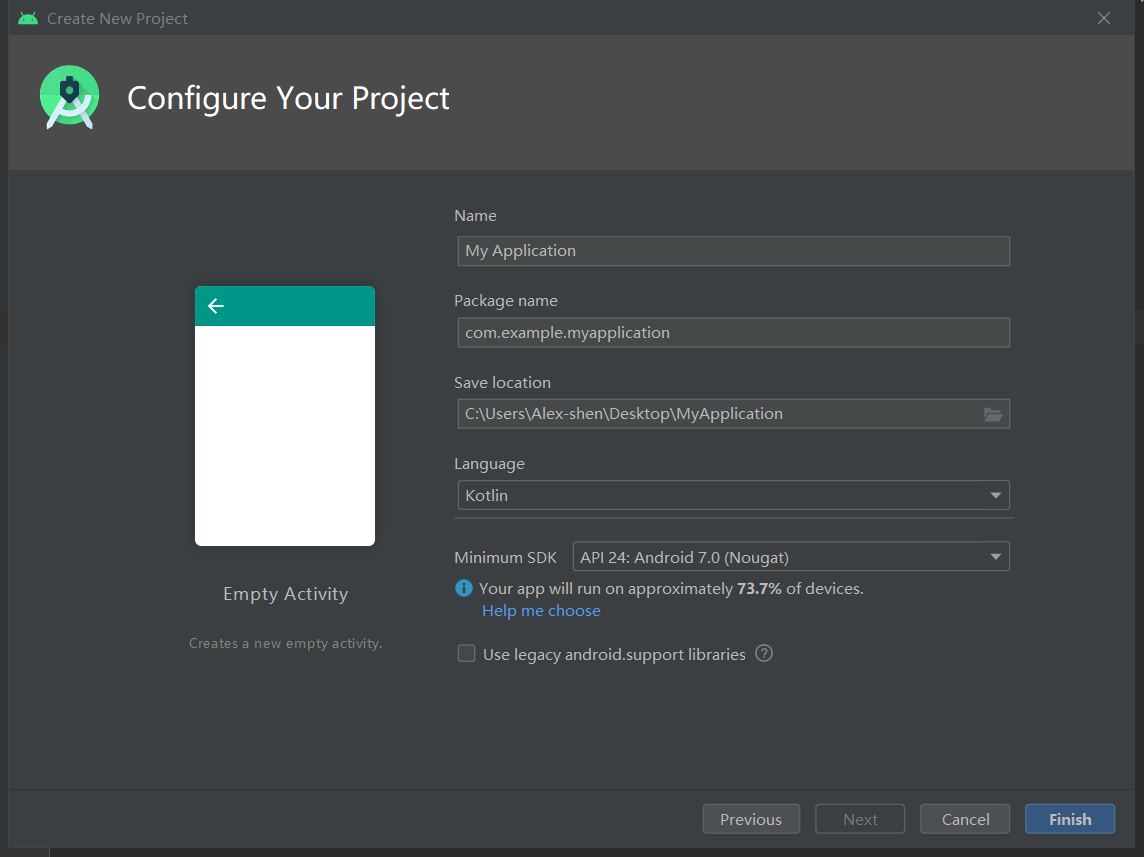


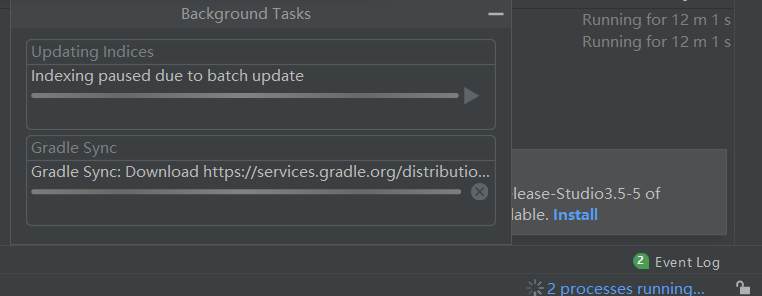


安装成功。

**1.2 Android Studio 环境测试与软件试运行**

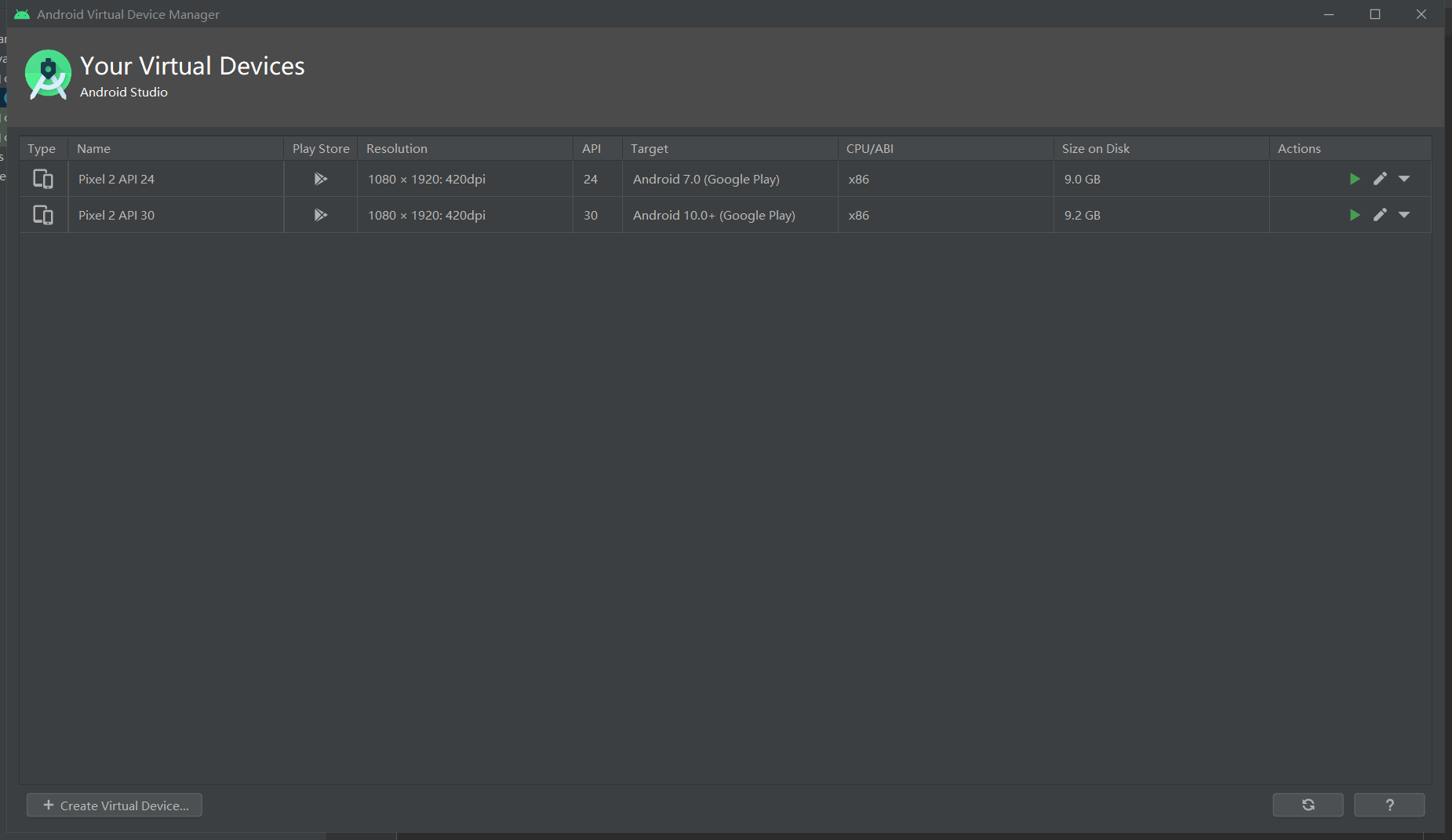
