**android 关于Bitmap压缩处理解析**

原理分析

如何将一张大图压缩到100kb以下并且保持不失真的特性？这就需要用到下面这个类了。

主要运用BitmapFactory.Options

BitmapFactory.Options缩放图片主要用到inSample采样率

inSample = 1，采样后图片的宽高为原始宽高  
inSample > 1，例如2，宽高均为原图的宽高的1/2

一个采用ARGB8888的1024 1024 的图片  
inSample = 1，占用内存就 1024\*1024\*4 = 4M  
inSample = 2，占用内存就 512\*512\*4 = 1M

BitmapFactory 给我们提供了一个解析图片大小的参数类 BitmapFactory.Options ，把这个类的对象的 inJustDecodeBounds 参数设置为 true，这样解析出来的 Bitmap 虽然是个 null，但是 options 中可以得到图片的宽和高以及图片的类型。得到了图片实际的宽和高之后我们就可以进行压缩设置了，主要是计算图片的采样率。

// 第一次解析将inJustDecodeBounds设置为true,用以获取图片大小,并且不需要将Bitmap对象加载到内存中

BitmapFactory.Options options = new BitmapFactory.Options();

options.inJustDecodeBounds = true;

BitmapFactory.decodeFile(filePath, options); // 第一次解析

接下来就需要进行选定压缩的采样率了。目前市场上的主流手机分辨率一般最低是720\*1280了所以就按照此分辨率进行压缩

//原始图片的宽度与720的比值，然后向上取整这里为8

int wRatio = (int) Math.ceil(options.outWidth / (float) 720);

//原始图片的高度与1280的比值，然后向上取整这里为3

int hRatio = (int) Math.ceil(options.outHeight / (float) 1280);

//获取采样率

if (wRatio > 1 && hRatio > 1) {

if (wRatio > hRatio) {

options.inSampleSize = wRatio;

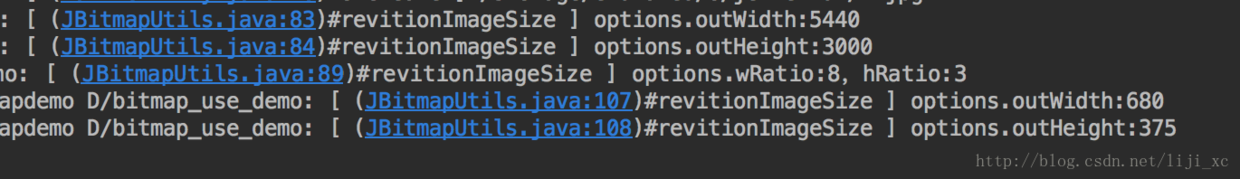
} else {

options.inSampleSize = hRatio;

}

}

经过上面这个采样率进行压缩后的宽和高肯定是小于720\*1270的，我们计算的结果是：680\*375



我们来实际比较一下压缩结果：

原先：5440\*3000  
如果采用ARGB\_8888模式的话，那么如果不压缩直接加载到内存的话，那么它将占：

5440/1024 3000/1024 4 = 62.25M，不崩溃才怪呢~

那么现在：680\*375  
见证奇迹的时候，680/1024 375/1024 4=0.9M

两者一比较的话，那么效果还是比较明显的，相差大约64倍，所以还是可以的。当然了经过上面的压缩方法，我们将压缩后的图片上传到服务器的话，那么将会大大的减少流量同时也会减少上传超时的几率的。

当然了，如果还嫌大的话，我们可以进一步增加压缩的比例，可以设置成480\*800，那么这样的话，质量肯定是有所下降的。