

红岩网校移动开发部Android方向第五次课



步入Android世界的第一课

红岩网校移动开发部Android方向第五次课

步入Android世界的第一课

Android王国简介

什么是Android

一些关于Android开发的名词

开始启程——你的第一个APP

Android四大组件

UI

创建一个Android项目

运行你的第一个APP

用虚拟机进行调试

分析你的第一个APP

从看得见的入手——探究Activity

活动是啥？

手动创建一次activity

在activity之间跳转

Activity

生命周期

Activity的四种启动模式

Activity之间的通信

XML及Android常用控件

一些杂项

代码规范

写在最后

Android王国简介

什么是Android

以下来自百度百科：

安卓（Android）是一种基于Linux的自由及开放源代码的操作系统。主要使用于移动设备，如智能手机和平板电脑，由Google公司和开放手机联盟领导及开发。Android操作系统最初由Andy Rubin开发，主要支持手机。2005年8月由Google收购注资。2007年11月，Google与84家硬件制造商、软件开发商及电信营运商组建开放手机联盟共同研发改良Android系统。随后Google以Apache开源许可证的授权方式，发布了Android的源代码。第一部Android智能手机发布于2008年10月。Android逐渐扩展到平板电脑及其他领域上，如电视、数码相机、游戏机、智能手表等。2011年第一季度，Android在全球的市场份额首次超过塞班系统，跃居全球第一。2013年的第四季度，Android平台手机的全球市场份额已经达到78.1%。2013年09月24日谷歌开发的操作系统Android在迎来了5岁生日，全世界采用这款系统的设备数量已经达到10亿台。

一些关于Android开发的名词

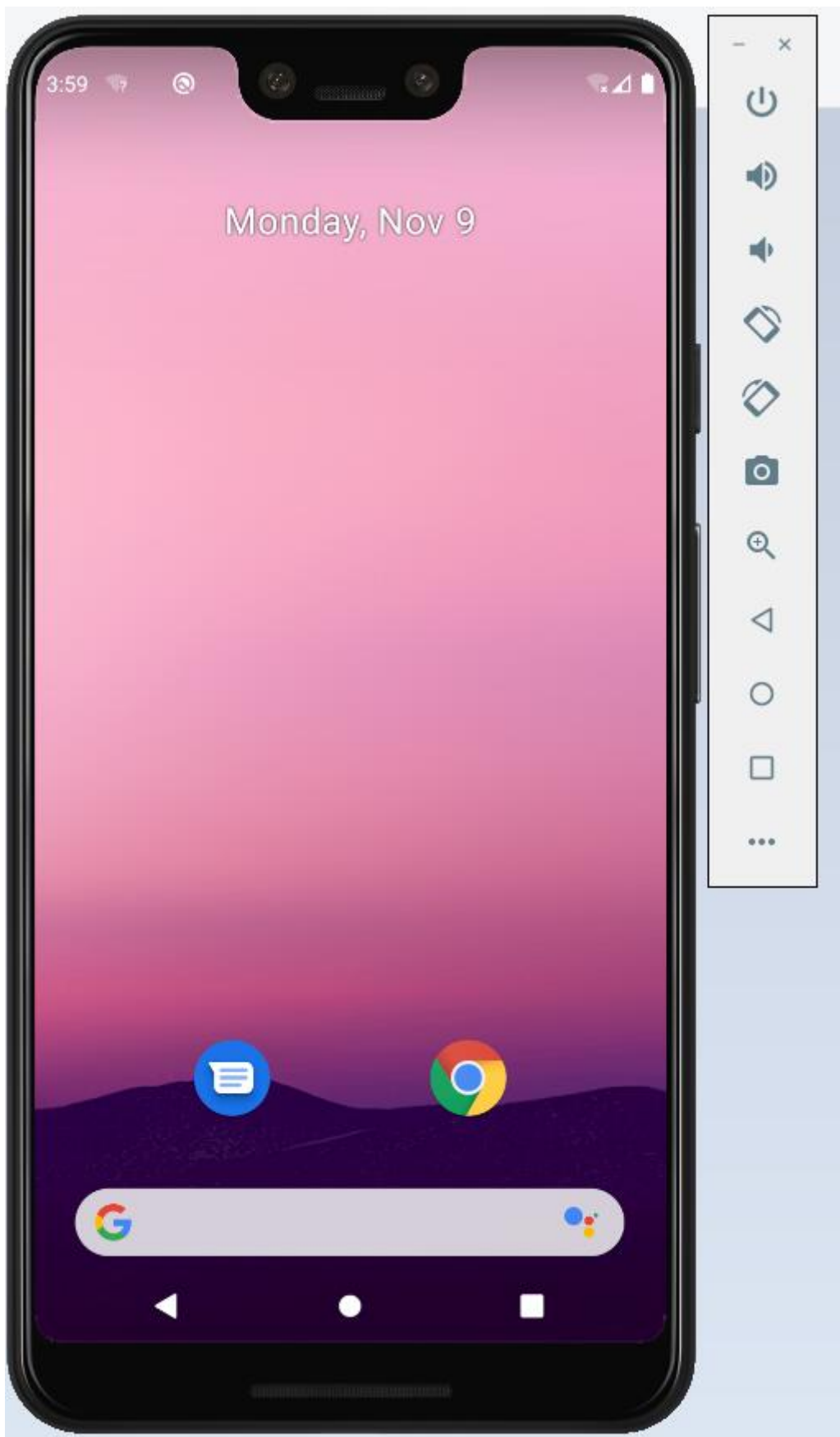
- Android SDK(Android software development kit) 软件开发工具包。

被软件开发工程师用于为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具的集合。**SDK主要是以Java语言为基础，用户可以使用Java语言来开发Android平台上的软件应用。**通过SDK提供的一些工具将其打包成Android平台使用的apk文件，然后用SDK中的模拟器（Emulator）来模拟和测试软件在Android平台上运行情况和效果。

- AVD(Android Virtual Device)

Android运行的**虚拟设备**，他是Android的模拟器识别。建立的Android要运行，必须创建AVD，每个AVD上可以配置很多的运行项目

在Android studio里，你甚至可以拥有一部**刘海屏手机**(太酷了)



- JDK(java development kit)

bin	java编译器、解析器	2017/8/14 10:52	文件夹	
db		2017/8/14 10:52	文件夹	
include		2017/8/14 10:52	文件夹	
jre	java运行环境	2017/8/14 10:52	文件夹	
lib	类库	2017/8/14 10:52	文件夹	
COPYRIGHT		2017/7/12 4:51	文件	4 KB
javafx-src.zip		2017/8/14 10:52	360压缩 ZIP 文件	4,979 KB
LICENSE		2017/8/14 10:52	文件	1 KB
README.html		2017/8/14 10:52	360 se HTML Do...	1 KB
release		2017/8/14 10:52	文件	1 KB
src.zip	源代码	2017/7/12 4:51	360压缩 ZIP 文件	20,758 KB
THIRDPARTYLICENSEREADME.txt		2017/8/14 10:52	文本文档	142 KB
THIRDPARTYLICENSEREADME-JAVAF...		2017/8/14 10:52	文本文档	63 KB

- JRE(Java Runtime Environment)

这是一个软件，由太阳微系统所研发，JRE可以让计算机系统运行Java应用程序 JRE内部有一个Java虚拟机(Java Virtual Machine, JVM),以及一些标准的类别函数库

开始启程——你的第一个APP

在敲出你的第一个APP前，我们需要了解一些Android开发的一些基础知识，了解这些知识将给畅游Android世界很大的帮助。

Android四大组件

- **Activity (本节课重点)**

一般一个Activity通常就是一个单独的屏幕，它上面可以显示一些控件，也可以监听并处理用户的事件做出响应。

掌邮整个登录界面就是一个Activity



整个地图也是个Activity (不会有人没用过地图吧不会吧不会吧) 烂梗扣钱



风华运动场

运动场

- Service

服务，一个Service 是一段长生命周期的，没有用户界面的程序，可以用来开发如监控类程序。

- Broadcast

广播接收器，我们的应用可以使用它对外部事件进行过滤只对感兴趣的外部事件(如当电话呼入时，或者数据网络可用时)进行接收并做出响应。广播接收器没有用户界面。然而，它们可以启动一个activity或服务来响应它们收到的信息，或者用NotificationManager 来通知用户。通知可以用很多种方式来吸引用户的注意力——闪烁背灯、震动、播放声音等。一般来说是在状态栏上放

一个持久的图标，用户可以打开它并获取消息。

- ContentProvider

内容提供者，其他应用可以通过ContentResolver类(见ContentProviderAccessApp例子)从该内容提供者中获取或存入数据。(相当于在应用外包了一层壳),只有需要在多个应用程序间共享数据是才需要内容提供者。例如，通讯录数据被多个应用程序使用，且必须存储在一个内容提供者中，它的好处是统一了数据访问方式。

UI

- 什么是UI

UI即User Interface(用户界面)的简称。UI设计是指对软件的人机交互、操作逻辑、界面美观的整体设计。好的UI设计不仅是让软件变得有个性有品位,还要让软件的操作变得舒适简单、自由,充分体现软件的定位和特点。软件设计可分为两个部分:编码设计与UI设计。UI的本意是用户界面,是英文User和Interface的缩写。从字面上看是用户与界面2个部分组成,但实际上还包括用户与界面之间的交互关系。(来自百度)

—(黑框框爪巴!)—



- 如何编写程序界面

Android Studio是给我们提供了可视化编辑器的，允许我们通过拖放控件的方式来编写布局，但是这种方式做出来的界面通常不具有很好的适配性(ps:也不是说不能有适配，只是相对来说比较麻烦，个人还是认为用代码编写布局的更加舒服)，我们还可以通过**xml代码**的方式来编写布局界面（主要方式）。

这时候，正在认真听课的果冻冻私聊我说：



xml是什么？想摸了？



看来果冻冻同学听课很认真，不错不错



那么下面给大家解释下什么是XML

- XML

可扩展标记语言，标准通用标记语言的子集，简称XML。是一种用于标记电子文件使其具有结构性的标记语言。在XML文件中设计UI可以更好地将应用的外观与控制应用行为的代码隔离，每次修改或调整界面布局只需要修改XML文件的代码而不是修改源码和重新编译。

也就是说，我们想要跳出黑框框，编写丰富多彩的界面就需要使用XML。

XML的简单使其易于在任何应用程序中读写数据，这使XML很快成为数据交换的唯一公共语言，虽然不同的应用软件也支持其它的数据交换格式，但不久之后他们都将支持XML，那就意味着程序可以更容易的与Windows、Mac OS, Linux以及其他平台下产生的信息结合，然后可以很容易加载XML数据到程序中并分析他，并以XML格式输出结果。