**1.2.1** 创建一个新的Empty Activity

****

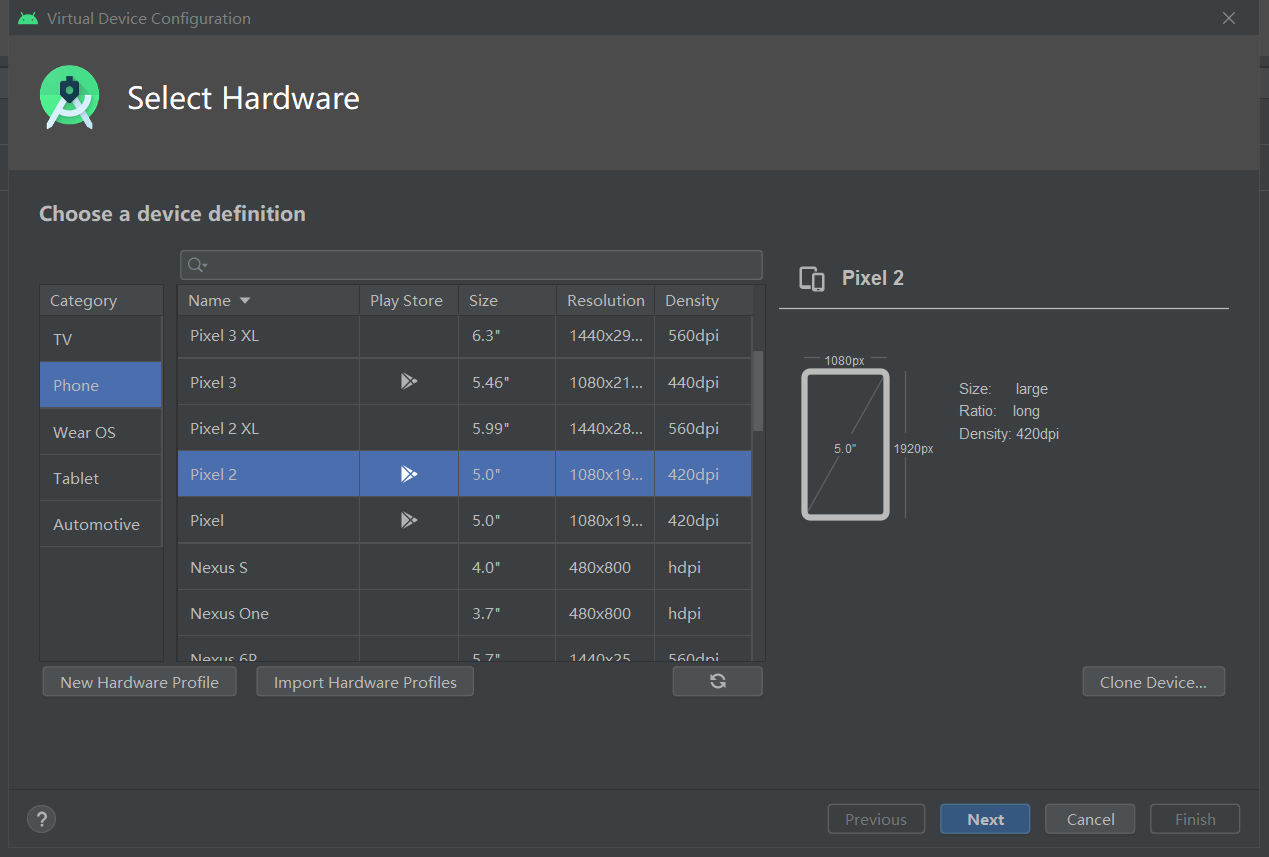
注：第一次运行Android Studio需要较长时间进行软件包在线安装，加速方法见实验总结。****

