**Google推荐的图片加载库Glide介绍**

泡在网上的日子 发表于 2015-03-27 13:02 第 23367 次阅读 [Glide](http://jcodecraeer.com/tags.php?/Glide/)

**5**

**编辑推荐：**[稀土掘金](http://gold.xitu.io/)，这是一个针对技术开发者的一个应用，你可以在掘金上获取最新最优质的技术干货，不仅仅是Android知识、前端、后端以至于产品和设计都有涉猎，想成为全栈工程师的朋友不要错过！

在泰国举行的谷歌开发者论坛上，谷歌为我们介绍了一个名叫 [Glide](https://github.com/bumptech/glide) 的图片加载库，作者是bumptech。这个库被广泛的运用在google的开源项目中，包括2014年google I/O大会上发布的官方app。

它的成功让我非常感兴趣。我花了一整晚的时间把玩，决定分享一些自己的经验。在开始之前我想说，Glide和Picasso有90%的相似度，准确的说，就是Picasso的克隆版本。但是在细节上还是有不少区别的。

https://github.com/bumptech/glide

**导入库**

Picasso和Glide都在jcenter上。在项目中添加依赖非常简单：

**Picasso**

|  |  |
| --- | --- |
| 1  2  3 | dependencies {      compile 'com.squareup.picasso:picasso:2.5.1'  } |

**Glide**

|  |  |
| --- | --- |
| 1  2  3  4 | dependencies {      compile 'com.github.bumptech.glide:glide:3.5.2'      compile 'com.android.support:support-v4:22.0.0'  } |

Glide需要依赖**Support Library v4**，别忘了。其实**Support Library v4**已经是应用程序的标配了，这不是什么问题。

**基础**

就如我所说的Glide和Picasso非常相似，Glide加载图片的方法和Picasso如出一辙。

**Picasso**

|  |  |
| --- | --- |
| 1  2  3 | Picasso.with(context)      .load("<http://inthecheesefactory.com/uploads/source/glidepicasso/cover.jpg>")      .into(ivImg); |

**Glide**

|  |  |
| --- | --- |
| 1  2  3 | Glide.with(context)      .load("<http://inthecheesefactory.com/uploads/source/glidepicasso/cover.jpg>")      .into(ivImg); |

虽然两者看起来一样，但是Glide更易用，因为Glide的with方法不光接受Context，还接受Activity 和 Fragment，Context会自动的从他们获取。

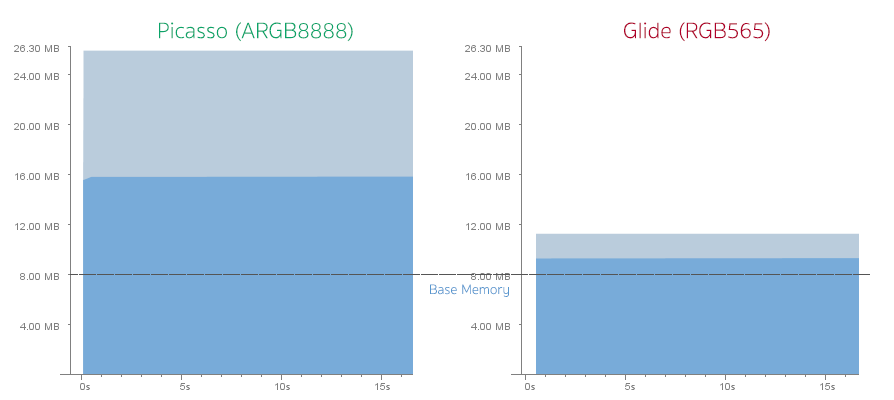
同 时将Activity/Fragment作为with()参数的好处是：图片加载会和Activity/Fragment的生命周期保持一致，比如 Paused状态在暂停加载，在Resumed的时候又自动重新加载。所以我建议传参的时候传递Activity 和 Fragment给Glide，而不是Context。

**默认Bitmap格式是RGB\_565**

下面是加载图片时和Picasso的比较（1920x1080 像素的图片加载到768x432的ImageView中）



可以看到Glide加载的图片质量要差于Picasso（ps：我看不出来哈），为什么？这是因为Glide默认的Bitmap格式是RGB\_565 ，比ARGB\_8888格式的内存开销要小一半。下面是Picasso在ARGB8888下与Glide在RGB565下的内存开销图（应用自身占用了8m，因此以8为基准线比较）：