XML比较简单，易于理解，大家不用过于担心XML的问题。

好了，基础知识讲了挺久，大家都听困了吧



那么现在我们就上号吧！



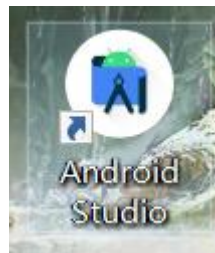
不好意思，放错图了



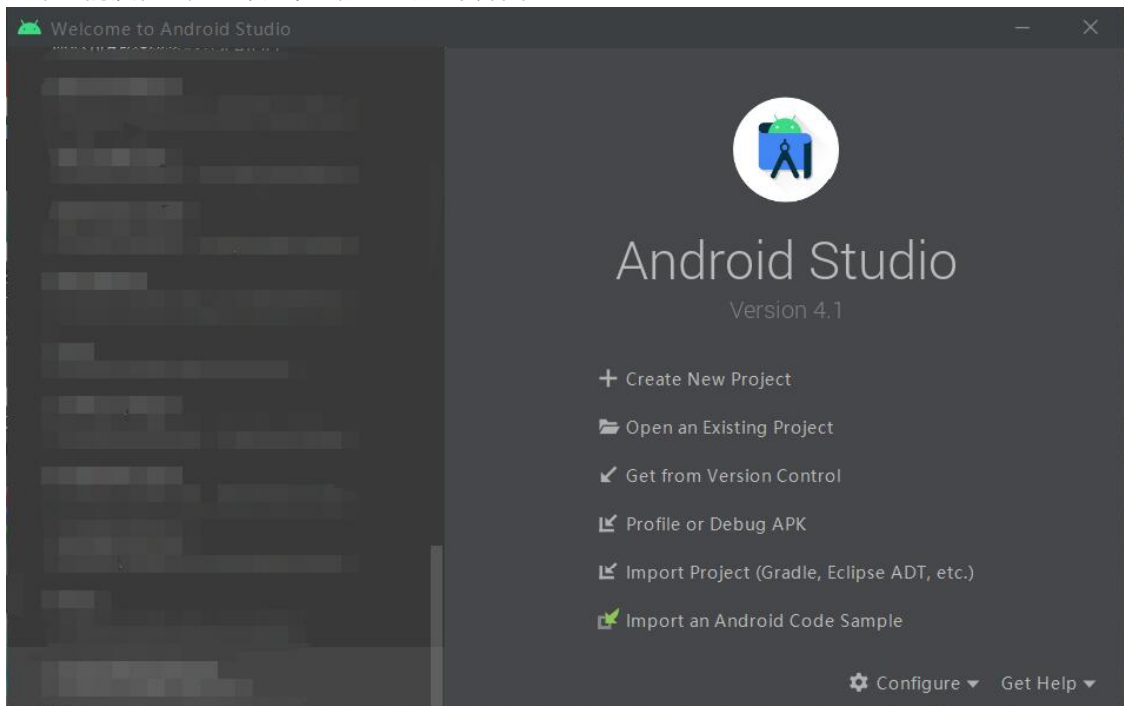
创建一个Android项目

保姆级手把手教学开始

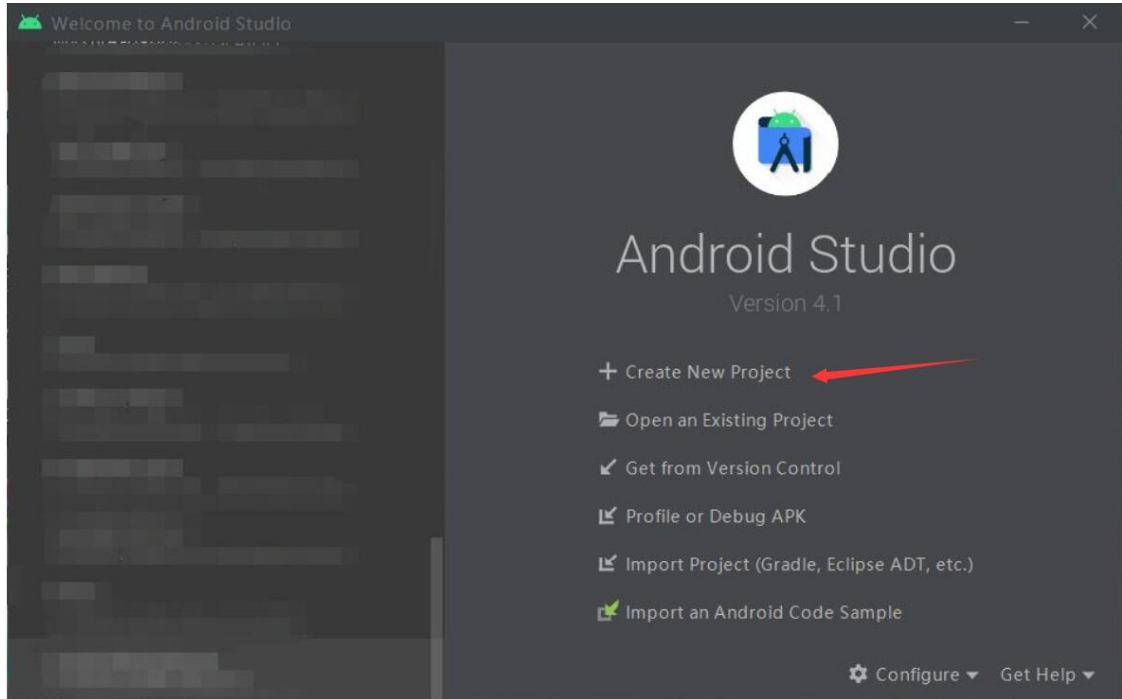
1. 双击可爱的Android studio图标



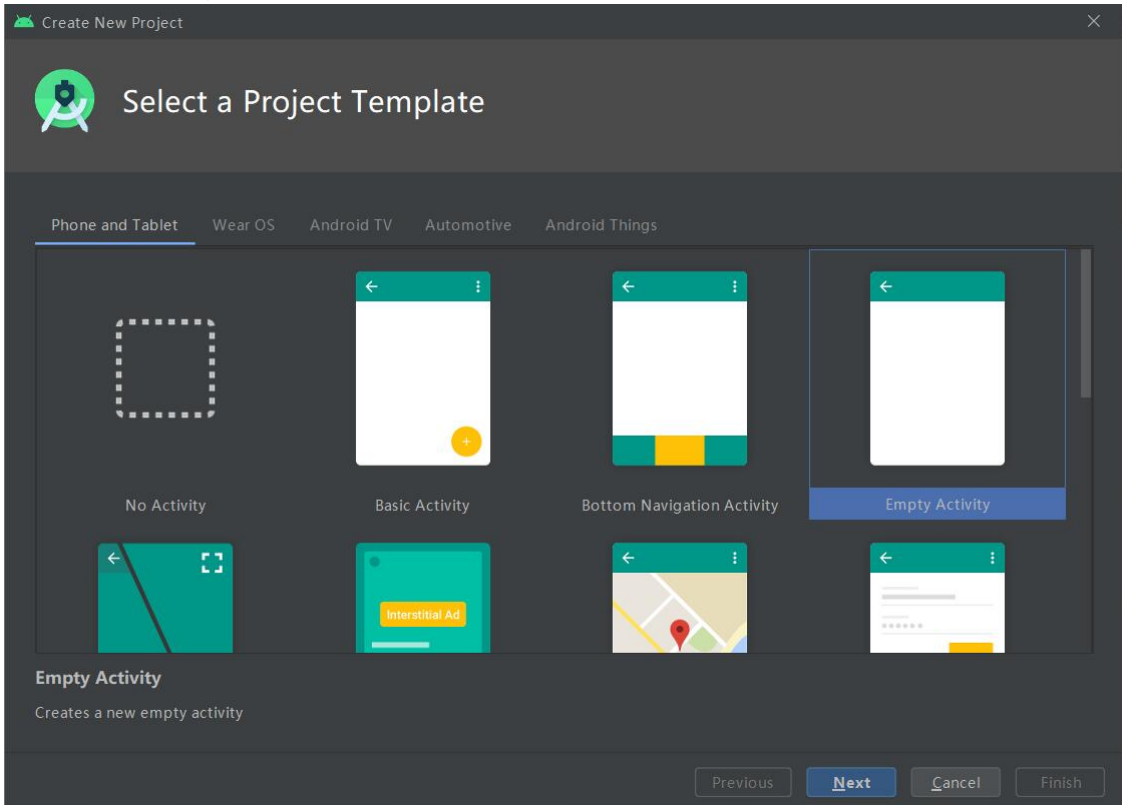
2. 如果之前没有创建过项目，会进入到这个界面



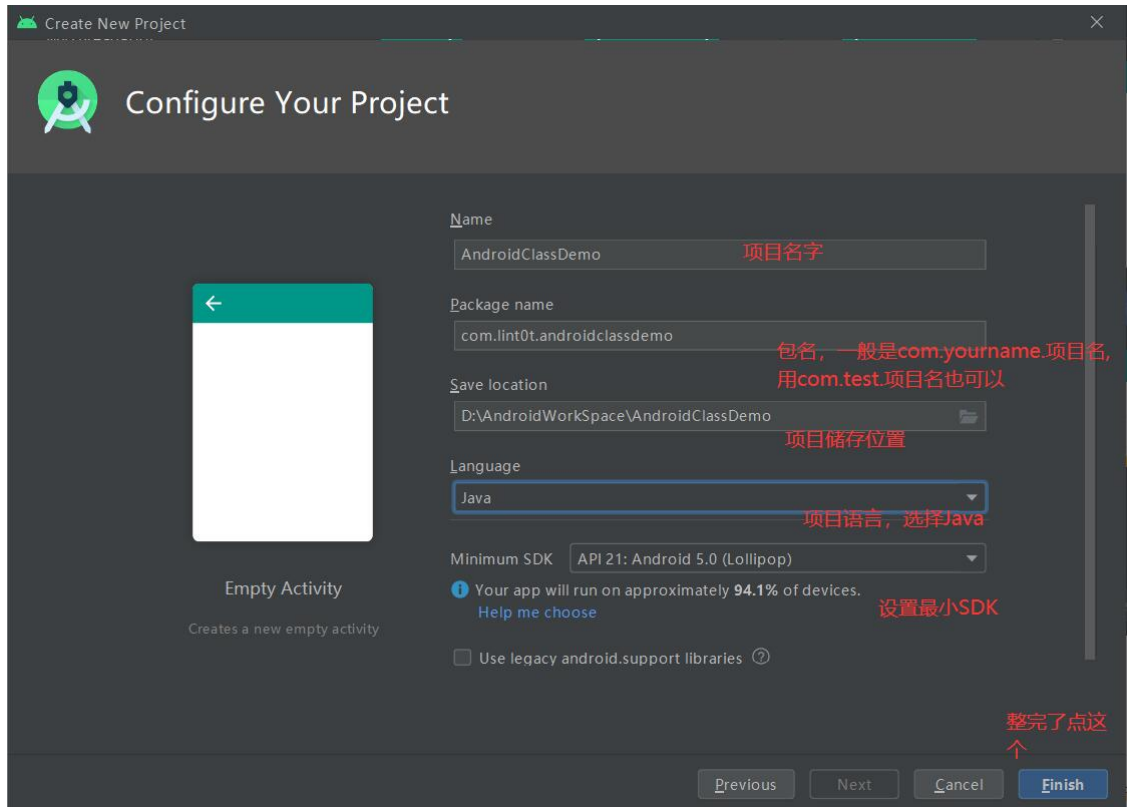
3. 然后点击这个



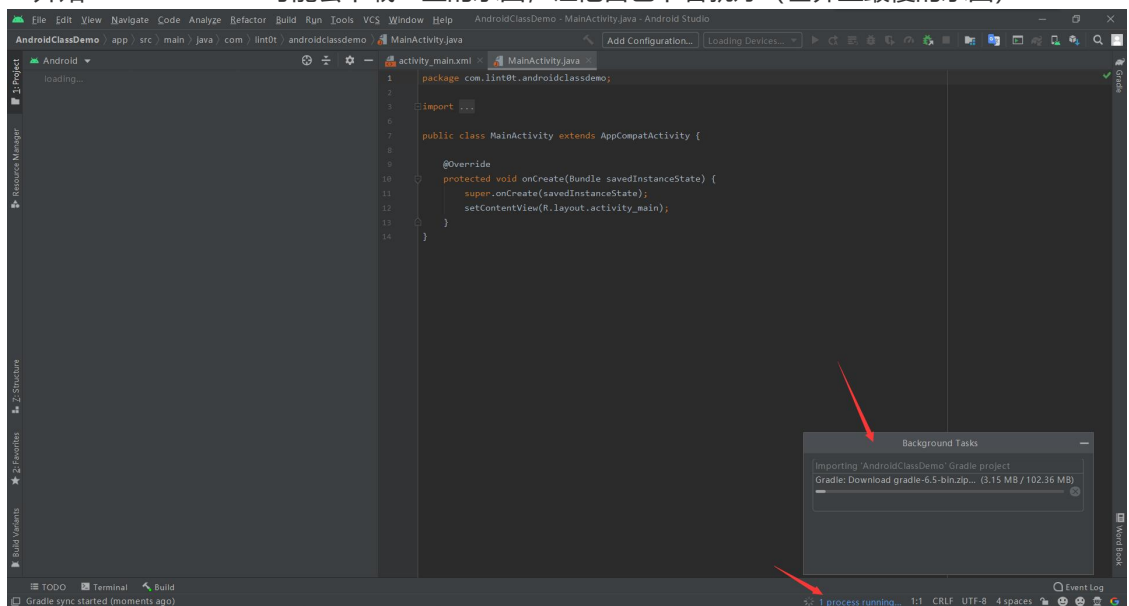
