**屏幕适配**

# 1 为什么屏幕适配

**提出需求：**

举个例子：创建一个Demo，修改HelloWorld文本的TextSize大小为140dp，在480x320分辨率上只会显示一个“Hello”如下图1.1，而在1280x720分辨率上却可以显示更多如下图1.2，这就是显示的差异性的体现

图1.2

图1.1

现在Android手机设备的分辨率的型号过多，市面上的手机分辨率各有不同，有1920x1080的，有1024x720、有1280x800的以及其他的分辨率。而手机的分辨率不一样的话，写出来的一个布局页面显示时也可能跟上面这个例子一样发生差异化。

我们企业里面要做一个应用的话，肯定是希望在每个用户的手机上显示起来都是一样的。不可能说在大屏上显示一个小人的图片，在小屏手机上就只显示一个脑袋吧？所以我们为了解决这些差异性，就需要进行屏幕适配来保证应用在各分辨率设备上的显示一致性。

# 2 图片适配

**引出：**

我们先说图片方面的适配，在座的各位，还记得咱们先前复制图片文件放在哪？（drawable-hdpi文件夹）

没错，大家记性很好。其实我们做Android项目所需要的图片，不但可以放drawable-hdpi、还可以放drawable-hdpi、drawable-mdpi、drawable-xhdpi等这些以drawbale开头的文件夹。

**讲解：**

不同像素密度的手机，会去加载不同资源文件下的图片。。。。。。。。。。。。。。。。。。。。。。。。下面来证明一下

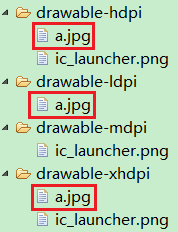
**步骤：**

创建项目；

往drawable-hdpi、drawable-mdpi、drawable-ldpi、drawable-xhdpi这些文件夹下放同名图片；

然后在界面布局文件中创建ImageView将该图片展示；

最后运行在不同分辨率手机上，查看效果。



**实现：**

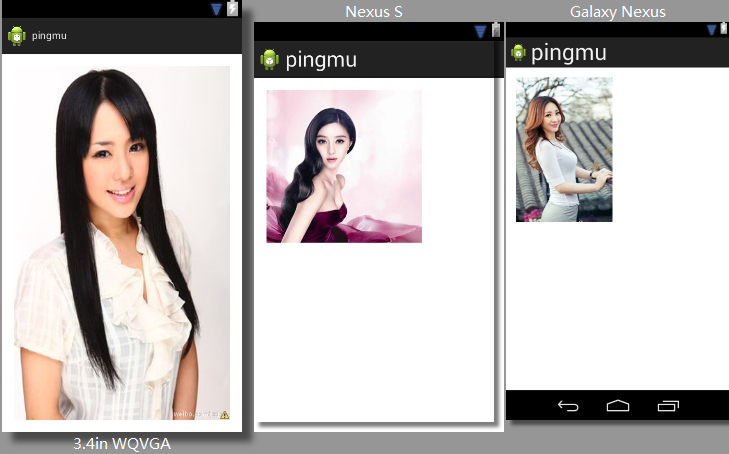
假设同一个名字不同内容的图片放在各文件夹下，ldpi放的是显示为苍老师的a.jpg、mdpi没有放图片、hdpi放的显示为范冰冰的a.jpg、xhdpi放的显示为柳岩的a.jpg。加载结果如下：

ldpi：(320\*240) 某知名女星

mdpi：(480\*320) 默认文件夹下没有图片时，将会去找最接近的高分辨率文件夹。范冰冰

hdpi：(800\*480) 范冰冰

xhdpi：(1280\*720) 柳岩

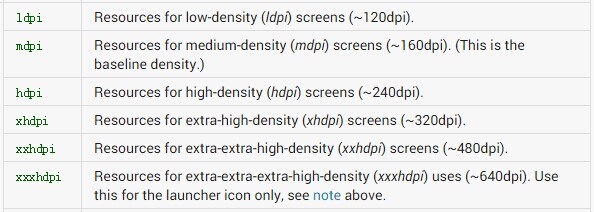


**总结+剖析原理：**

图片适配是根据什么来找对应这些文件夹的？

图片是根据手机像素密度(ppi)来决定的。其实ppi就等于dpi（每英寸点数）。手机像素密度如何计算？先由勾股定理求出对角线的密度数（w\*w+h\*h然后开根号）。再除以对角线的长度就得到密度了（对角线长度即手机的大小，商家卖手机说的屏幕大小指的就是对角线）

手机会根据像素密度来匹配下表中最近的一种分辨率。



（上图截自Android4.3版本的官方文档，详情可见官方文档，目录为sdk/docs/desigin/style/devices-displays，版本5.0文档描述为下图）



