# Android适配

从事Android开发多年，但是一直没了解过Android适配，由于近期从事手持超声Android版本部分UI工作，故学习了解Android适配方面的知识。

本文先了解一下Android适配方面的基本概念，然后了解市面上主流的适配方案。

## 基本概念

我们在AndroidStudio中创建虚拟机时，如下图，可以看见一部手机有以下相关参数。

**分辨率**：手机屏幕的像素点数，一般描述成屏幕的“宽×高”，eg:1080\*1920px

**屏幕大小：**手机对角线的物理尺寸，以英寸（inch）为单位。eg:5英寸

**像素密度**（dpi，dots per inch）每英寸的像素点数，数值越高当然显示越细腻。

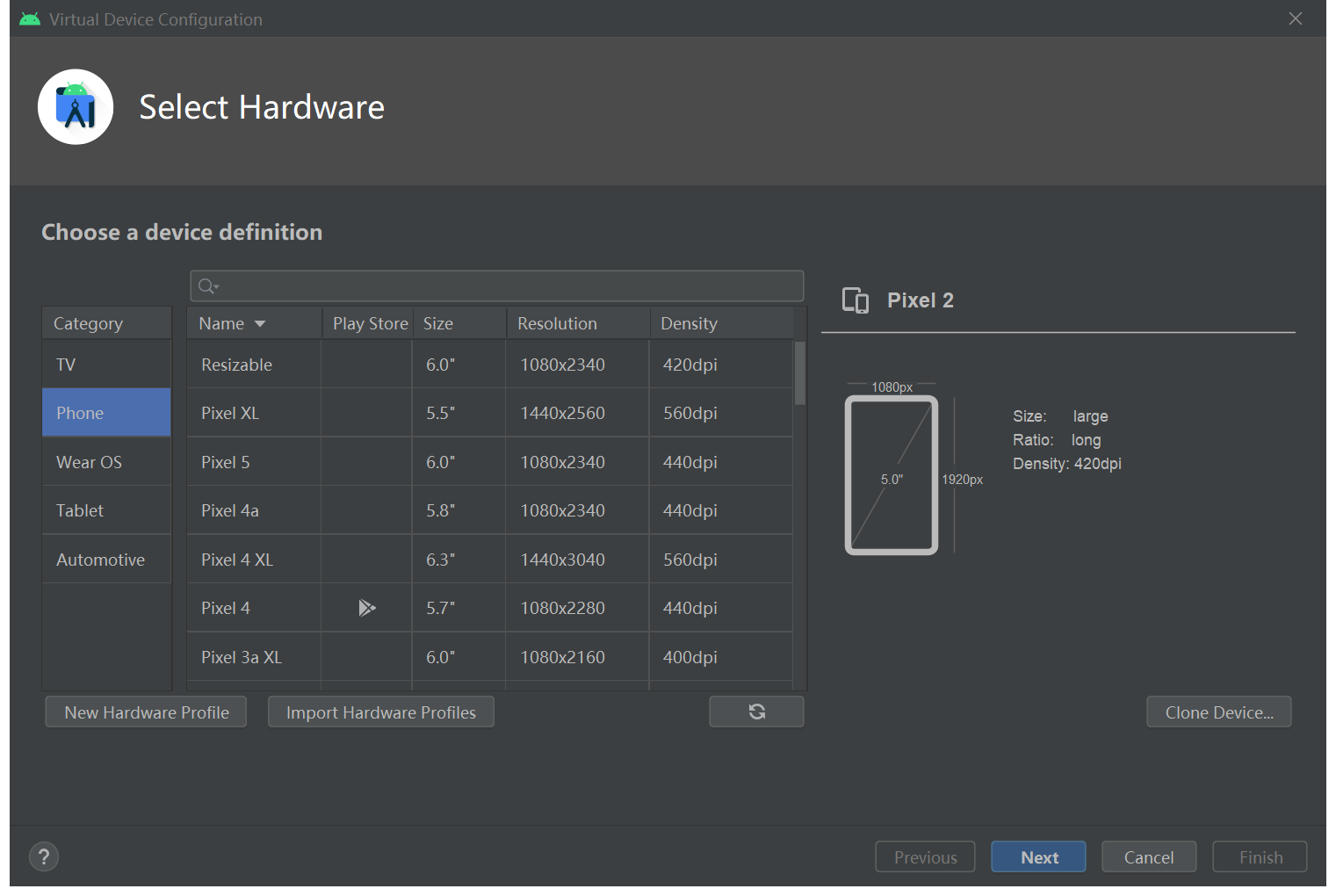
**密度**（Density），计算公式px = dendity \* dp ，表示**1 dp** 占当前设备多少像素

px = density \* dp;