4. 接着选择空activity, 点击next



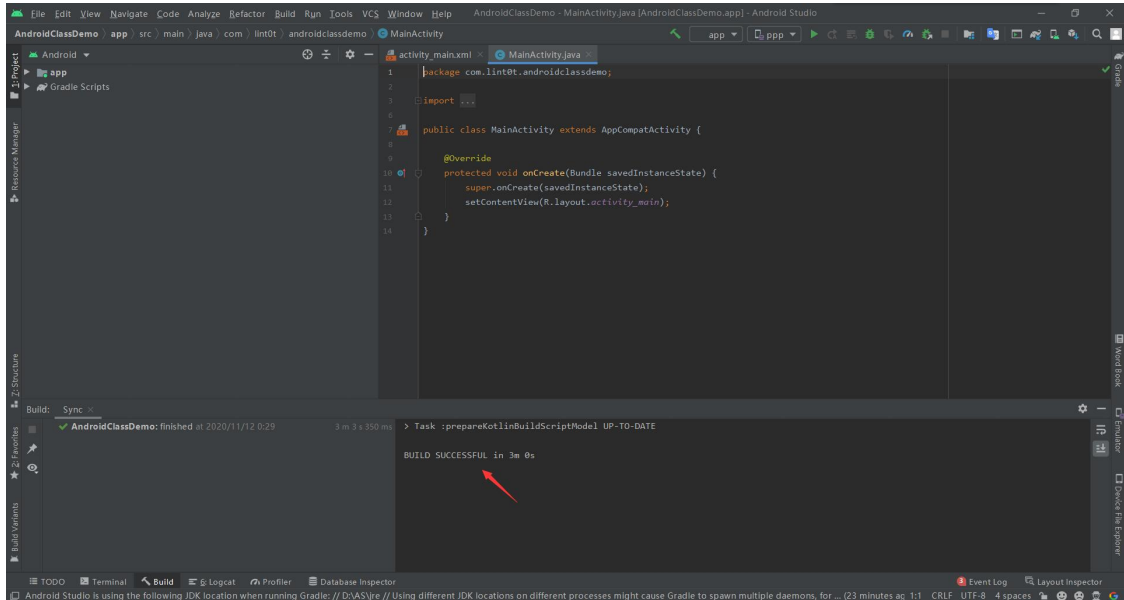
5. 下一步，设置项目详细内容



6. 一开始Android studio可能会下载一些的东西，让他自己下着就好（世界上最慢的东西）



7. 完成后会有提示，这样一个新项目就创建完成了！芜湖~



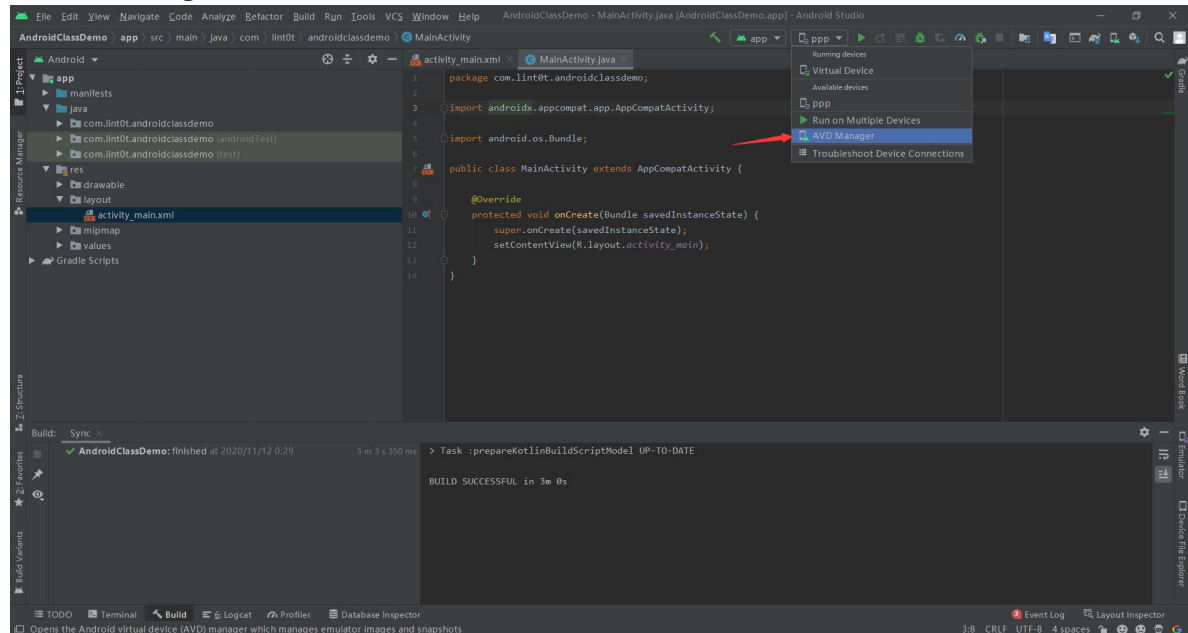
运行你的第一个APP

当我们创建好一个新项目后，可以看见Android studio已经默认帮我们写了一些代码，我们可以运行一下看看结果，当然，Android项目要跑在Android手机上，所以我们需要有个手机，可以使用真机调试，也可以使用虚拟机调试。由于真机每个牌子的手机设置有一定区别，所以我们这里只讲虚拟机调试。想使用真机调试的同学可以百度：Android studio XX手机真机调试，一般是打开开发者模式，开启USB调试，用数据线连接到电脑即可。