如果你对默认的RGB\_565效果还比较满意，可以不做任何事，但是如果你觉得难以接受，可以创建一个新的GlideModule将Bitmap格式转换到ARGB\_8888：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | public class GlideConfiguration implements GlideModule {        @Override      public void applyOptions(Context context, GlideBuilder builder) {          // Apply options to the builder here.          builder.setDecodeFormat(DecodeFormat.PREFER\_ARGB\_8888);      }        @Override      public void registerComponents(Context context, Glide glide) {          // register ModelLoaders here.      }  } |

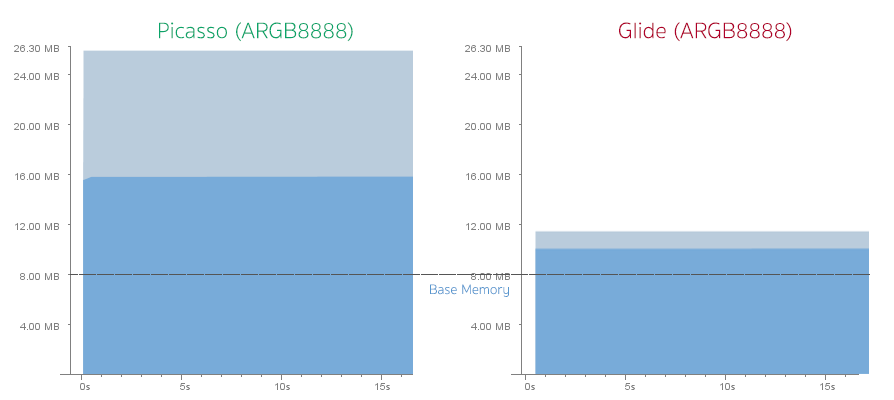
同时在AndroidManifest.xml中将GlideModule定义为meta-data

|  |  |
| --- | --- |
| 1  2 | <meta-data android:name="com.inthecheesefactory.lab.glidepicasso.GlideConfiguration"              android:value="GlideModule"/> |



这样看起来就会好很多。

我们再来看看内存开销图，这次貌似Glide花费了两倍于上次的内存，但是Picasso的内存开销仍然远大于Glide。



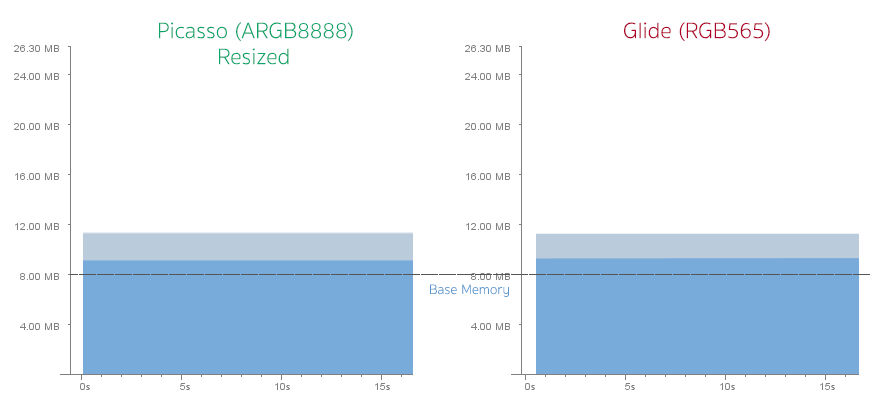
原因在于Picasso是加载了全尺寸的图片到内存，然后让GPU来实时重绘大小。而Glide加载的大小和ImageView的大小是一致的，因此更小。当然，Picasso也可以指定加载的图片大小的：

|  |  |
| --- | --- |
| 1  2  3  4 | Picasso.with(this)      .load("<http://nuuneoi.com/uploads/source/playstore/cover.jpg>")      .resize(768, 432)      .into(ivImgPicasso); |

但是问题在于你需要主动计算ImageView的大小，或者说你的ImageView大小是具体的值（而不是wrap\_content），你也可以这样：

|  |  |
| --- | --- |
| 1  2  3  4  5 | Glide.with(this)      .load("<http://nuuneoi.com/uploads/source/playstore/cover.jpg>")      .fitCenter()     .centerCrop()      .into(ivImgPicasso); |