**1.2.2** 安装AVD虚拟机

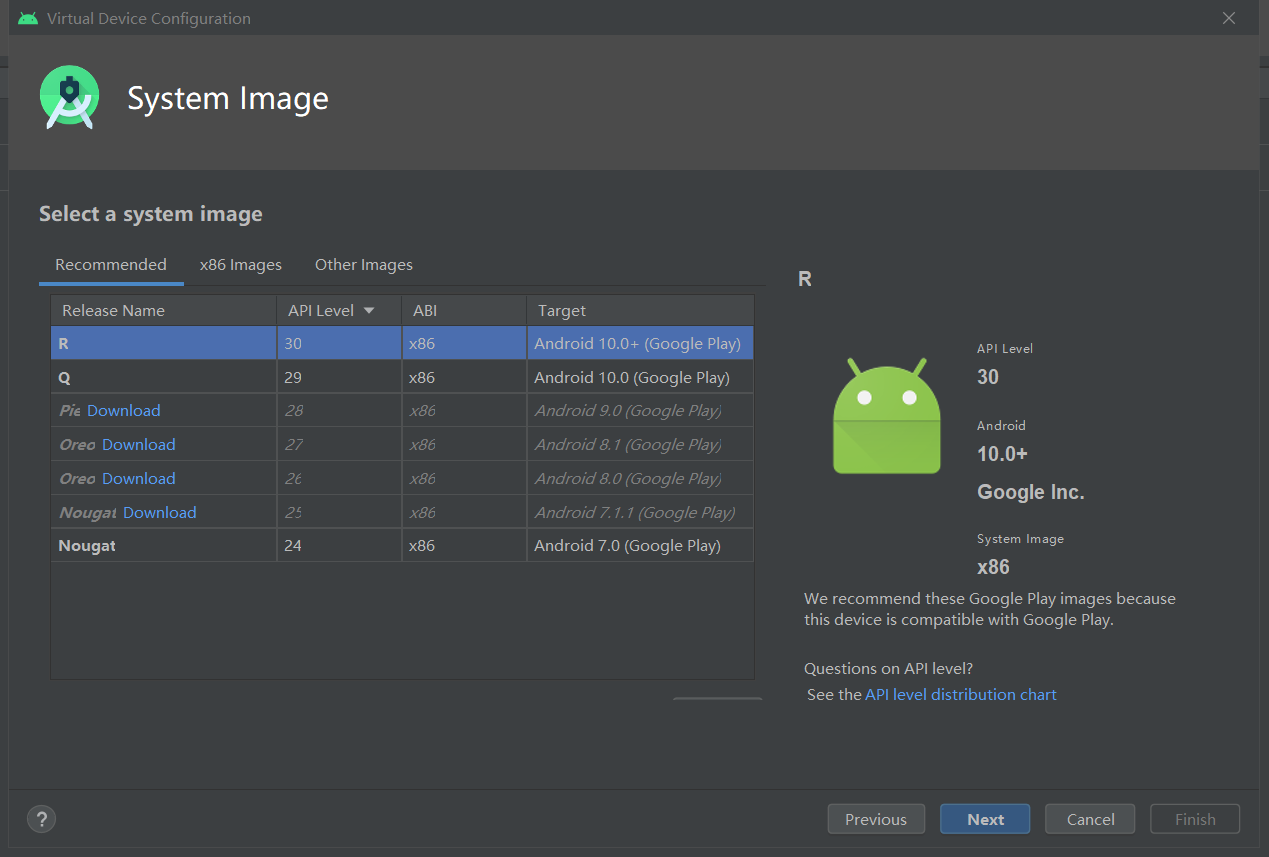
点击导航栏下方中间的图标 进入AVD安装界面



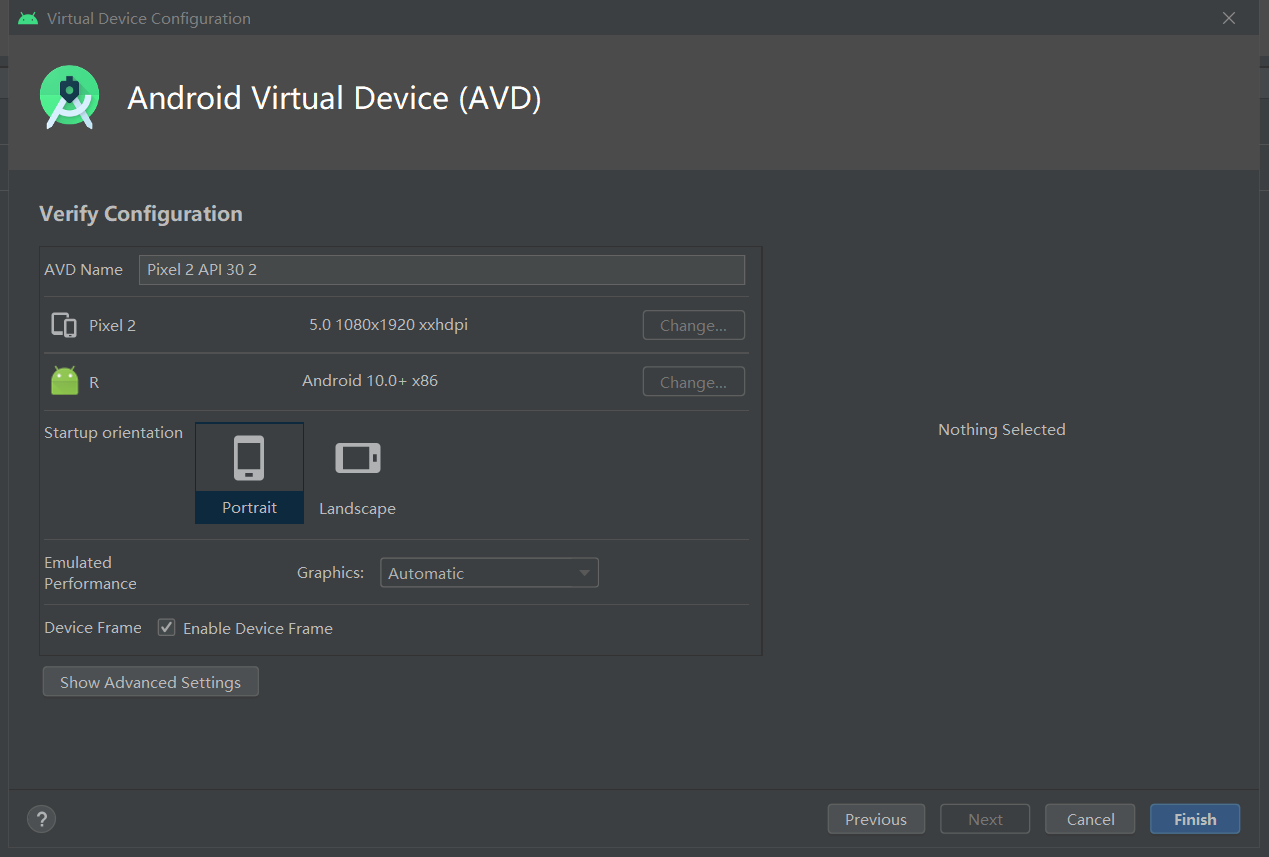
点击下方Create Virtual Device，选择需要安装的虚拟机设备（TV,Phone,Wear OS,Tablet,Automotive)，此处选择Pixel 2作为测试案例。