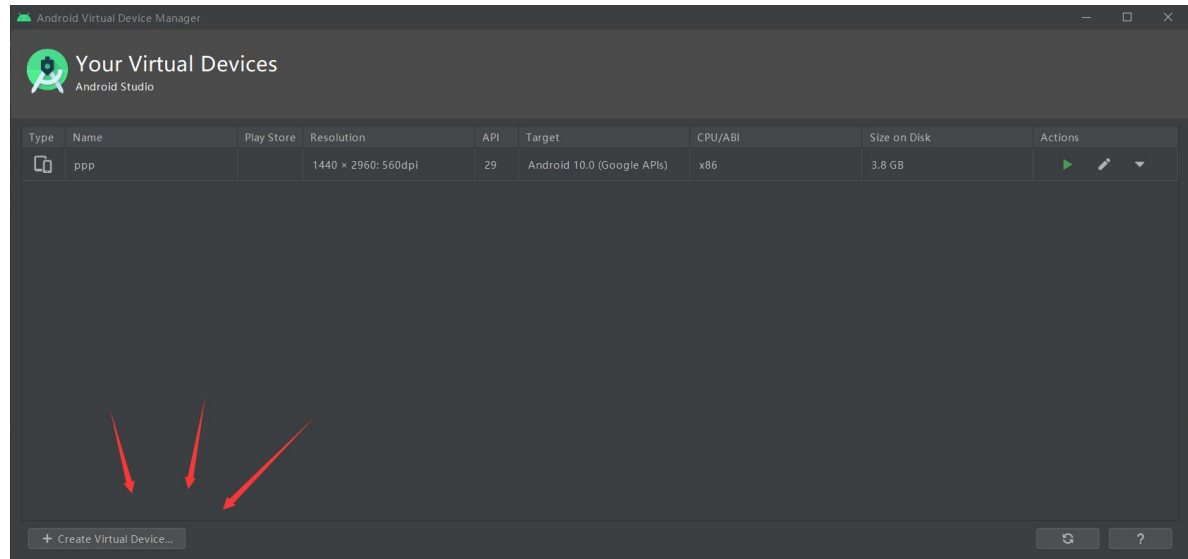
用虚拟机进行调试

如果在安装时勾选了虚拟机选项（详情参考群文件里御姐的安装教程），我们就可以在Android studio里创建虚拟机。

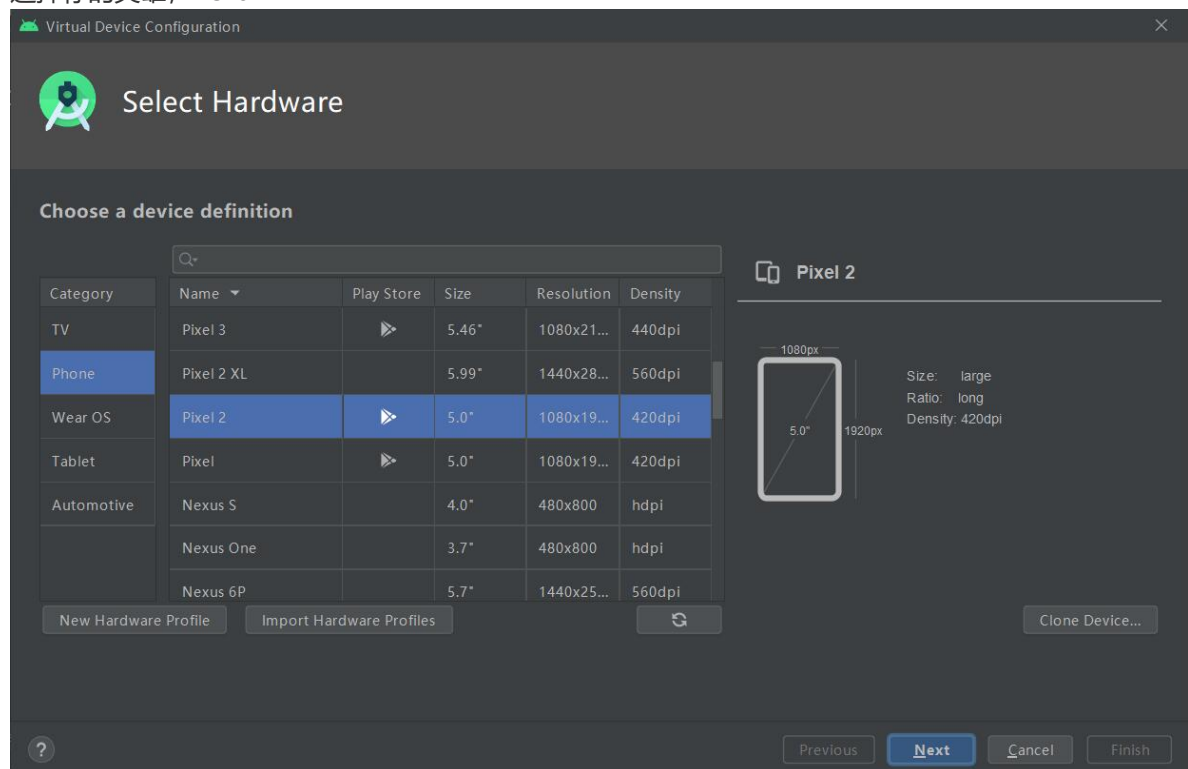
点击avd manager



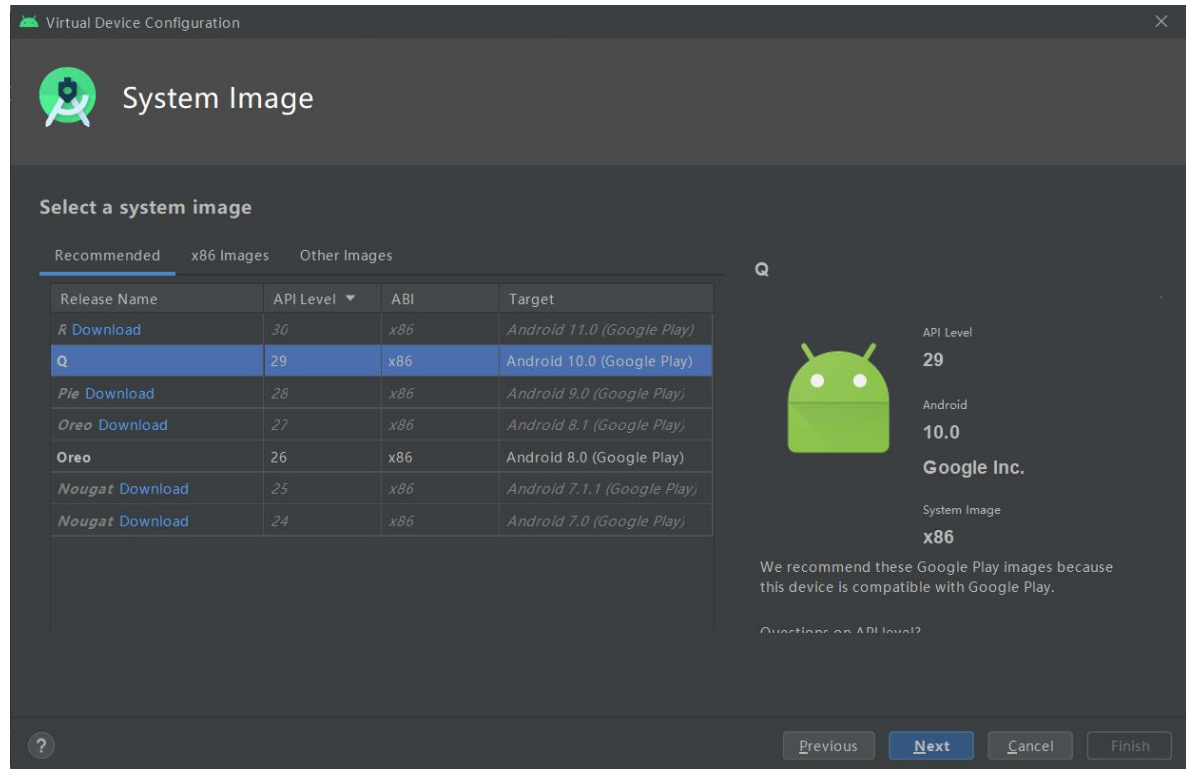
在弹出来的界面选择创建新虚拟机



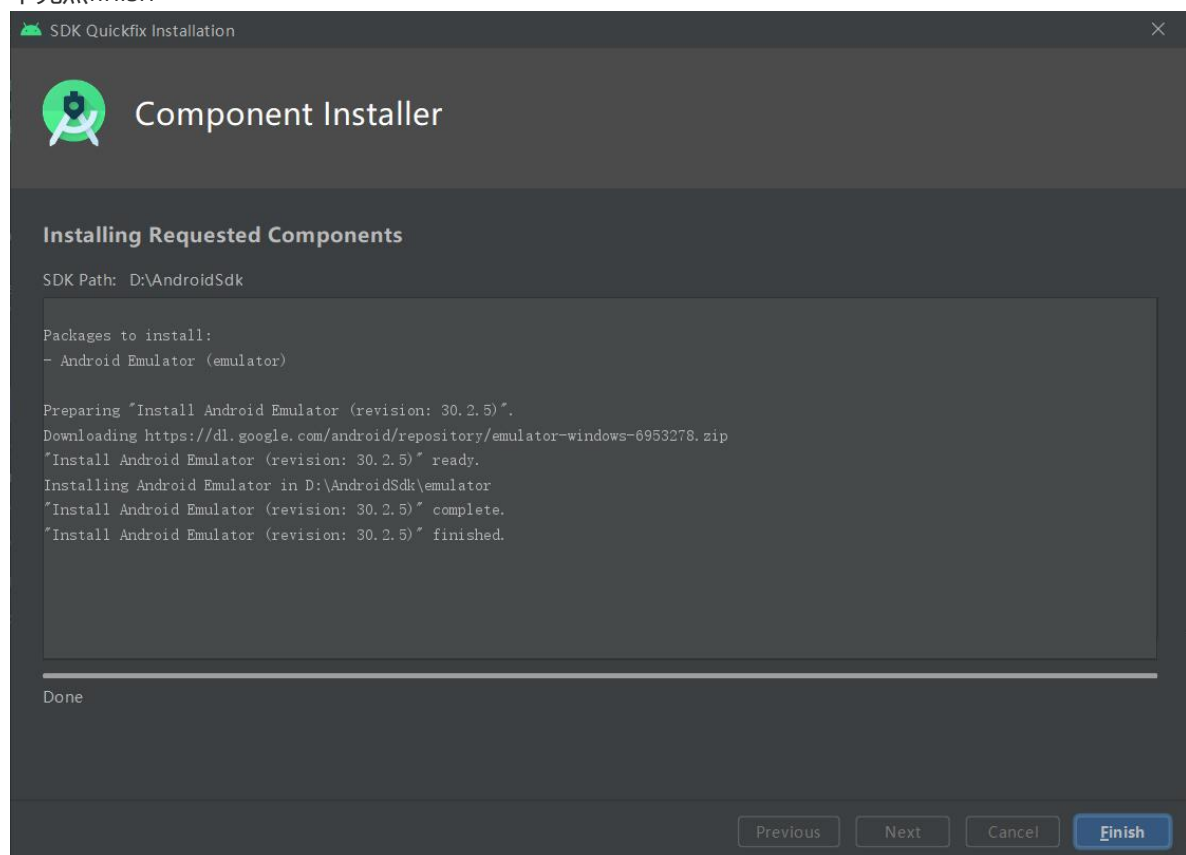
选择你的英雄，next



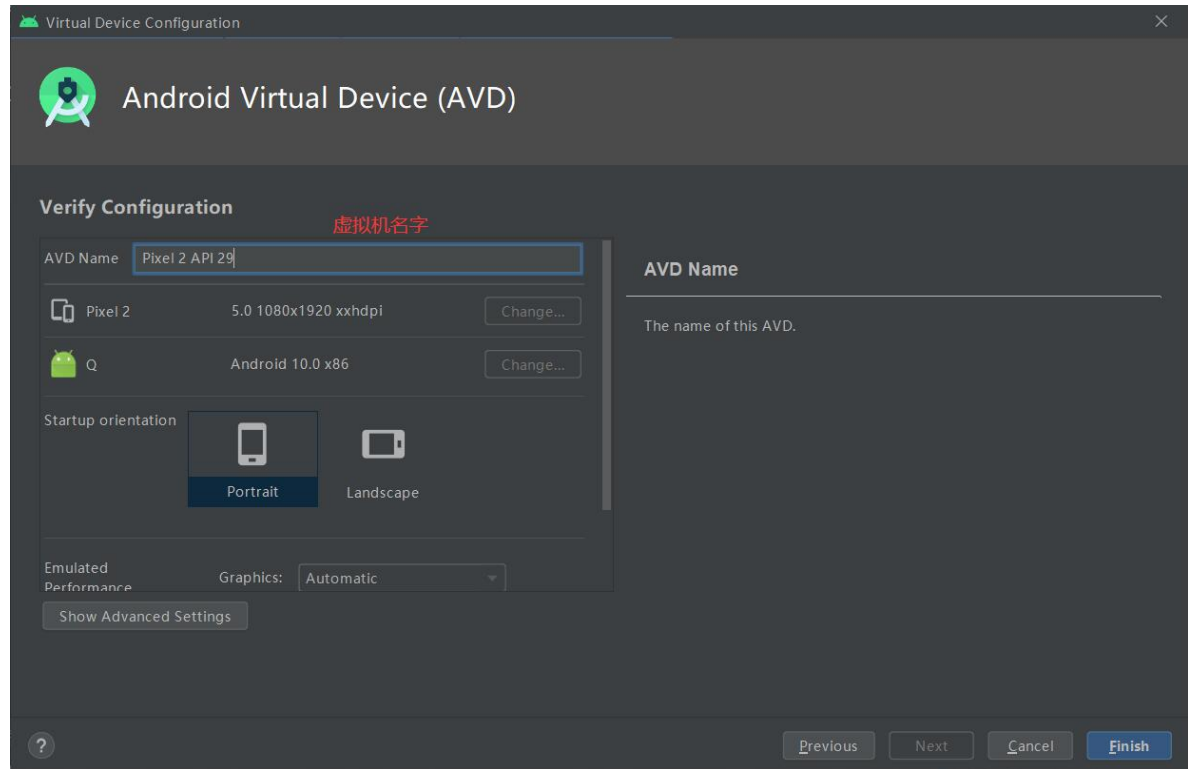
选择你的系统，这里主要关注一下API level和Android版本,如果没有下载的话要点击download下载