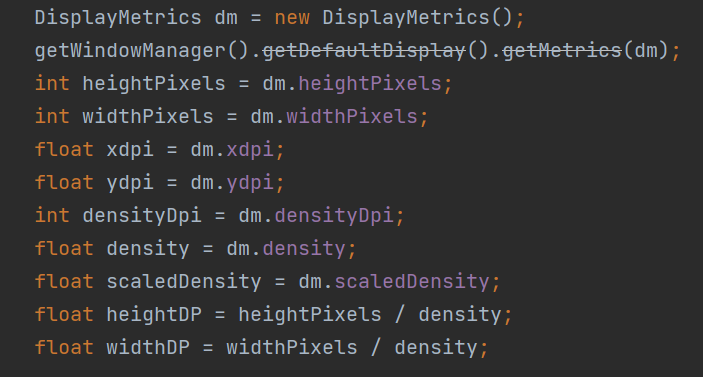
density = dpi / 160;

px = dp \* (dpi / 160);

假如我们知道一部手机的分辨率是1080×1920(实际是1080\*1857)，屏幕大小是5（实际是5.11）英寸，你 能否算出此屏幕的密度呢？哈哈，中学的勾股定理派上用场啦！通过宽1080和高1920，根据勾股定理，我们得出对角线的像素数大约是2203，那么用 2203除以5就是此屏幕的密度了，计算结果是440。440dpi的屏幕已经相当细腻了



下图是获取手机相关参数的方法：

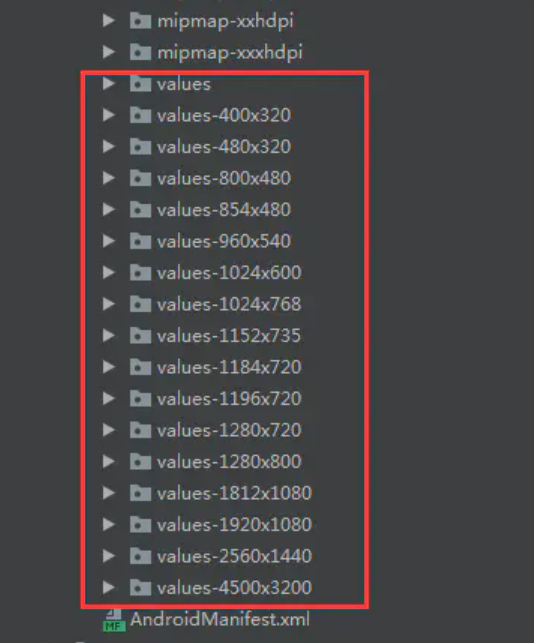


## 适配方案

目前主流的适配工作都是采用百分比的思想。根据基准尺寸与目标手机尺寸的百分比，计算器对应控件的宽高。下面四种方案都是采用百分比的思想。

### 宽高限定符适配

宽高限定符适配也可以叫屏幕分辨率适配，需要在res文件下创建各种针对不同屏幕分辨率对应的values文件夹。



当软件运行在手机上时，系统会根据手机的屏幕分辨率去找对应的values文件，如手机分辨率是1280X800,这个时候系统就会加载values-1280x800的dimens文件，这样设计图上设计的控件在手机上显示就正常了。达到了适配的需求，简而言之就是，需要创建不同分辨率的文件夹，系统会根据当前手机的分辨率去加载对应文件。如果手机的分辨率种类太多，就会造成需要创建的文件较多，导致apk包变大。而且只有屏幕分辨率和文件完全匹配才能进行适配，否则就只能用统一的默认的dimens文件了。而使用默认的尺寸的话，UI就很可能变形。

其原理如下：

以UI提供的尺寸为基准尺寸，其他分辨率都根据这个基准分辨率来计算，在不同的尺寸文件夹内部，根据该尺寸编写对应的dimens文件。

比如以480x320为基准分辨率  
宽度为320，将任何分辨率的宽度整分为320份，取值为x1-x320  
高度为480，将任何分辨率的高度整分为480份，取值为y1-y480

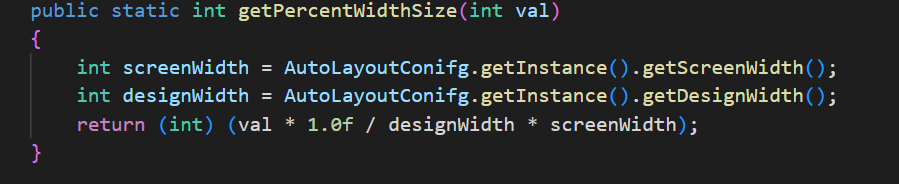
那么对于800\*480的分辨率的dimens文件来说，  
x1=(480/320)\*1=1.5px  
x2=(480/320)\*2=3px