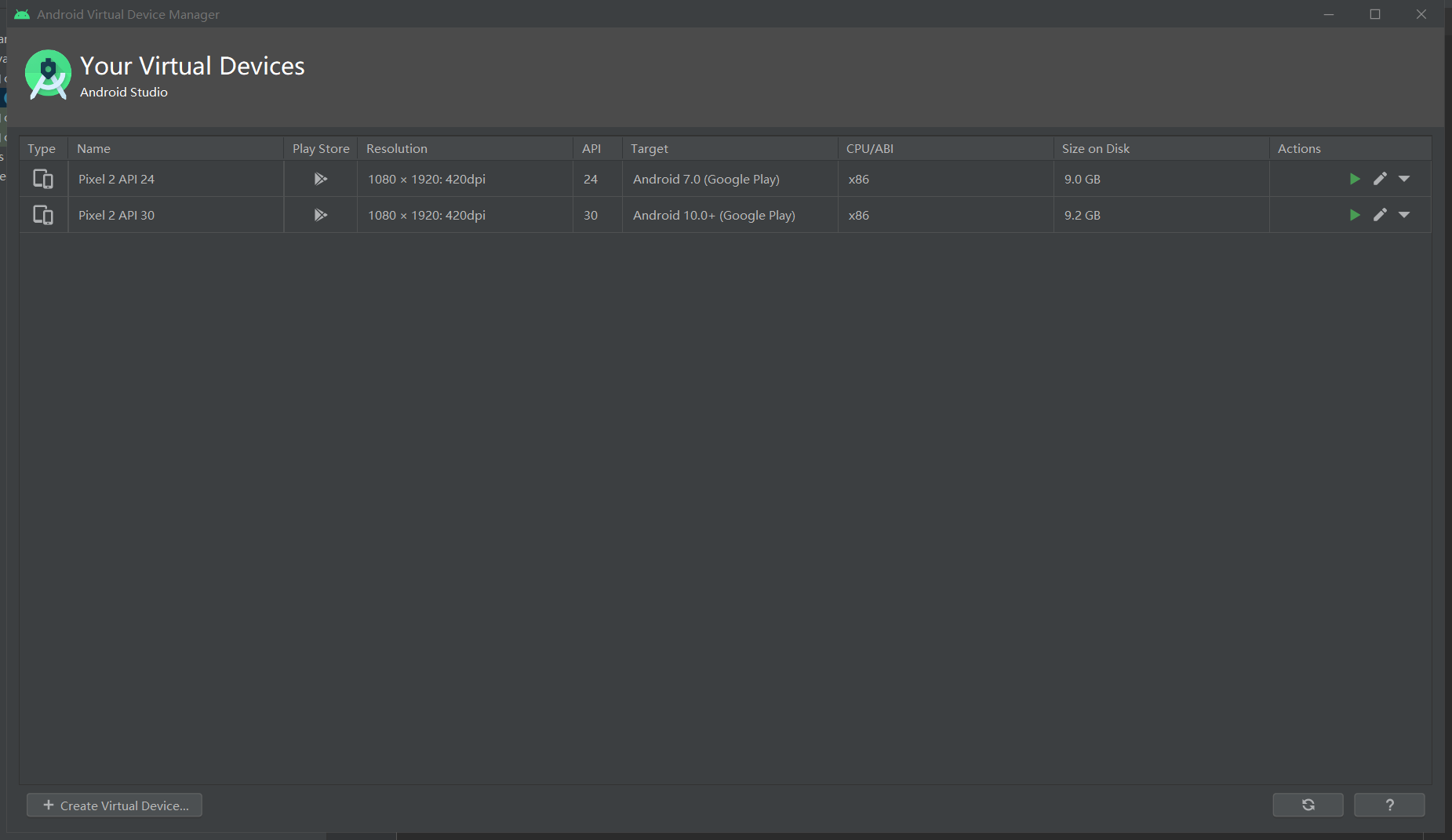
选择下载需要安装的安卓系统版本，如未下载过，点击Download按照提示下载。此处选择Android 10.0+作为测试案例



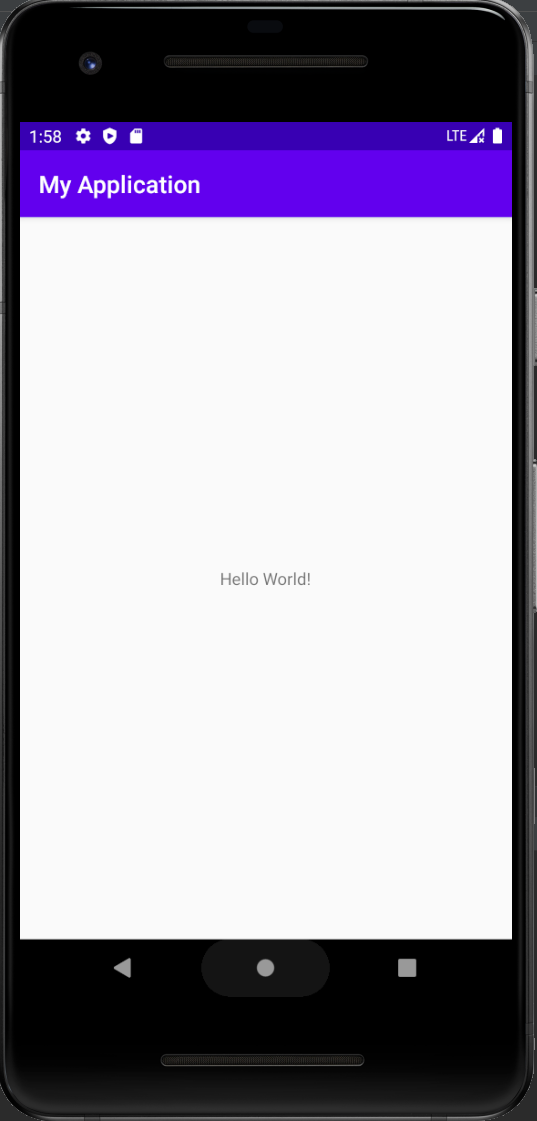
在这里我们可以对模拟器的一些配置进行确认，比如说指定模拟器的名字、分辨率、横竖屏等信息，如果没有特殊需求的话，全部保持默认就可以了。点击“Finish”完成模拟器的创建。



可以看到，现在模拟器列表中已经存在一个创建好的模拟器设备了，点击Actions栏目中最左边的三角形按钮即可启动模拟器。模拟器会像手机一样，有一个开机过程，启动完成之后的界面如图所示。

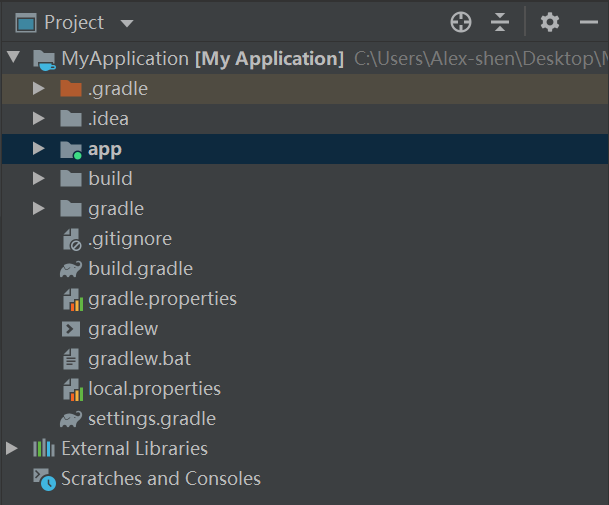
**1.2.3** 运行Hello World程序

现在模拟器已经启动起来了，观察Android Studio顶部工具栏中的图标，左边的锤子按钮是用来编译项目的。中间有两个下拉列表：一个是用来选择运行哪一个项目的，通常app就是当前的主项目；另一个是用来选择运行到哪台设备上的。点击三角形按钮即可在虚拟机上运行HelloWorld程序，效果如图所示。

1. **自己建立的Base Activity程序的解析过程**

**2.1 项目结构解析**

首先展开HelloWorld项目，你会看到如图所示的项目结构。首先将对所有文件夹进行部分功能解析。



1. .gradle和.idea

这两个目录下放置的都是Android Studio自动生成的一些文件，无须关心，也不要去手动编辑。

2. app

项目中的代码、资源等内容都是放置在这个目录下的，后面的开发工作也基本是在这个目录下进行的，待会儿还会对这个目录单独展开讲解。

3. build

这个目录主要包含了一些在编译时自动生成的文件，也不需要过多关心。

4. gradle

这个目录下包含了gradle wrapper的配置文件，使用gradle wrapper的方式不需要提前将gradle下载好，而是会自动根据本地的缓存情况决定是否需要联网下载gradle。Android Studio默认就是启用gradle wrapper方式的，如果需要更改成离线模式，可以点击Android Studio导航栏→File→Settings→Build, Execution, Deployment→Gradle，进行配置更改。