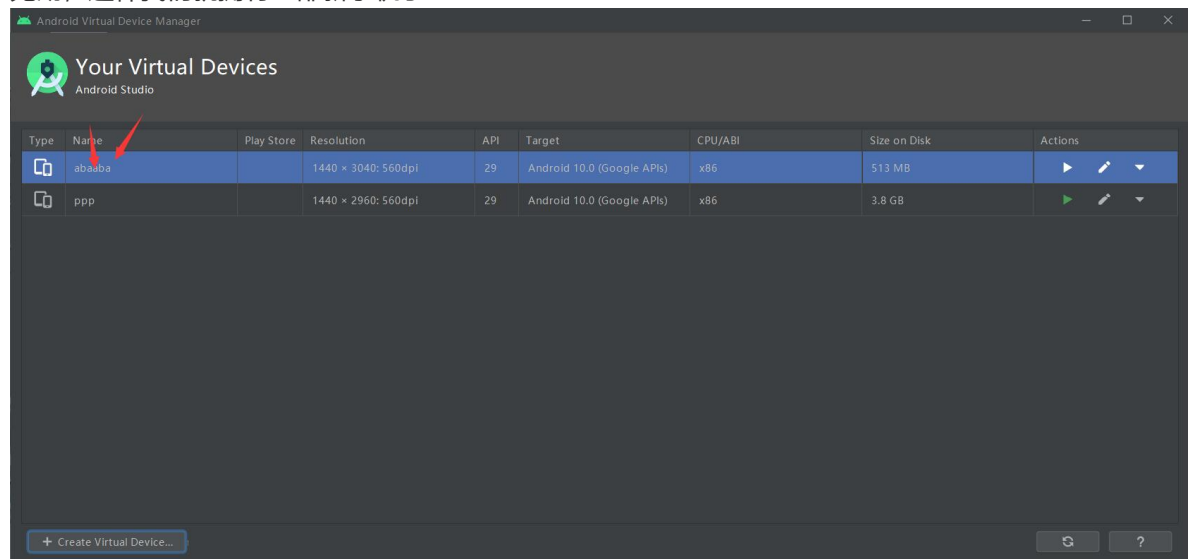
下完点finish



进行详细设置，其实也就改个名字，其他默认就行，设置完点finish

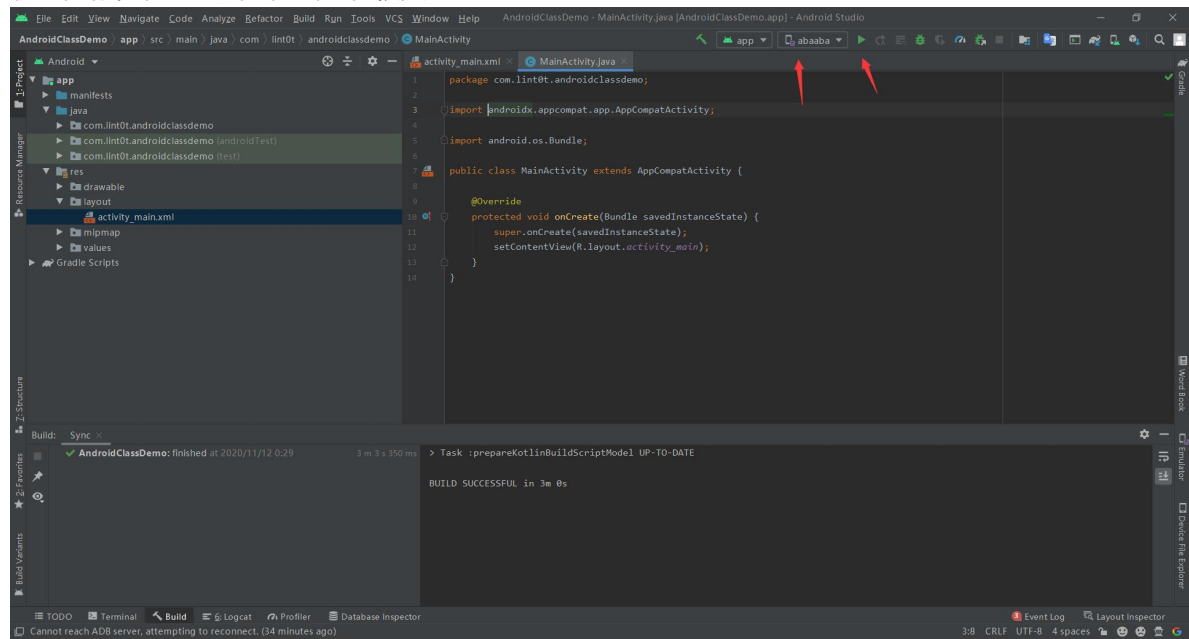


芜湖，这样我们就拥有一部新手机了！



起飞

现在回到Android studio里，运行项目吧，选择刚才创建的虚拟机，点击旁边绿色箭头运行，虚拟机比较吃内存，有些电脑可能会运行很久

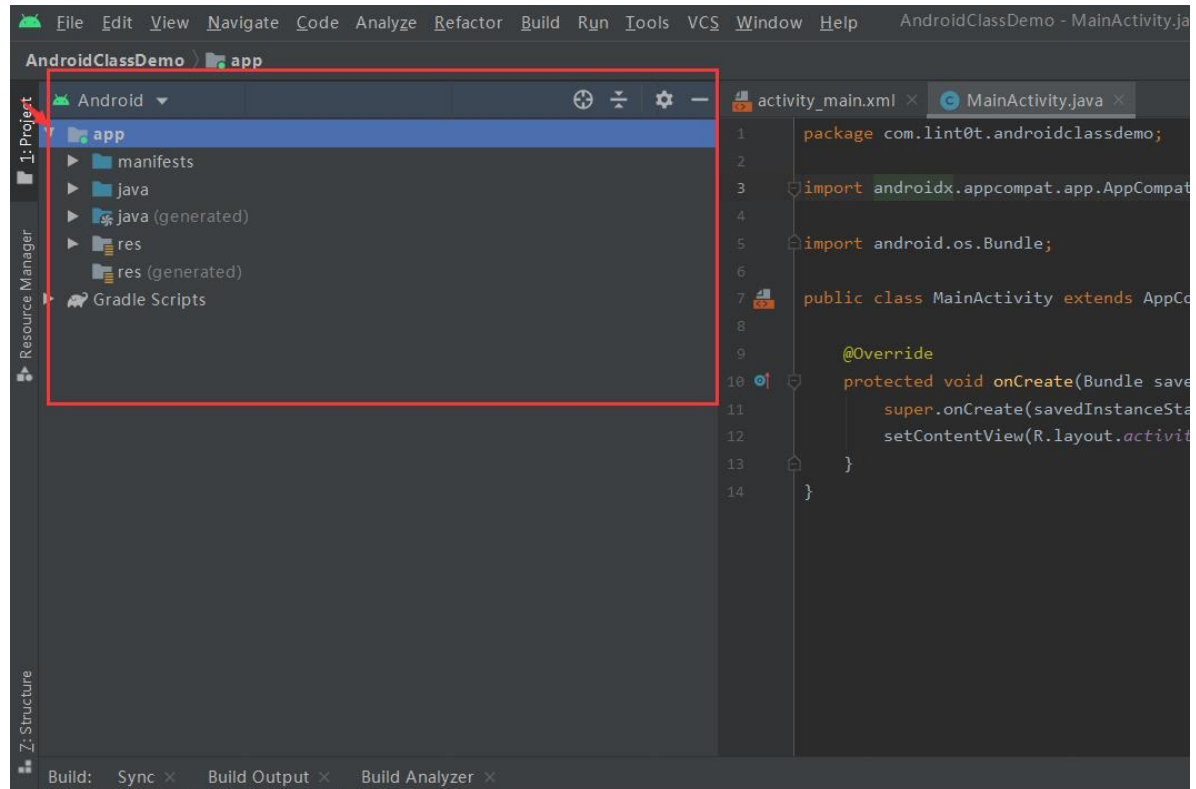


好耶！我们的第一个APP跑起来了，Android studio自动帮我们写了一个Hello World

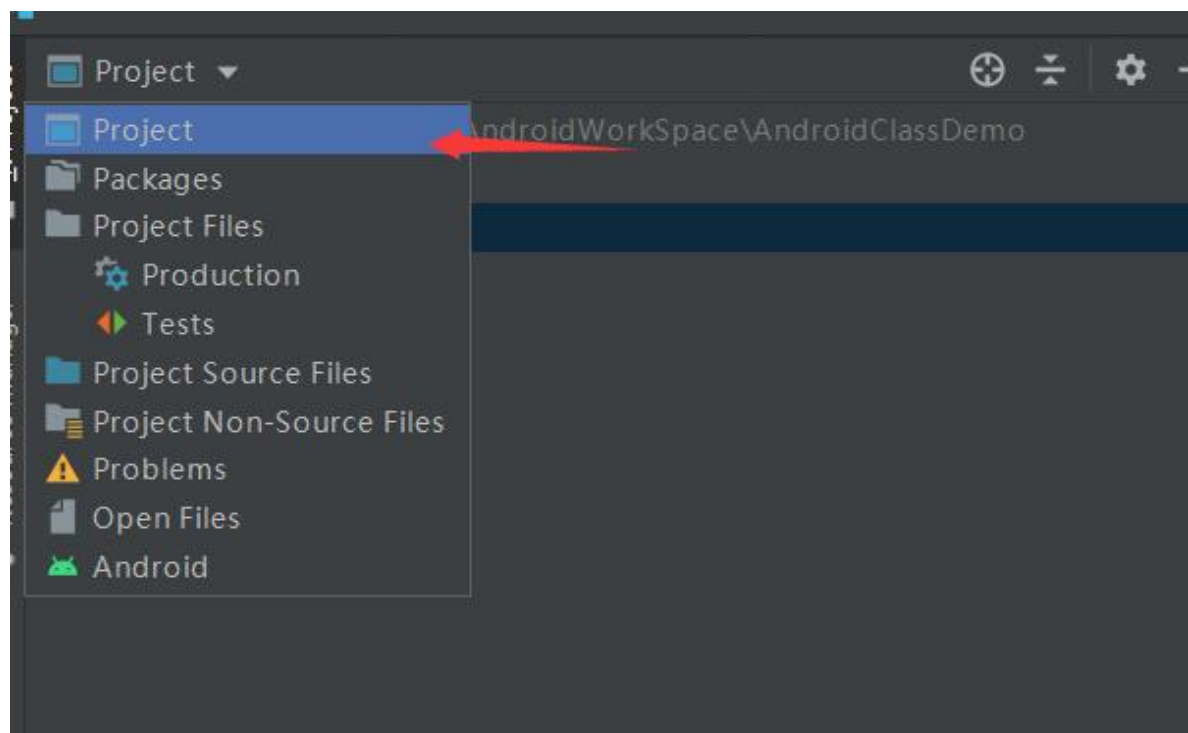


分析你的第一个APP

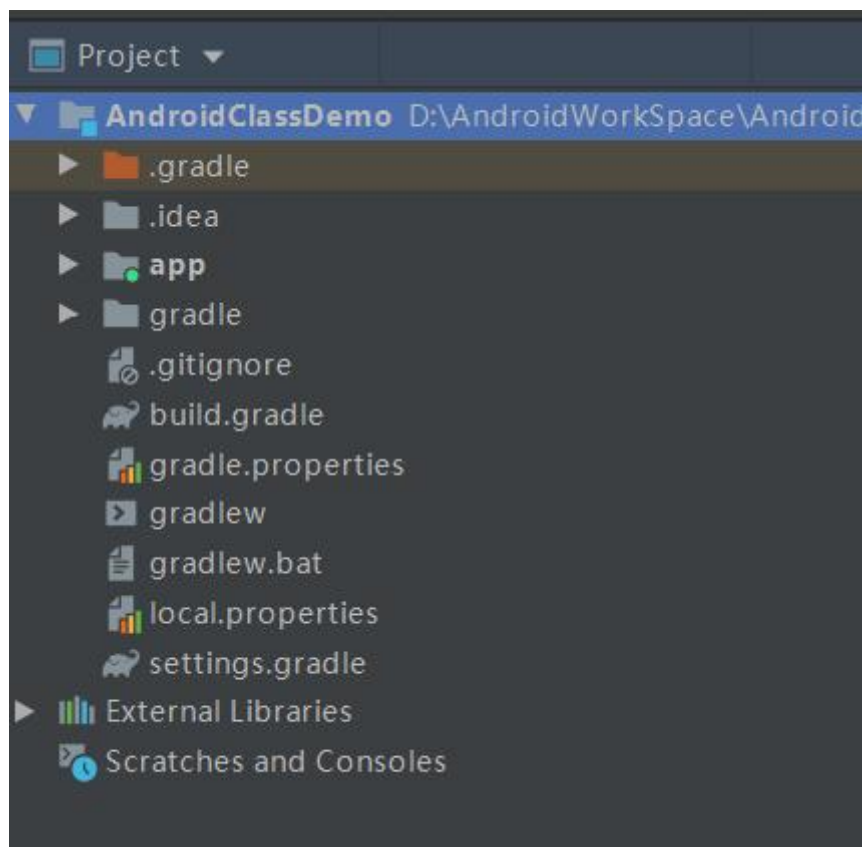
回到Android studio, 点击左侧展开项目结构



任何一个新建的项目都会默认使用这种Android模式的项目结构, 但这并不是项目真实的目录结构, 而是被Android Studio转换过的。这种项目结构简洁明了, 适合进行快速开发, 但是对于新手来说可能并不易于理解。点击Android区域可以切换项目结构模式为Project



切换后是这样的



这才是我们项目的真实结构，一开始看到这么多东西，一定会很头晕吧，没事，我从第一行代码中cv了各个内容的解释，有书的同学可以看书。

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，我们无须关心，也不要手动编辑。

2. **app**

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

3. build

这个目录你也不需要过多关心，它主要包含了一些在编译时自动生成的文件。

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要联网下载gradle。Android Studio默认没有启用gradle wrapper的方式,如果需要打开,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,关于版本控制我们将在第5章中开始正式的学习。

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。稍后我们将会详细分析gradle构建脚本中的具体内容。

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,通常内容都是自动生成的,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项目中就只有一个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都是自动完成的,需要我们手动去修改这个文件的场景可能比较少。

12. AndroidManifest.xml

这是你整个Android项目的配置文件,你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。由于这个文件以后会经常用到,我们用的时候再做详细说明。

现在整个项目的外层目录结构已经介绍完了。你会发现,除了app目录之外,大多数的文件和目录都是自动生成的,我们并不需要进行修改。也就是说,app目录下的内容才是我们以后的工作重点。篇幅有限,app下内容的详解请参看《第一行代码》。

从看得见的入手——探究Activity

通过之前的学习,你已经成功创建并运行了你的第一个项目,不过仅仅满足于此是不够的,是时候学点新东西了。

活动是啥?