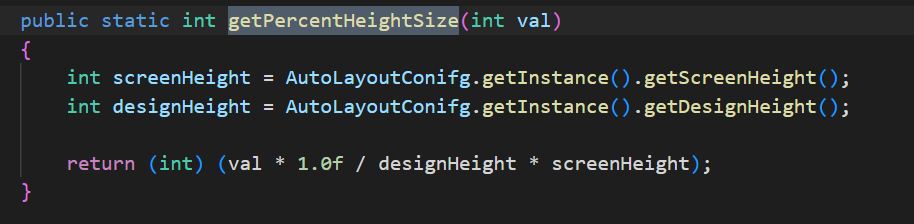
### AndroidAutoLayout

参考网址如下：

https://github.com/hongyangAndroid/AndroidAutoLayout

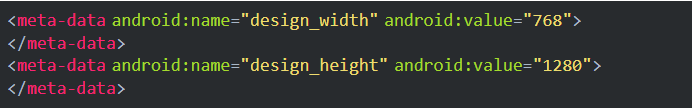
这是张鸿洋写的UI适配框架，主要是在View的OnMeasure方法中根据基准尺寸和实际尺寸的比例计算出合适的size，然后重新设置给View。





#### 使用方法：

第一步：  
在你的项目的AndroidManifest中注明你的设计稿的尺寸。



第二步：

让你的Activity继承自AutoLayoutActivity.

然后我们就可以直接在布局文件里面使用具体的像素值了，比如，设计稿上是96\*96,那么我们可以直接写96px，APP运行时，框架会帮助我们根据不同手机的具体尺寸按比例伸缩。

这可以说是一个极好的方案，因为它在宽高限定符适配的基础上更进一步，并且解决了容错机制的问题，可以说完美的达成了开发高效和适配精准的两个要求。

但是我们能够想到，因为框架要在运行时会在onMeasure里面做变换，我们自定义的控件可能会被影响或限制，可能有些特定的控件，需要单独适配，这里面可能存在的暗坑是不可预见的，还有一个比较重要的问题，那就是整个适配工作是有框架完成的，而不是系统完成的，一旦使用这个框架，未来一旦遇到很难解决的问题，替换起来是非常麻烦的，而且项目一旦停止维护，后续的升级就只能靠你自己了，这种代价团队能否承受？并且这个框架已经停止维护了。

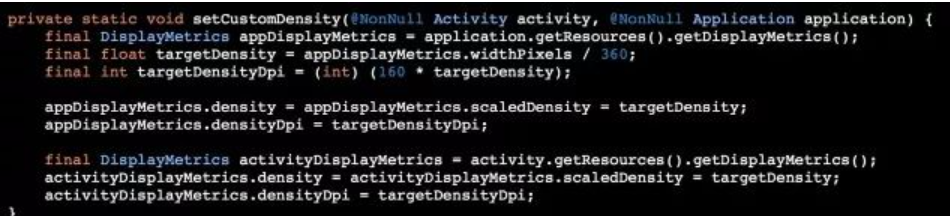
### Android AutoSize适配方案

从dp和px的转换公式 ：px = dp \* density ，可以看出，UI工程师给出指定宽度的设计图，假如为360dp,想要保证在所有设备计算得出的px值都正好是屏幕宽度的话，我们只能修改 density 的值，density=px/360;

Android中的dp和px的转换基本都是通过 DisplayMetrics（DisplayMetrics 实例通过 Resources#getDisplayMetrics 可以获得，而Resouces通过Activity或者Application的Context获得）中相关的值来计算的。

先来熟悉下 DisplayMetrics 中和适配相关的几个变量：

* DisplayMetrics#density 就是上述的density
* DisplayMetrics#densityDpi 就是上述的dpi
* DisplayMetrics#scaledDensity 字体的缩放因子，正常情况下和density相等，但是调节系统字体大小后会改变这个值
* 主要思路如下：