最后说明实战中的知识应用：展示一张大图时，图片太高清，小屏展示时会消耗很多资源，图片不高清，大屏展示时会失真，所以可以通过这个图片适配方法来解决这个问题。

# 2+ 使用.9图片适配

通过工具绘制.9图片，图片中的某一部分将会被拉伸。比如说聊天时的背景图片。

# 3 dimens适配

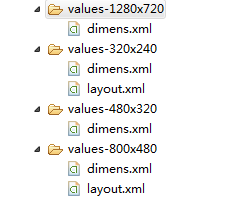
**引出：**

图片适配只能是在处理图片上的适配，如果要适配的控件不是图片的话怎么办？我们还有其他的适配方案，接下来我介绍一下另一种适配方案：dimens适配

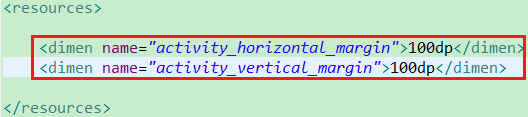
**讲解：**

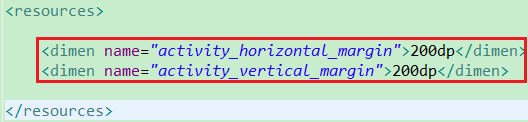
什么是dimens？dimens是dimension（尺寸）的简称，我们每个控件的宽高都可以通过一些固定值来设置尺寸，

所以，希望在不同分辨率设备上控件的宽高尺寸不一样（比如1280x720上是200px，800x480上却不按比例显示，想显示成100px），就可以通过设置dimens.xml来进行屏幕适配。通过多个values文件夹下（values-1280x720、values-480x320等等文件夹）放置dimens文件，



里面写上相同名称的dimen，如下图





就能实现对应的分辨率加载对应的dimens单位。

**步骤：**

创建项目；

提出一个需求：完成一个容器组件被两个子控件平分的效果；

通过固定值240dp来设置每个子控件的宽度（实现在800x480分辨率上平分）；

在800x480分辨率上展示平分效果，然后在1280x720分辨率上运行，发现没有平分；

引出结论：先设置一个变量a为固定值，说明变量的概念，然后再去创建dimens适配文件

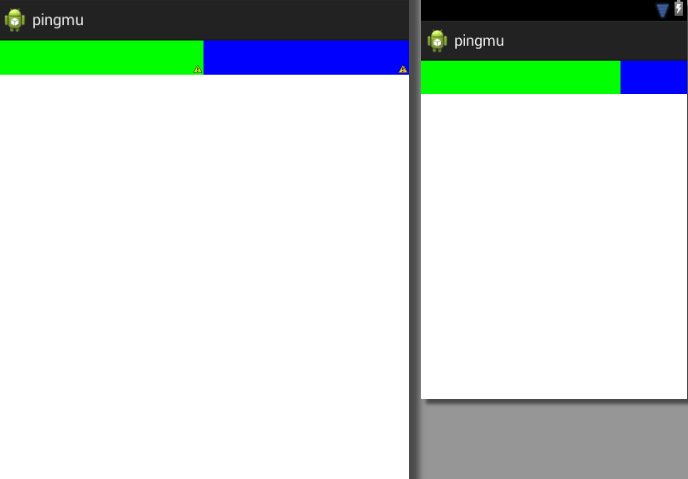
创建两个values文件夹（values-1280x720、values-800x480），然后往每个文件夹下创建dimens.xml，在里面都写上同名的dimen，但是单位不一样（一个是240dp、一个是360dp）

让子控件引用dimens文件中的单位（@dimen/....）；

分别运行在不同分辨率设备上，查看效果。

**实现：**

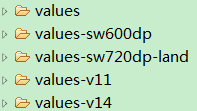
创建Demo，希望有一个容器能被两个子控件给平分，这个时候通过设置固定值来平分，在A分辨率做到了，但是在B分辨率上平分效果却做不到。



**总结+剖析原理：**

dimens适配是根据什么来找对应这些文件夹的？

图片是根据手机分辨率来查找的（即values后面跟着的1280x720这些分辨率），当然我们这里也可以写成其他的形式。如下图：



sw600dp代表宽度或高度中最小的超过了600dp的分辨率情况下，才满足了调用该文件夹下的文件的前提。

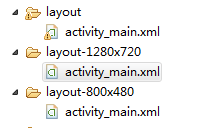
sw720dp代表宽度或高度中最小的超过了600dp的分辨率情况下，land代表为屏幕的朝向为横屏，所以必须要同时满足这两个前提，这个values-sw720dp-land文件夹才有机会被调用。

values-v11代表Android版本号为11或大于11时，才有机会被调用。

values-v14代表Android版本号为11或大于11时，才有机会被调用。

具体调用顺序参见：http://my.oschina.net/wangjunhe/blog/98739。

# 4 layout适配



引出：跟dimens适配的写法很类似的，还有一种layout适配

原理类似于dimens适配，只是改成创建一个layout布局xml文件来为某个布局页面进行单独适配。

步骤、原理仿照于第三个dimens适配，不再列出。

这里总结说明：手机运行app时，会先通过当前手机的分辨率、版本等信息，来按照格式读取不同的资源文件（格式：资源文件夹-。。。。横杠后面可以写hdpi、分辨率、版本、sw等，需要按照这个格式去写，不然Android编译生成app时会不识别，直接报错）。