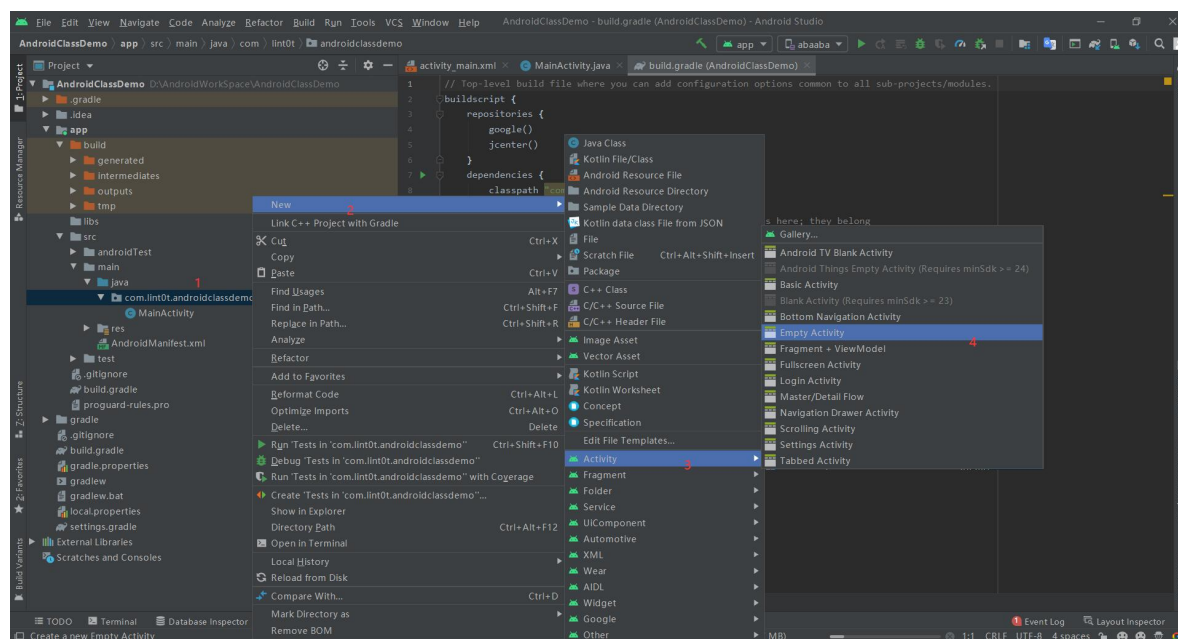
活动(Activity)是最容易吸引用户的地方,它是一种可以包含用户界面的组件,主要用于和用户进行交互。一个应用程序中可以包含零个或多个活动,但不包含任何活动的应用程序很少见,谁也不想让自己的应用永远无法被用户看到吧?

其实在前面,你已经和活动打过交道了。不过前面我们的重点是创建你的第一个Android项目,对活动的介绍并不多,在本章中我将对活动进行详细的介绍。

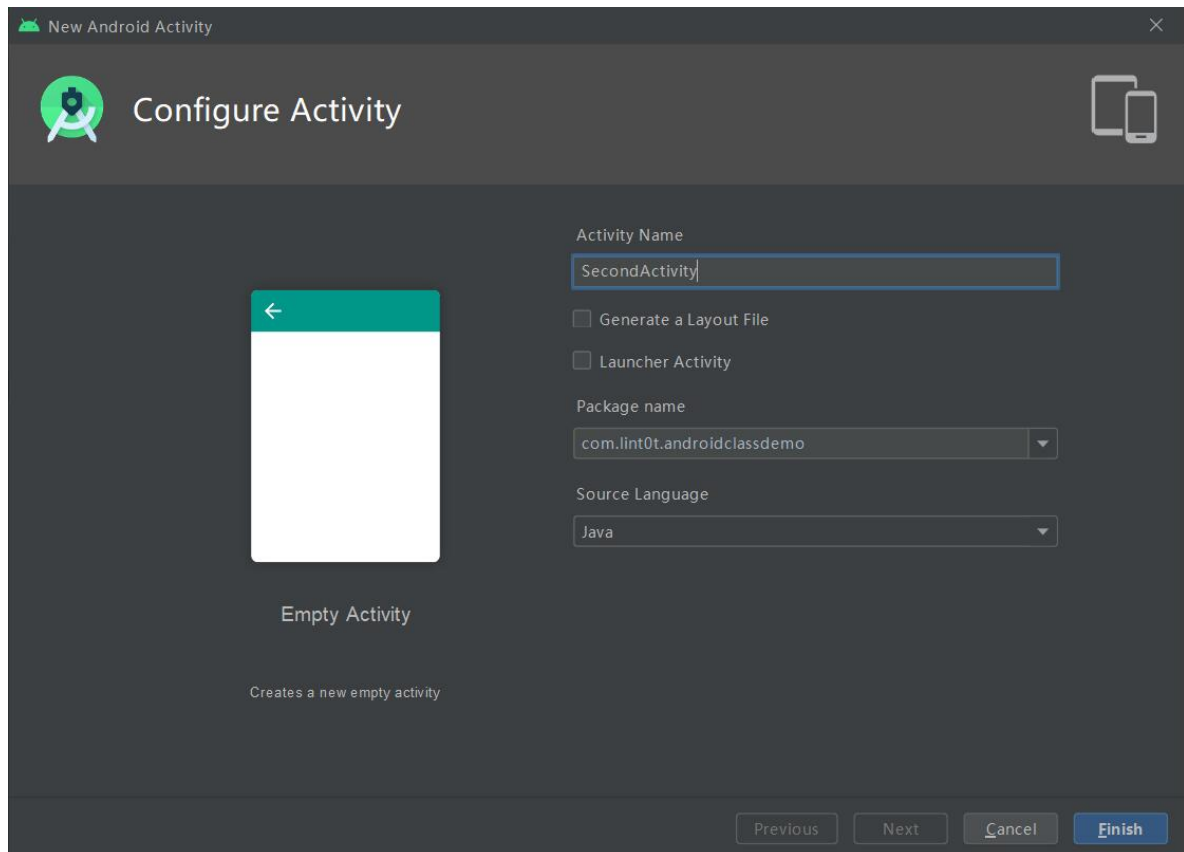
手动创建一次activity

为了加深大家对activity的理解,让我们来手动创建一次activity吧。

依次展开src→main→java→com.test.项目名,右键这个文件夹,选择new→activity→empty activity



将此activity命名为SecondActivity，勾选Generate Layout File表示会自动为FirstActivity创建一个对应的布局文件,勾选LauncherActivity表示会自动将SecondActivity 设置为当前项目的主活动，这里由于你是第一次手动创建活动，这些自动生成的东西暂时都不要勾选，下面我们将会一个个手动来完成。点击Finish完成创建。



观察一下我们SecondActivity的代码，再和MainActivity对比一下

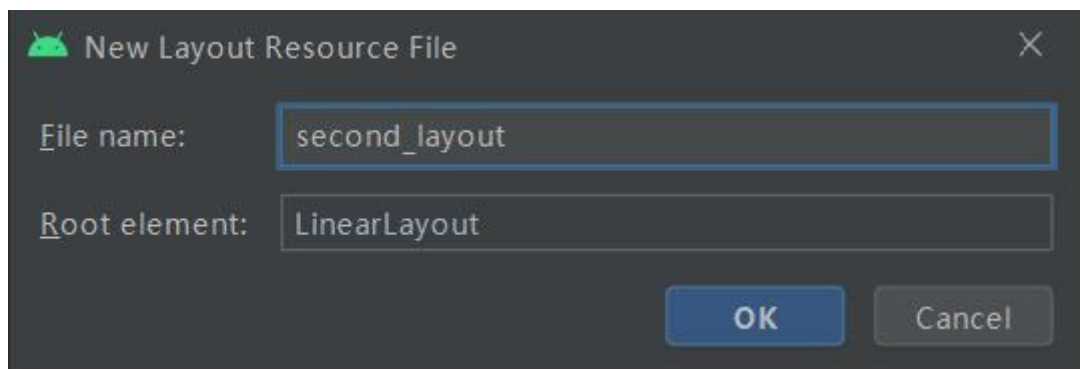
```
public class SecondActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

我们会发现，代码中少了一句

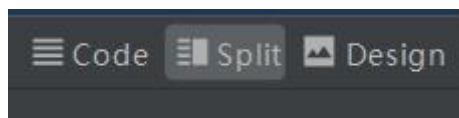
```
setContentView(R.layout.activity_main);
```

这行代码是什么意思呢？前面我们说过，Android程序的设计讲究逻辑和视图分离，每一个活动最好都能对应一个布局，布局就是用来显示界面内容的，这行代码的作用就是给这个activity设置一个布局。因此我们现在就来手动创建一个布局文件。

右击 app/src/main/res/layout→New→Layout Resource File，会弹出一个新建布局资源文件的窗口，我们将这个布局文件命名为second_layout，根元素先选择为LinearLayout。点击OK。



一开始会自动选择为Design界面，这是Android studio提供的一个可视化编辑器，我们不推荐使用可视化编辑器进行编写，我们可以点击右上角Split切换到代码+预览界面，或者点击Code切换到代码界面，我一般喜欢使用Split界面。

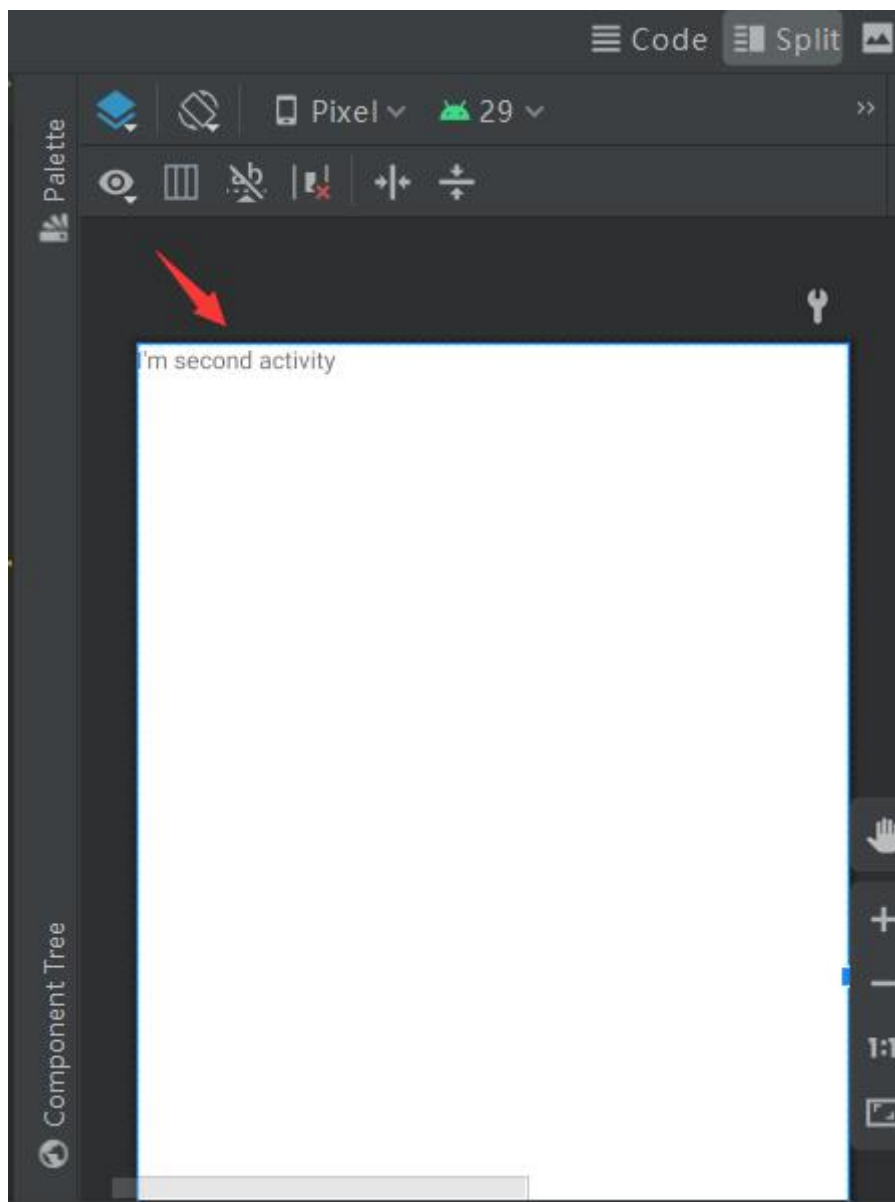


我们先在这个界面加上一个TextView控件（后面会讲），显示内容为"I'm second activity"，id为tv_test。可以在右侧看见预览。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/tv_test"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="I'm second activity"
    />

</LinearLayout>
```