5. .gitignore

这个文件是用来将指定的目录或文件排除在版本控制之外的。

6. build.gradle

这是项目全局的gradle构建脚本，通常这个文件中的内容是不需要修改的。

7. gradle.properties

这个文件是全局的gradle配置文件，在这里配置的属性将会影响到项目中所有的gradle编译脚本。

8. gradlew和gradlew.bat

这两个文件是用来在命令行界面中执行gradle命令的，其中gradlew是在Linux或Mac系统中使用的，gradlew.bat是在Windows系统中使用的。

9. HelloWorld.iml

iml文件是所有IntelliJ IDEA项目都会自动生成的一个文件（Android Studio是基于IntelliJ IDEA开发的），用于标识这是一个IntelliJ IDEA项目，我们不需要修改这个文件中的任何内容。

10. local.properties

这个文件用于指定本机中的Android SDK路径，通常内容是自动生成的，我们并不需要修改。除非你本机中的Android SDK位置发生了变化，那么就将这个文件中的路径改成新的位置即可。

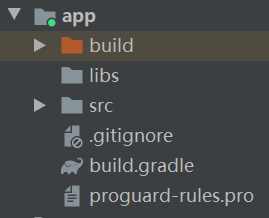
11. settings.gradle

这个文件用于指定项目中所有引入的模块。由于HelloWorld项目中只有一个app模块，因此该文件中也就只引入了app这一个模块。通常情况下，模块的引入是自动完成的，需要手动修改这个文件的场景可能比较少。

现在整个项目的外层目录结构已经介绍完了。通过介绍会发现，除了app目录之外，大多数的文件和目录是自动生成的，并不需要进行修改，所以开发的主要变化在于app目录下。

**2.2 app目录解析**

app目录下的内容展开之后的结构如图所示。



1. build

这个目录和外层的build目录类似，也包含了一些在编译时自动生成的文件，不过它里面的内容会更加更杂，不需要过多关心。

2. libs

如果项目中使用到了第三方jar包，就需要把这些jar包都放在libs目录下，放在这个目录下的jar包会被自动添加到项目的构建路径里。

3. androidTest

此处是用来编写Android Test测试用例的，可以对项目进行一些自动化测试。

4. java

毫无疑问，java目录是放置我们所有Java代码的地方（Kotlin代码也放在这里），展开该目录，可以看到系统帮我们自动生成了一个MainActivity文件。

5. res

这个目录下的内容就有点多了。简单点说，就是你在项目中使用到的所有图片、布局、字符串等资源都要存放在这个目录下。当然这个目录下还有很多子目录，图片放在drawable目录下，布局放在layout目录下，字符串放在values目录下，所以不用担心会把整个res目录弄得乱糟糟的。

6. AndroidManifest.xml

这是整个Android项目的配置文件，在程序中定义的所有四大组件都需要在这个文件里注册，另外还可以在这个文件中给应用程序添加权限声明。

7. test

此处是用来编写Unit Test测试用例的，是对项目进行自动化测试的另一种方式。

8. .gitignore

这个文件用于将app模块内指定的目录或文件排除在版本控制之外，作用和外层的.gitignore文件类似。

9. app.iml

IntelliJ IDEA项目自动生成的文件，不需要关心或修改这个文件中的内容。

10. build.gradle

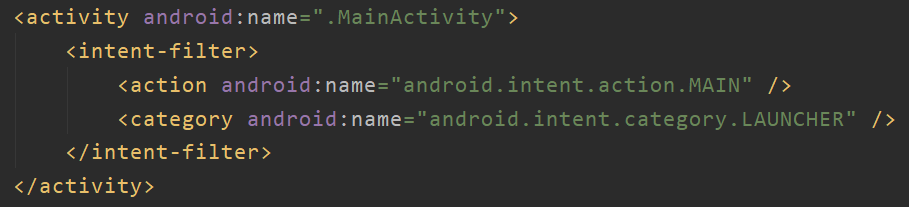
这是app模块的gradle构建脚本，这个文件中会指定很多项目构建相关的配置。

11. proguard-rules.pro

这个文件用于指定项目代码的混淆规则，当代码开发完成后打包成安装包文件，如果不希望代码被别人破解，通常会将代码进行混淆，从而让破解者难以阅读。

**2.3 HelloWorld程序解析**

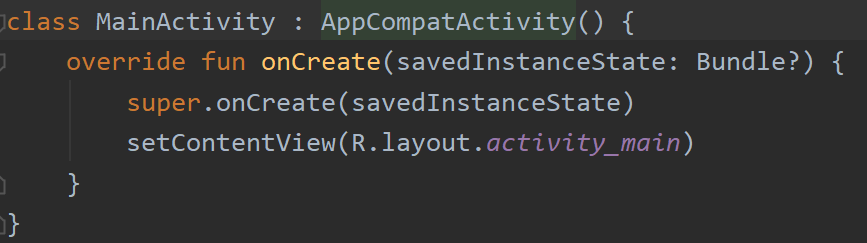
首先打开Android- Manifest.xml文件，从中可以找到如下代码：



这段代码表示对MainActivity进行注册，没有在AndroidManifest.xml里注册的Activity是不能使用的。其中intent-filter里的两行代码非常重要，<actionandroid:name="android.intent.action.MAIN"/>和<category android:name ="android.intent.category.LAUNCHER" />表示MainActivity是这个项目的主Activity，在手机上点击应用图标，首先启动的就是这个Activity。

那MainActivity具体作用是Android应用程序的门面，凡是在应用中你看得到的东西，都是放在Activity中的。

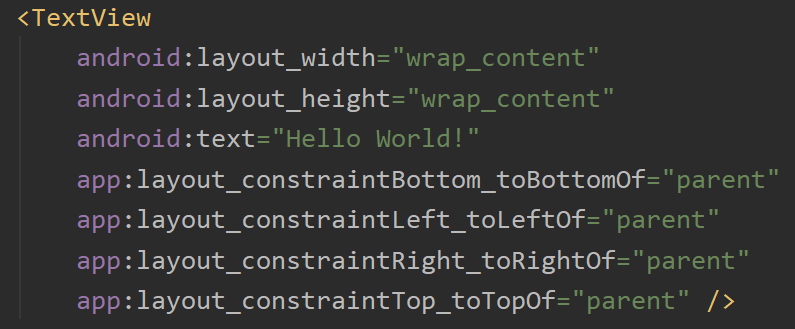
打开MainActivity，代码如下所示。



首先可以看到，MainActivity是继承自AppCompatActivity的。AppCompatActivity是AndroidX中提供的一种向下兼容的Activity，可以使Activity在不同系统版本中的功能保持一致性。而Activity类是Android系统提供的一个基类，我们项目中所有自定义的Activity都必须继承它或者它的子类才能拥有Activity的特性。然后可以看到MainActivity中有一个onCreate()方法，这个方法是一个Activity被创建时必定要执行的方法。