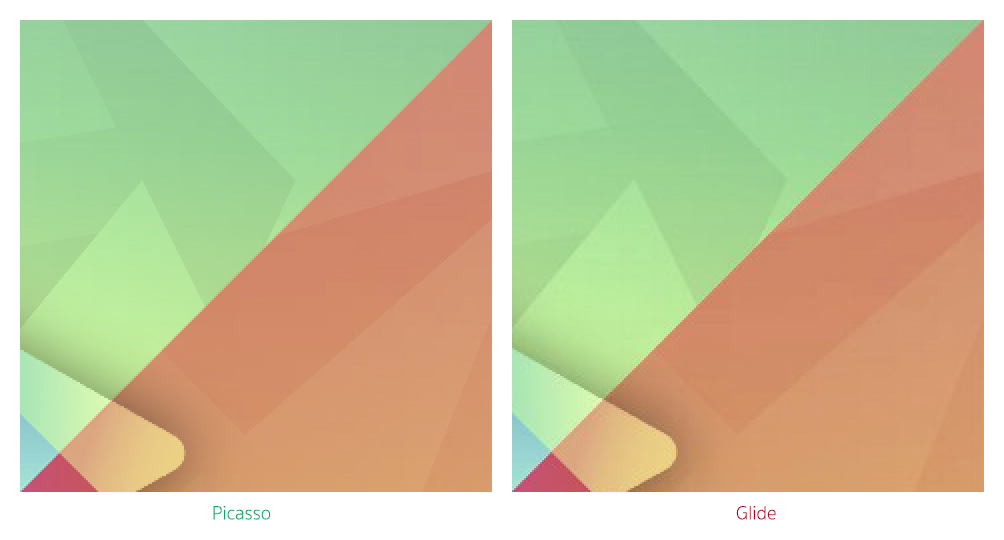
现在Picasso的内存开销就和Glide差不多了。



虽然内存开销差距不到，但是在这个问题上Glide完胜Picasso。因为Glide可以自动计算出任意情况下的ImageView大小。

**Image质量的细节**

这是将ImageView还原到真实大小时的比较。



你可以看到，Glide加载的图片没有Picasso那么平滑，我还没有找到一个可以直观改变图片大小调整算法的方法。

但是这并不算什么坏事，因为很难察觉。

**磁盘缓存**

Picasso和Glide在磁盘缓存策略上有很大的不同。Picasso缓存的是全尺寸的，而Glide缓存的是跟ImageView尺寸相同的。



上面提到的平滑度的问题依然存在，而且如果加载的是RGB565图片，那么缓存中的图片也是RGB565。

我 尝试将ImageView调整成不同大小，但不管大小如何Picasso只缓存一个全尺寸的。Glide则不同，它会为每种大小的ImageView缓存 一次。尽管一张图片已经缓存了一次，但是假如你要在另外一个地方再次以不同尺寸显示，需要重新下载，调整成新尺寸的大小，然后将这个尺寸的也缓存起来。

具体说来就是：假如在第一个页面有一个200x200的ImageView，在第二个页面有一个100x100的ImageView，这两个ImageView本来是要显示同一张图片，却需要下载两次。