文字不居中看起来有点难受，我们在TextView里加上一句

```
android:gravity="center"
```

这行代码可以让控件内的内容居中显示。现在看看预览窗口，就会发现文字已经居中了。

我想大家已经猜到下一步要干嘛了，我们拥有了一个布局，一个没有布局的activity，下一步自然是把他们两个贴在一起（activity和布局都在一起了，你呢）。现在我们回到SecondActivity，模仿MainActivity，在super.onCreate下面添加上

```
setContentView(R.layout.second_layout);
```

这里我们通过R.layout.second_layout告诉activity我们要把这个布局给他。项目中添加的任何资源都会在R文件中生成一个相应的资源id，因此我们刚才创建的second_layout.xml布局的id现在应该是已经添加到R文件中了。在代码中去引用布局文件的方法只需要调用R.layout.second_layout就可以得到second_layout.xml布局的id，然后将这个值传入setContentView()方法即可。但我们要注意到的一点是，所有的activity都必须在AndroidManifest里注册，才可以生效我们打开AndroidManifest看看，好家伙

```
<activity android:name=".SecondActivity"></activity>
```

Android studio已经自动帮我们注册好了。在标签中我们使用了android:name来指定具体注册哪一个活动，那么这里填入的.FirstActivity是什么意思呢?其实这不过就是com.test.activityclassdemo.SecondActivity的缩写而已。由于在最外层的标签中已经通过 package属性指定了程序的包名是com.test.activityclassdemo，因此在注册活动时这一部分就可以省略了，直接使用.SecondActivity就足够了。

我们观察一下MainActivity，发现它比SecondActivity多了这些东西

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

这些是啥呢，通过这些代码，我们可以指定MainActivity为启动时的主Activity。

android.intent.action.MAIN：决定应用的入口Activity，也就是我们启动应用时首先显示哪一个Activity。android.intent.category.LAUNCHER：表示activity应该被列入系统的启动器(launcher)(允许用户启动它)。Launcher是安卓系统中的桌面启动器，是桌面UI的统称。

到这一步，我们手动创建activity就结束了，在平时创建activity时，我们选择默认参数的话Android studio会自动帮我们进行到这里。

好了，我们运行一下试试。

啊这，我们只能看见MainActivity，看不见SecondActivity啊，咋办？这里我们就要用到Intent了。

在activity之间跳转

为了方便展示，我们先在MainActivity中加一个按钮。还记得怎么编写界面吗？打开我们的activity_main.xml，添加下面的代码

```
<Button
    android:id="@+id/btn_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="跳转到第二个activity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />
```

这里的Button是按钮控件，给他id为btn_test,文字为“跳转到第二个activity”。其他代码后面会说到。看下预览窗口，这里就有了一个Button按钮。现在运行一下试试

我们发现按钮可以按了，但并没有切换到第二个activity。这是肯定的，因为我们并未对这个按钮的点击事件进行监听，也就是说，在你按下这个按钮后，我们的app并不知道我们想要干嘛。所以我们需要监听了这个按钮的点击事件，告诉app在按下按钮时我们要切换到第二个activity。

在MainActivity中添加如下代码

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    /*
        通过findViewById，在activity中获取到我们在布局中设置的button控件,id是我们自己起的
        btn_test
    */
```

```

        Button mButton = findViewById(R.id.btn_test);
        /*
        给我们获取到的mButton添加点击事件监听
        */
        mButton.setOnClickListener(new View.OnClickListener() {
            //还记得匿名内部类吗
            @Override
            public void onClick(View v) {
                /*
                显式Intent，通过Intent可以设置要跳转到的activity，这样程序就可以知道你的“意
                图”。
                */
                Intent intent = new Intent(MainActivity.this, SecondActivity.class);
                startActivity(intent);
            }
        });
    }
}

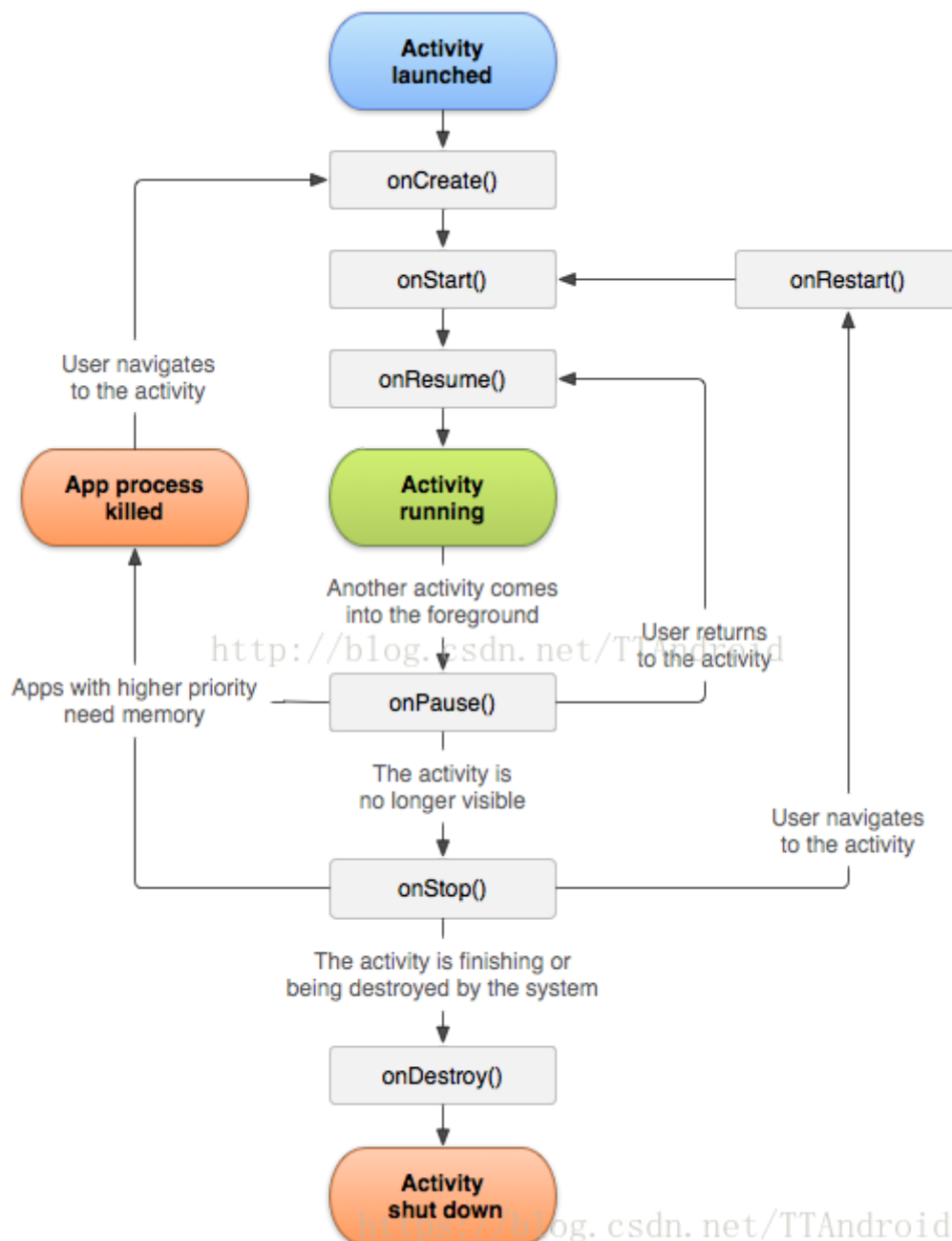
```

在运行看看，点击按钮，我们成功的跳转到了第二个activity!

Activity

下面我们讲解下activity的有关知识。

生命周期

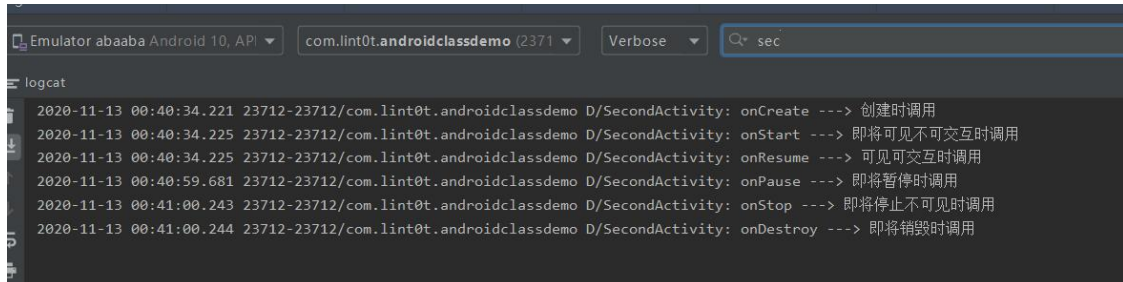


常规 onCreate->onStart->onResume->onPause->onStop->onDestroy

1. onCreate: 在首次创建 Activity 时调用。系统向此方法传递一个 Bundle 对象，其中包含 Activity 的上一状态，不过前提是捕获了该状态，而后会调用onStart方法。（可以在此方法中执行所有正常的静态设置，比如：创建视图、将数据绑定到列表等等。）
2. onStart: 在 Activity 即将对用户可见之前调用。而后如果Activity转入了前台就会调用onResume方法。如果此时直接屏幕熄灭或者用户按下home键则会直接调用onStop方法，当然这种情况比较极端。
3. onResume: 在 Activity 即将开始与用户进行交互之前调用。此时，Activity 处于 Activity 堆栈的顶层，并具有用户输入焦点。当跳转另一个Activity，或者退出当前Activity后会调用onPause方法。
4. onPause: 在系统即将开始继续另一个 Activity 时调用。此方法通常用于确认对持久性数据的未保存更改、停止动画以及其他可能消耗 CPU 的内容，诸如此类。它应该非常迅速地执行所需操作，因为它返回后，下一个 Activity 才能继续执行,所以不能执行耗时操作。而后正常情况下会调用onStop方法。但是有一种极端情况，就是如果这个时候快速让 当前Activity 返回前台，则会调用onResume方法。

5. onStop: 在 Activity 对用户不再可见时调用。如果 Activity 被销毁, 或另一个 Activity (一个现有 Activity 或新 Activity) 继续执行并将其覆盖, 就会调用此方法。而后如果 Activity 恢复与用户的交互, 则会调用 `onRestart` 方法, 如果 Activity 被销毁, 则会调用 `onDestroy` 方法。
6. onRestart: 在 Activity 被停止后再次启动时调用 (即屏幕熄灭后再次回到 app, 按下 home 键后再次回到 app), 而后会调用 `onStart` 方法。
7. onDestroy: 在 Activity 被销毁前调用, 这是 Activity 收到的最后调用。当 Activity 结束 (对 Activity 调用了 `finish` 方法), 或系统为节省空间而暂时销毁该 Activity 实例时, 可能会调用它。你可以通过 `isFinishing` 方法区分这两种情形。

通过代码测试下:



```
2020-11-13 00:40:34.221 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onCreate ---> 创建时调用
2020-11-13 00:40:34.225 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onStart ---> 即将可见不可交互时调用
2020-11-13 00:40:34.225 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onResume ---> 可见可交互时调用
2020-11-13 00:40:59.681 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onPause ---> 即将暂停时调用
2020-11-13 00:41:00.243 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onStop ---> 即将停止不可见时调用
2020-11-13 00:41:00.244 23712-23712/com.lint0t.androidclassdemo D/SecondActivity: onDestroy ---> 即将销毁时调用
```

Activity 切换时, 生命周期的变化

当 A_activity 切换 B_activity 的时候, 程序的执行流程如下

```
A_activity: onCreate() -> onStart() -> onResume -> onPause()
B_activity: onCreate() -> onStart() -> onResume()
A_activity: onStop() -> onDestroy()
```

Activity 的进程优先级 (从高到低排列)

1. 前台进程 (ps: 目前你们只用看第一点就行了, 第二点暂时不用管)
 - 1) 当前 Activity 正在与用户交互)
 - 2) 当前进程 service 正在与 activity 进行交互或者当前 service 调用了 `startForeground()` 属于前台进程或者当进程持有一个 `BroadcastReceiver`, 这个 `BroadcastReceiver` 正在执行 `onReceive()` 方法
2. 可见进程 (ps: 目前你们只用看第一点就行了, 第二点暂时不用管)
 - 1) 进程持有一个 activity, 这个 activity 不在前台, 处于 `onPause()` 状态下, 当前覆盖的 activity 是以弹窗形式存在的。
 - 2) 进程持有一个 service, 这个 service 是和一个可见的 activity 进行绑定的
3. 后台进程
activity 的 `onStop()` 被调用, 但是 `onDestroy()` 没有被调用的状态
4. 空进程
该进程没有任何运行的数据了, 且保留内存空间, 并没有被系统 killed, 属于空进程

Activity 的四种启动模式

1. standard
Activity 的默认的启动模式。在这个模式下, 每启动一个新的 Activity, 它就会进入返回栈, 并处于栈顶位置。对于使用本模式的 Activity, 系统不会在乎这个 Activity 是否已经处于返回栈的栈顶。每次启动都会创建一个新的 Activity
2. singleTop
当一个 Activity 的启动模式被指定为本模式时, 在启动该 Activity 的时候, 如果发现返回栈的栈顶已经是该 Activity, 则认为直接使用它, 不会再创建新的 Activity 实例。

3. singleTask

让被标记的Activity在整个程序的上下文中只存在一个实例。当一个Activity启动方式被指定为singleTask后，每次启动该Activity的时候系统会首先在返回栈中检查是否有该Activity的实例，如果发现存在则直接使用该实例，并把在这个Activity之上的所有Activity通通出栈，如果没有发现则创建一个该Activity的实例。

4. singleInstance

被指定为singleInstance模式的Activity会启用一个新的栈来管理这个Activity。意义所在，假如我们的一个Activity是允许被其他程序屏用的，如果我们想实现其他程序和我们的程序共享这个Activity的实例，其他三种模式是肯定做不到的，因为每一个应用程序都会有自己的返回栈，用一个Activity在不痛的返回栈中入栈时必定是创建了新的实例。而本模式下则会有一个单独的栈来管理这个Activity，不管是那个应用程序来访问这个Activity，都用的同一个返回栈，也就解决了前面提出的问题。

Activity之间的通信

- Intent
显示跳转、隐式跳转 (本节课不讲，课后看书理解)
- intent.putExtra()
通过这个向下一个activity传递数据。
- startActivityForResult
通过这个向下一个activity传递数据后，获取返回值

XML及Android常用控件

1. android前缀

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

- 这句话的意思是，声明这个命名空间引用的是Android系统的，而其中的android作为前缀，是这个引用别称的意思 (ps: 换成其他的也不是不行)
- 后面的schemas的意思是xml文件的约束 (也就是xml的书写规范)，还有一种xml的约束是DTD，但是被DTD取代了
- 有了这个，Android Studio就会在我们编写布局文件的时候给出提示，提示我们可以输入什么，不可以输入什么。也可以理解为语法文件吗，或者语法判断器

2. app前缀

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

- 在项目需求中，我们往往使用系统自带的属性和控件是不够的，我们可能需要导入自定义控件的一些属性，或者support支持包之类的
- 为了引入自定义属性，我们以xmlns:前缀=<http://schemas.android.com/apk/res/>你的应用程序包路径，将其导入，但是现在的普遍走法是使用xmlns:app="<http://schemas.android.com/apk/res-auto>"，因为res-auto可以引用所有的自定义包名。

3. tool前缀

```
xmlns:tools="http://schemas.android.com/tools"
```

- tools可以告诉Android Studio，那些属性在运行的时候是被忽略的，只是在设计布局的时候有效果

- tools可以覆盖android所有的标准属性，将 android: 换成 tools: 即可，而且在运行 的时候连tools本身都是被忽略的，不会被带进apk

4. 一些常用属性及控件

1) dp

安卓中的相对大小

其实dp就是为了使得开发者设置的长度能够根据不同屏幕(分辨率/尺寸也就是dpi)获得 不同的像素(px)数量。比如：我将一个控件设置长度为1dp，那么在160dpi屏幕上该控 件长度为1px，在240dpi的屏幕上该控件的长度为 $1240/160=1.5$ 个像素点。

也就是dp会随着不同屏幕而改变控件长度的像素数量。

关于dp的官方叙述为当屏幕每英寸有160个像素时(也就是160dpi)，dp与px等价的。那 如果每英寸240个像素呢？ $1dp \rightarrow 1 \times 240 / 160 = 1.5px$ ，即1dp与1.5px等价了。

其实记住一点，**dp最终都要化为像素数量来衡量大小的，因为只有像素数量最直观。**

2) dpi (dot per inch)

每英寸像素多少

要想判别手机屏幕的显示好坏，还要考虑屏幕的宽高(英寸)，也就是用dpi即每英寸多少像素 来评价屏幕的显示效果。（不然假如手机分辨率是 1920×1080 ，但是屏幕是几十寸的，那显 示效果将不会很好，甚至你有可能看到小的像素块，那将更影响视觉效果。）

3) px

像素点

平常所说的 1920×1080 只是像素数量，也就是 $1920px \times 1080px$ ，代表手机高度上有1920个 像素点，宽度上有1080个像素点。

4) sp

sp除了能够像dp一样可以适应屏幕密度的变化，还可以随着系统字体的大小设置改变作出变化。 如果不想文字随着手机设置中字体的大小发生改变（例如标题），可以使用dp代替。

```
android:id
android:layout_width
android:layout_height
android:margin
android:padding
android:background
```

- TextView 文本
- Button 按钮
- EditText 输入框
- ImageView 图片

5. 布局

课上将会演示前三个布局

◦ LinearLayout(线性布局)

一种非常常用的布局，这个布局中的控件会在线性方向排列，可以通过android:orientation这个 属性来指定线性排列的方向是垂直的(vertical)还是水平的(horizontal)

这里讲一下 android:layout_gravity 和 android:gravity 。这两个都是设置对齐方式的 属性，内部 的值都相同

1. android:layout_gravity 是设置自身相当于父容器的对齐方式。比如一个TextView 设置 layout_gravity属性，则表示这个TextView本身相对于父容器的对齐方式。

2. 需要注意，如果要使用gravity属性的话，该组件的 layout_width 和 layout_height 不能设置为wrap_content，此时设置的gravity属性没有效果，因为组件包裹着内容，无论设置什么，也不能有改变
3. android:gravity 是设置自身内部元素的对齐方式。比如一个TextView，则是设置的 内部文字的对齐方式。如果是ViewGroup组件是LinearLayout的话，则设置它内部view组件的对齐方式。（但是如果在FrameLayout中，这个属性就没有任何的作用） layout_gravity 属性不是什么情况下都能设置的属性（比如LinearLayout的排列方向 是horizontal时，只有在垂直的方向上才会生效，因为水平方向上的长度是不固定的，每添加一个控件，水平方向上的长度都会改变，因而无法指定该方向上的对齐方式），而且在不同的ViewGroup中也会产生的效果也会不同

- **ConstraintLayout(约束布局)**

在2016年之前，如果我们要写一些设计很复杂的页面，我们可能会借助于嵌套实现，但是我们嵌套的越多，性能就越低。为了提升开发者的可视化编程，谷歌官方在2016年的I/O大会上提出了新的组件ConstrainLayout(约束布局)。

- **FrameLayout(帧布局)**

这种布局，所有的控件都会默认摆在布局的左上角

- **RelativeLayout(相对布局)**

它可以通过相对定位的方式让控件出现在布局的任何位置，属性非常多。不过她的属性十分有规律，目前基本可以用ConstraintLayout替代。

- **PercentFrameLayout(百分比布局)**

这种布局最大的特点就是我们可以不使用wrap_content、match_parent等方式来指定控件的大小，而是允许直接使用控件在布局中所占的百分比

一些杂项

- 使用Log而不是System.out.println
- Toast
- String及color

代码规范

xml中控件id一般为：

控件简写_用途

如在MainActivity中，用于登录的按钮控件，一般为

btn_main_login

在Java文件中要使用驼峰命名法，如

mButton

更多规范请参看群文件《阿里巴巴Android开发手册》

写在最后

大家已经坚持了半个学期，接下来就是精彩纷呈的Android世界，希望大家不要放弃。既然选择了成为红岩网校的一名学员，牺牲休息时间来上课，做作业，那么就证明你的初心一定是追求更卓越的自己。我们希望你们在时间的冲刷下不要忘了自己的初心。既然选择了远方，便只顾风雨兼程。去年这个时候，我心中一片迷茫，因为我Java掌握的一般，Android又昏头昏脑（其实是我太菜了，哈哈），但我没有放弃，因为我想起了一开始加入红岩时的那份热血，那份初心。你们这一届，真的很多人都很优秀，绝大部分人已经超越大一时候的我太多太多，就算有些同学现目前进度有些跟不太上也没有关系，你们还有这学期剩下的时间和一整个寒假的时间来弥补。大家来到红岩，成为移动开发部的一名学员，那就要真真正正的学到东西，不要辜负了自己，这也是我们所希望的。大家现在回头想一想，数据类型，方法，集合，类，继承，封装，多态，多线程，泛型，异常……你已经学到了很多很多的东西，再和一开始什么都不会的你比一比，是不是感觉到了自己的进步，和进步带来的喜悦？如果开心的感觉充满你的心中，那就对了，就是这种感觉，希望你在之后的旅程中，想要放弃时，回忆一下这种感觉，然后收拾收拾心情，砥砺前行，在下一个山顶回头看看，之前的困难不过小如蚂蚁。