项目实战：在某些特定机型展示一些不同页面布局时，可以使用该适配。实际应用较少

# 5权重适配

**引入：**

先前我们为了在多分辨率设备上实现一个平分效果，使用了dimens适配来实现，比较麻烦，我们还有一种简单的方法来完成这样的一个效果，那就是权重适配。

**讲解：**

使用LineaLayout线性布局来实现权重适配（子控件中设置weight属性）

**步骤：**

以dimens适配中的例子为例。为了能达到平分效果，先将容器设置为线性布局；

然后设置子控件的宽度属性为0dp，然后多设置一个weight属性（权重），令wight都为1；

运行在各分辨率设备上，查看效果。

**原理：**

类似于百分比划分。。。。。。。。设置weight不一定是1。

项目实战：百分比式的布局，比如每个控件线性排列，然后尺寸按屏幕百分比展示时，可以使用，比较方便，不用去dimens适配那么麻烦。

# 6 java代码适配（dp）

**引出：**

我们布局中的控件除了在xml文件中创建，还可以在代码中来创建控件。在代码中创建控件时，也需要进行屏幕适配。

**讲解：**

在代码中直接设置某些控件的宽或高，默认单位都是像素，会出现很严重的差异性，我们需要改成dp单位来解决。

说明px、dp、sp等单位

解释这些单位的含义。。。。。。。。。。

px: pixels(像素). 1920x1080代表屏幕的宽度方向上最多有1080个像素，屏幕长度方向上最多有1920个像素。

dip: device independent pixels(设备独立像素). 不同设备有不同的显示效果，每个手机的像素密度不一样，有的密度高显示效果好，有的密度低显示效果差， 它不依赖像素。 使用dip单位，可以消除像素密度对显示的影响。

dp: dip是一样的

pt: point，是一个标准的长度单位，1pt＝1/72英寸，用于印刷业，非常简单易用；

sp: scaled pixels(放大像素). 主要用于字体显示best for textsize。

in（英寸）：长度单位。

mm（毫米）：长度单位。

然后提到dp相关：

需要明确，控件的长宽设置即使为dp的单位形式也是不能做屏幕适配的。不同像素密度的手机的话， dp px 有一个不同的比例关系

ldpi 1dp = 0.75px 320\*240 160dp = 120px

mdpi 1dp = 1px 480\*320 160dp = 160px

hdpi 1dp = 1.5px 800\*480 160dp = 240px

xhdpi 1dp = 2px 1280\*720 160dp = 320px<360px 180dp = 360px

xxhdpi 1dp = 3px 1920\*1080 160dp = 480px < 540px

**步骤：**

在原先项目基础上，通过代码创建一个ImageView控件；

随便设个背景色，然后设置它的宽度大小为一个数值；

运行在不同分辨率的手机上，发现图片宽度的差异很大，从而得知代码中直接用数值设置控件大小不行。

引入解决方案（代码如下）：

**public** **static** **int** dp2px(Context context, **int** dp){

//获取密度

**final** **float** scale = context.getResources().getDisplayMetrics().density;

//通过密度进行换算

**return** (**int**)(dp\*scale+0.5f);

}

**总结+原理：**

这里需要在写java代码时就要说明：获取了手机屏幕密度参数，然后将数值乘以密度，再四舍五入（需要说明为什么要+0.5f）就可以将像素单位转换为dp单位了。

全部总结：图片可以通过图片适配，页面的控件尺寸可以使用dimens适配，使用特定页面使用layout适配，有百分比需求使用线性布局中权重适配，代码创建控件是的关于dp单位换算的适配。

# 7使用百分比布局进行屏幕适配

google官方提供了一个百分比布局，它可以供开发者进行屏幕适配。使用的前提是：

* Android SDK v22 or high
* Android Build Tools v22.0.1 or high
* Android Percent Support Repository v22.2.0 or high
* Android Support v4 Repository v22.2.0 or high

在你的moudle中的build.gradle文件里添加一下依赖：

dependencies {

compile 'com.android.support:percent:22.2.0'

}

该百分比布局中包含了可以使用百分比的相对布局PercentRelativeLayout和使用百分比的帧布局PercentFrameLayout。

PercentRelativeLayout用法举例：

<android.support.percent.PercentRelativeLayout

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<View

android:id="@+id/top\_left"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:layout\_alignParentTop="true"

android:background="#ff44aacc"

app:layout\_heightPercent="20%"

app:layout\_widthPercent="70%" />

<View

android:id="@+id/top\_right"

android:layout\_width="0dp"

android:layout\_height="0dp"

android:layout\_alignParentTop="true"

android:layout\_toRightOf="@+id/top\_left"

android:background="#ffe40000"

app:layout\_heightPercent="20%"

app:layout\_widthPercent="30%" />

<View

android:id="@+id/bottom"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_below="@+id/top\_left"

android:background="#ff00ff22"

app:layout\_heightPercent="80%" />

</android.support.percent.PercentRelativeLayout>