Android程序的设计讲究逻辑和视图分离，因此是不推荐在Activity中直接编写界面的。一种更加通用的做法是，在布局文件中编写界面，然后在Activity中引入进来。可以看到，在onCreate()方法的第二行调用了setContentView()方法，就是这个方法给当前的Activity引入了一个activity\_main布局，所以“Hello World!”就是在这里定义的了。

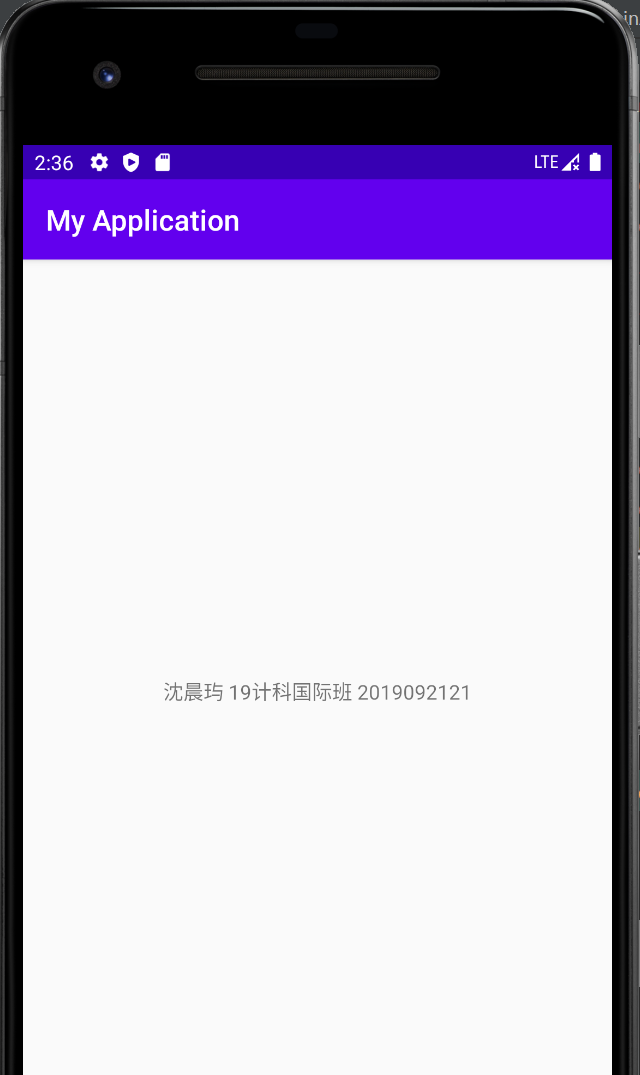
打开activity\_main.xml，代码如下所示。



看到上面代码中有一个TextView，这是Android系统提供的一个控件，用于在布局中显示文字。终于在TextView中看到了“Hello World！”的字样，原来就是通过android:text="Hello World!"这句代码定义的。

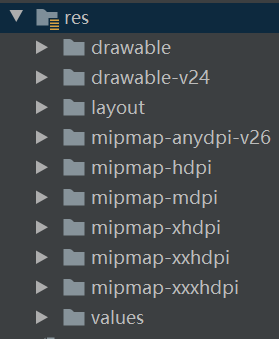
作为验证，修改其中的文本信息为本人个人信息，并在虚拟机中进行展示。





**2.4 res目录解析**

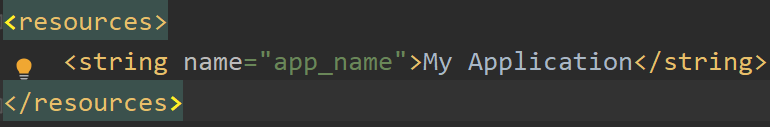
Res目录下的内容展开之后的结构如图所示。



所有以“drawable”开头的目录都是用来放图片的，所有以“mipmap”开头的目录都是用来放应用图标的，所有以“values”开头的目录都是用来放字符串、样式、颜色等配置的，所有以“layout”开头的目录都是用来放布局文件的。

之所以有这么多“mipmap”开头的目录，其实主要是为了让程序能够更好地兼容各种设备。drawable目录也是相同的道理，虽然Android Studio没有帮我们自动生成，但是我们应该自己创建drawable-hdpi、drawable-xhdpi、drawable-xxhdpi等目录。在制作程序的时候，最好能够给同一张图片提供几个不同分辨率的版本，分别放在这些目录下，然后程序运行的时候，会自动根据当前运行设备分辨率的高低选择加载哪个目录下的图片。当然这只是理想情况，更多的时候美工只会提供给我们一份图片，这时你把所有图片都放在drawable-xxhdpi目录下就好了，因为这是最主流的设备分辨率目录。