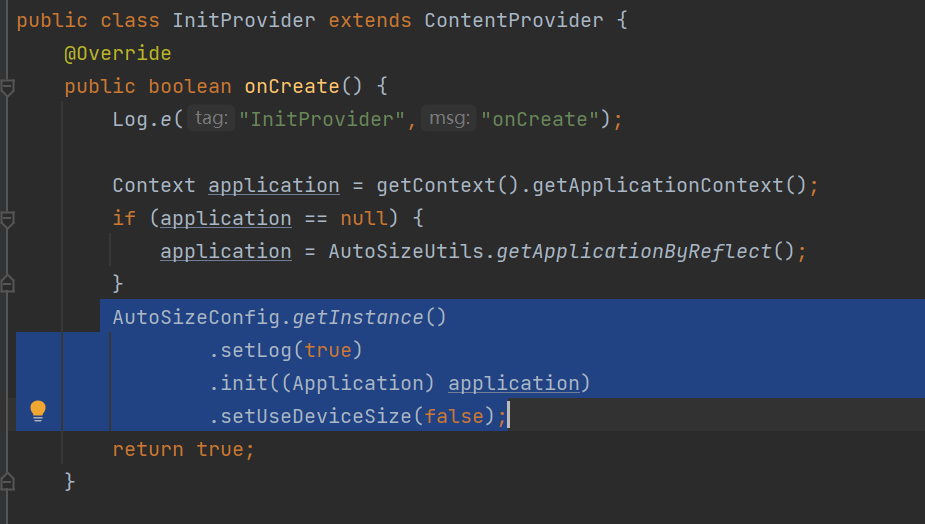


#### 使用方法：

1. 获取autosize库文件。
   1. 在MVNRepository <https://mvnrepository.com/artifact/me.jessyan/autosize/1.2.1> 下载AutoSize.aar文件
   2. 下载AutoSize< [https://github.com/JessYanCoding/AndroidAutoSize>源码，手动编译autosize.aar](https://github.com/JessYanCoding/AndroidAutoSize%3e源码，手动编译autosize.aar)
2. 在AndroidManifest.xml文件中添加UI给定尺寸的基准宽高



由于AutoSize方案是在InitProvider（继承于ContentProvider）的onCreate方法中设置的density。这种方案侵入性极小，和代码完全解耦，只需在AndroidMainfest.xml中添加目标宽高的配置项即可。再者由于修改的 **density** 在整个项目中是全局的，所以只要一次修改，项目中的所有地方都会受益。



### SmllestWidth方案

#### 什么是 smallestWidth

**smallestWidth** 翻译为中文的意思就是 **最小宽度**，那这个 **最小宽度** 是什么意思呢？

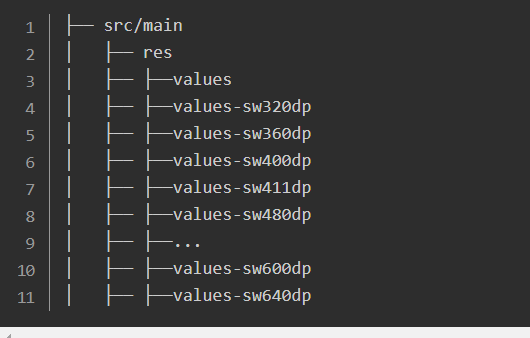
系统会根据当前设备屏幕的 **最小宽度** 来匹配 **values-sw<N>dp**，为什么不是根据 **宽度** 来匹配，而要加上 **最小** 这两个字呢？

这就要说到，移动设备都是允许屏幕可以旋转的，当屏幕旋转时，屏幕的高宽就会互换，加上 **最小** 这两个字，是因为这个方案是不区分屏幕方向的，它只会把屏幕的高度和宽度中值最小的一方认为是 **最小宽度**，这个 **最小宽度** 是根据屏幕来定的，是固定不变的，意思是不管您怎么旋转屏幕，只要这个屏幕的高度大于宽度，那系统就只会认定宽度的值为 **最小宽度**，反之如果屏幕的宽度大于高度，那系统就会认定屏幕的高度的值为 **最小宽度**

如果想让屏幕宽度随着屏幕的旋转而做出改变该怎么办呢？可以再根据 **values-w<N>dp** (去掉 **sw** 中的 **s**) 生成一套资源文件

如果想区分屏幕的方向来做适配该怎么办呢？那就只有再根据 **屏幕方向限定符** 生成一套资源文件咯，后缀加上 -land 或 -port 即可，像这样，values-sw400dp-land (最小宽度 400 dp 横向)，values-sw400dp-port (最小宽度 400 dp 纵向)

smallestWidth 是宽高限定符适配的升级， 只是把 **dimens.xml** 文件中的值从 **px** 换成了 **dp**，根据dp值进行等比例缩放，其使用方式是没变的。



开发者先在项目中根据主流屏幕的 最小宽度 (smallestWidth) 生成一系列 values-sw<N>dp 文件夹 (含有 dimens.xml 文件)，当把项目运行到设备上时，系统会根据当前设备屏幕的 最小宽度 (smallestWidth) 去匹配对应的 values-sw<N>dp 文件夹，而对应的 values-sw<N>dp 文件夹中的 dimens.xml 文字中的值，又是根据当前设备屏幕的 最小宽度 (smallestWidth) 而定制的，所以一定能适配当前设备

如果系统根据当前设备屏幕的 最小宽度 (smallestWidth) 没找到对应的 values-sw<N>dp 文件夹，则会去寻找与之 最小宽度 (smallestWidth) 相近的 values-sw<N>dp 文件夹，系统只会寻找小于或等于当前设备 最小宽度 (smallestWidth) 的 values-sw<N>dp，这就是优于 宽高限定符屏幕适配方案 的容错率，并且也可以少生成很多 values-sw<N>dp 文件夹，减轻 App 的体积。