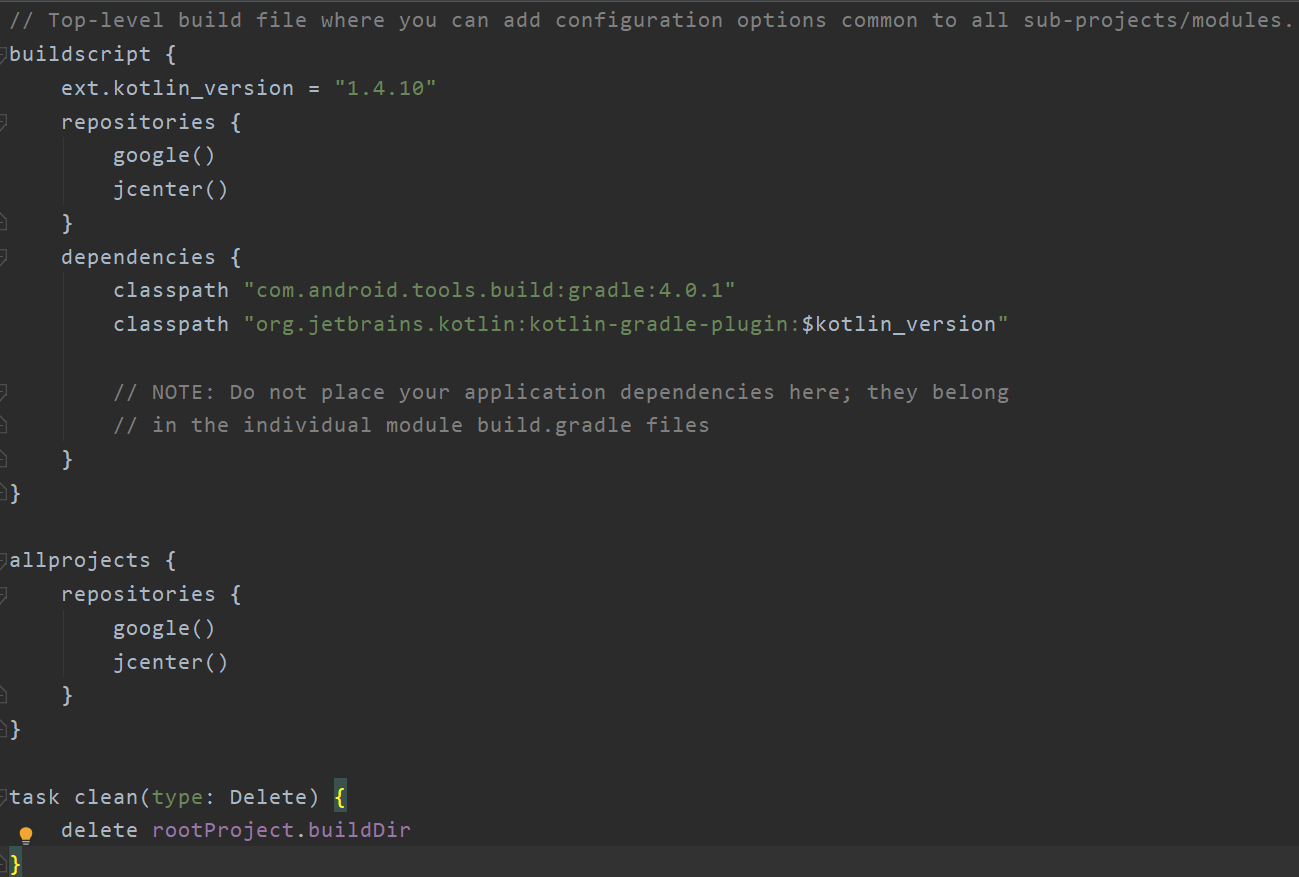
打开res/values/ strings.xml文件，内容如下所示：



可以看到，这里定义了一个应用程序名的字符串，在XML中通过@string/app\_name可以获得该字符串的引用。其中string部分是可以替换的，如果是引用的图片资源就可以替换成drawable，如果是引用的应用图标就可以替换成mipmap，如果是引用的布局文件就可以替换成layout，以此类推。

**2.5 build.gradle文件解析**

**2.5.1 最外层目录下的build.gradle文件**

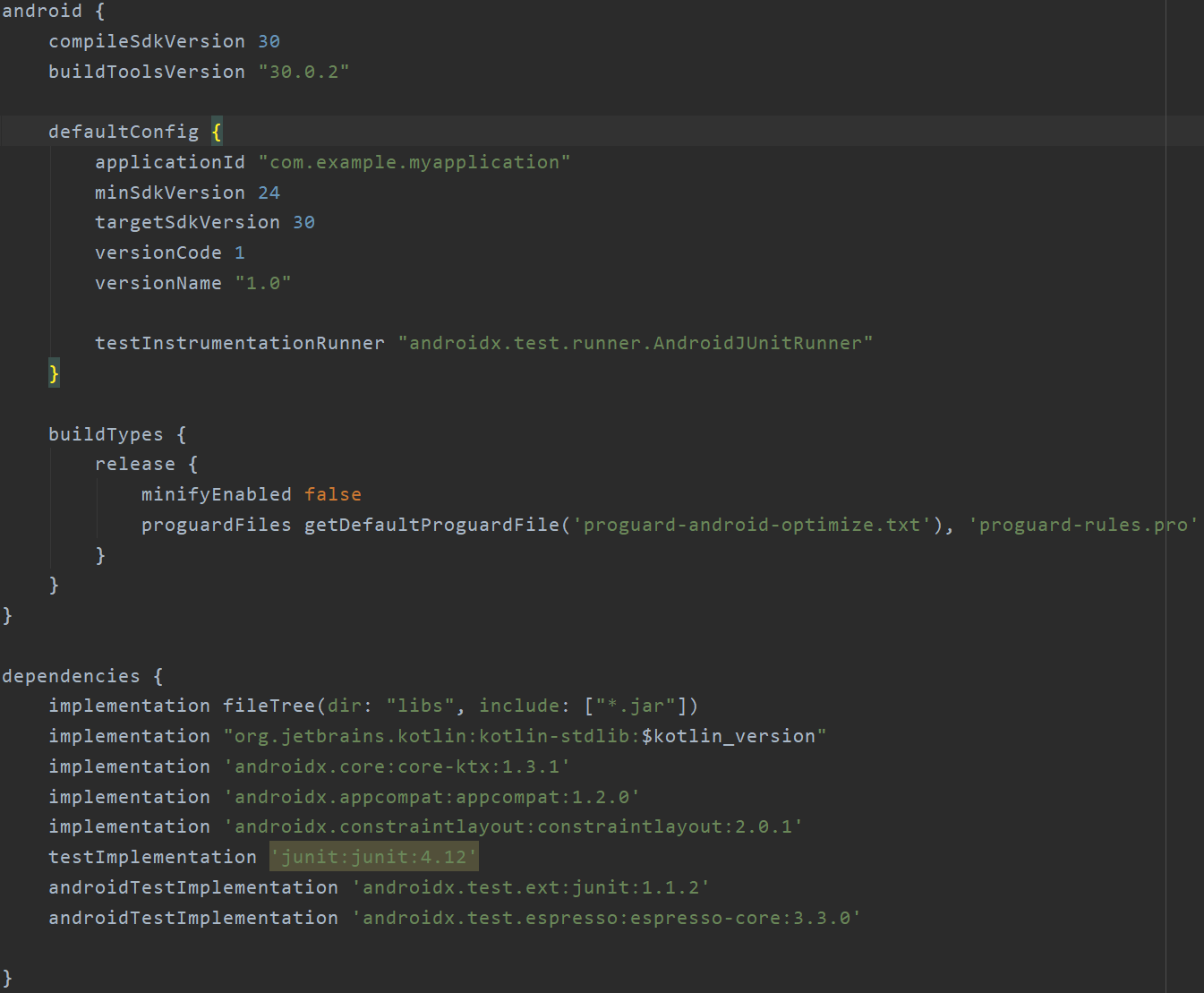


首先，两处repositories的闭包中都声明了google()和jcenter()这两行配置，它们分别对应了一个代码仓库，google仓库中包含的主要是Google自家的扩展依赖库，而jcenter仓库中包含的大多是一些第三方的开源库。声明了这两行配置之后，就可以在项目中轻松引用任何google和jcenter仓库中的依赖库了。

接下来，dependencies闭包中使用classpath声明了两个插件：一个Gradle插件和一个Kotlin插件。因为Gradle并不是专门为构建Android项目而开发的，Java、C++等很多种项目也可以使用Gradle来构建，因此如果我们要想使用它来构建Android项目，则需要声明com.android.tools.build:gradle:4.0.1这个插件。其中，最后面的部分是插件的版本号，它通常和当前Android Studio的版本是对应的，比如我现在使用的是Android Studio 4.0.1版本，那么这里的插件版本号就应该是4.0.1。而另外一个Kotlin插件则表示当前项目是使用Kotlin进行开发的，如果是Java版的Android项目，则不需要声明这个插件。

通常情况下，你并不需要修改这个文件中的内容，除非你想添加一些全局的项目构建配置。

**2.5.2 app目录下的build.gradle文件**



首先第一行应用了一个插件，一般有两种值可选：com.android.application表示这是一个应用程序模块，com.android.library表示这是一个库模块。二者最大的区别在于，应用程序模块是可以直接运行的，库模块只能作为代码库依附于别的应用程序模块来运行。

接下来的两行应用了kotlin-android和kotlin-android-extensions这两个插件。如果想要使用Kotlin来开发Android项目，那么第一个插件就是必须应用的。而第二个插件帮助我们实现了一些非常好用的Kotlin扩展功能。

紧接着是一个大的android闭包，在这个闭包中可以配置项目构建的各种属性。其中，compileSdkVersion用于指定项目的编译版本，这里指定成30表示使用Android 10.0系统的SDK编译。buildToolsVersion用于指定项目构建工具的版本，目前最新的版本就是30.0.2，如果有更新的版本时，Android Studio会进行提示。

然后我们看到，android闭包中又嵌套了一个defaultConfig闭包，defaultConfig闭包中可以对项目的更多细节进行配置。其中，applicationId是每一个应用的唯一标识符，绝对不能重复，默认会使用我们在创建项目时指定的包名，如果你想在后面对其进行修改，那么就是在这里修改的。minSdkVersion用于指定项目最低兼容的Android系统版本，这里指定成24表示最低兼容到Android 7.0系统。targetSdkVersion指定的值表示你在该目标版本上已经做过了充分的测试，系统将会为你的应用程序启用一些最新的功能和特性。接下来的两个属性都比较简单，versionCode用于指定项目的版本号，versionName用于指定项目的版本名。最后，testInstrumentationRunner用于在当前项目中启用JUnit测试，可以为当前项目编写测试用例，以保证功能的正确性和稳定性。

分析完了defaultConfig闭包，接下来我们看一下buildTypes闭包。buildTypes闭包中用于指定生成安装文件的相关配置，通常只会有两个子闭包：一个是debug，一个是release。debug闭包用于指定生成测试版安装文件的配置，release闭包用于指定生成正式版安装文件的配置。另外，debug闭包是可以忽略不写的，因此我们看到上面的代码中就只有一个release闭包。下面来看一下release闭包中的具体内容吧，minifyEnabled用于指定是否对项目的代码进行混淆，true表示混淆，false表示不混淆。proguardFiles用于指定混淆时使用的规则文件，这里指定了两个文件：第一个proguard-android-optimize.txt是在/tools/proguard目录下的，里面是所有项目通用的混淆规则；第二个proguard-rules.pro是在当前项目的根目录下的，里面可以编写当前项目特有的混淆规则。

接下来还剩一个dependencies闭包。这个闭包的功能非常强大，它可以指定当前项目所有的依赖关系。通常Android Studio项目一共有3种依赖方式：本地依赖、库依赖和远程依赖。本地依赖可以对本地的jar包或目录添加依赖关系，库依赖可以对项目中的库模块添加依赖关系，远程依赖则可以对jcenter仓库上的开源项目添加依赖关系。

观察一下dependencies闭包中的配置，第一行的implementation fileTree就是一个本地依赖声明，它表示将libs目录下所有.jar后缀的文件都添加到项目的构建路径中。而implementation则是远程依赖声明，androidx.appcompat:appcompat:1.2.0就是一个标准的远程依赖库格式，其中androidx.appcompat是域名部分，用于和其他公司的库做区分；appcompat是工程名部分，用于和同一个公司中不同的库工程做区分；1.2.0是版本号，用于和同一个库不同的版本做区分。加上这句声明后，Gradle在构建项目时会首先检查一下本地是否已经有这个库的缓存，如果没有的话则会自动联网下载，然后再添加到项目的构建路径中。至于库依赖声明这里没有用到，它的基本格式是implementation project后面加上要依赖的库的名称，比如有一个库模块的名字叫helper，那么添加这个库的依赖关系只需要加入implementation project(':helper')这句声明即可。

三、实验总结

（此处写你的过程，比如遇到的错误，以及解决方法，你的所想、所得）

由于是第一次接触安卓应用开发，在安装和调试Android Studio的过程中遇到了许多问题，例如版本不兼容，下载速度慢，同步速度慢，各种报错，模拟器启动失败等。接下来我将以此展示我遇到的问题的解决方法。

1. **安装AS汉化补丁**
   1. 从百度云链接下载汉化补丁
      1. 链接https://pan.baidu.com/share/init?surl=LwV8KVT09FkD-KNcBY4onA
      2. 密码：1234
   2. 不需要重命名，不需要解压，不需要删除任何 jar 包，不会覆盖任何 jar 包。软件安装路径的 lib 目录示例 D:\software\JetBrains\AndroidStudio\lib该目录下应该有一个文件: resources\_en.jar 如果没有，说明没有找对路径

个人体验：一开始因为对于AS的不熟悉，许多功能不了解，需要经常查询英文解释。但是因为汉化补丁不完整，仍有许多bug，最终选择删除并选择英文原版。

1. 修改Android Studio 中AVD安装路径的方法
2. **启动模拟器提示 unable to locate adb**
   1. 添加环境变量。在桌面右击“我的电脑”选择“属性”，进入“高级—-环境变量—–系统变量—-新建“，从而新建一个环境变量ANDROID\_SDK\_HOME，变量值设置为：D:\Android\_Studio。这个就是AVD的安装路径。一路确定下来，保存环境变量。重新启动计算机….
   2. 可能遇到的问题：启动设备后可能会出现install apps失败的情况。
      1. 解决方法：到file->settings菜单下去修改设置，取消红色区域的勾选，再启动AVD。
3. **Failed to resolve:com.android.support:appcompat-v7:报错处理**
   1. **主要原因是android studio SDK平台工具的版本太低。**
   2. 教程太长，故贴上CSDN论坛链接https://blog.csdn.net/mhl18820672087/article/details/78385361/
4. **同步Sync下载超级慢的问题**
   1. 添加阿里云镜像

build.gradle里的buildscript和allprojects添加阿里镜像

repositories {

maven{ url ‘http://maven.aliyun.com/nexus/content/groups/public/’}

}

重启AS之后同步速度就会加快。

以上是我遇到的一些问题以及解决方案。

在完成程序的解析过程中，确实遇到了很大的问题。源码很难去读懂，更多的是去查询了《第一行代码 Android》中的相关介绍，最终也是完成了一部分的代码以及结构的解析，为后一步的安卓应用开发打下基础。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：解为成  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。