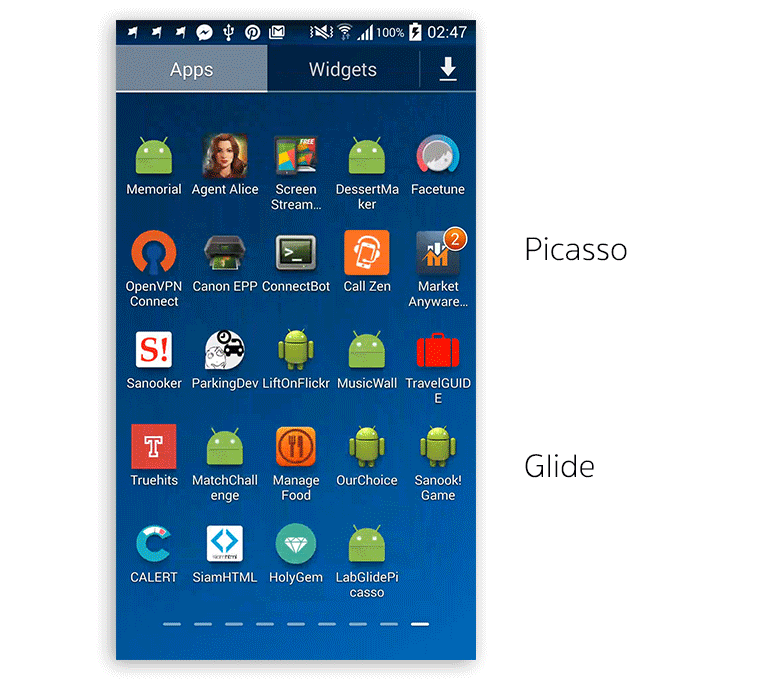
不过，你可以改变这种行为，让Glide既缓存全尺寸又缓存其他尺寸：

|  |  |
| --- | --- |
| 1  2  3  4 | Glide.with(this)       .load("<http://nuuneoi.com/uploads/source/playstore/cover.jpg>")       .diskCacheStrategy(DiskCacheStrategy.ALL)       .into(ivImgGlide); |

下次在任何ImageView中加载图片的时候，全尺寸的图片将从缓存中取出，重新调整大小，然后缓存。

Glide的这种方式优点是加载显示非常快。而Picasso的方式则因为需要在显示之前重新调整大小而导致一些延迟，即便你添加了这段代码来让其立即显示：

|  |  |
| --- | --- |
| 1  2 | //Picasso  .noFade(); |



Picasso和Glide各有所长，你根据自己的需求选择合适的。

对我而言，我更喜欢Glide，因为它远比Picasso快，虽然需要更大的空间来缓存。

**特性**

你可以做到几乎和Picasso一样多的事情，代码也几乎一样。

**Image Resizing**

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Picasso  .resize(300, 200);    // Glide  .override(300, 200); |

**Center Cropping**

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Picasso  .centerCrop();    // Glide  .centerCrop(); |

**Transforming**

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Picasso  .transform(new CircleTransform())    // Glide  .transform(new CircleTransform(context)) |

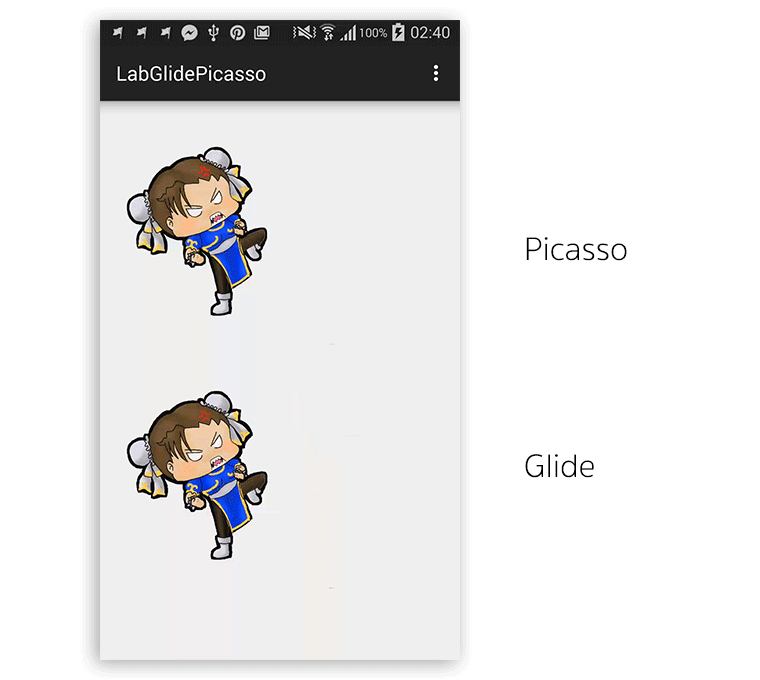
设置占位图或者加载错误图：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Picasso  .placeholder(R.drawable.placeholder)  .error(R.drawable.imagenotfound)    // Glide  .placeholder(R.drawable.placeholder)  .error(R.drawable.imagenotfound) |

几乎和Picasso一样，从Picasso转换到Glide对你来说就是小菜一碟。

**有什么Glide可以做而Picasso 做不到**

Glide可以加载GIF动态图，而Picasso不能。



同时因为Glide和Activity/Fragment的生命周期是一致的，因此gif的动画也会自动的随着Activity/Fragment的状态暂停、重放。Glide 的缓存在gif这里也是一样，调整大小然后缓存。

但是从我的一次测试结果来看Glide 动画会消费太多的内存，因此谨慎使用。

除了gif动画之外，Glide还可以将任何的本地视频解码成一张静态图片。

还有一个特性是你可以配置图片显示的动画，而Picasso只有一种动画：fading in。

最后一个是可以使用thumbnail()产生一个你所加载图片的thumbnail。

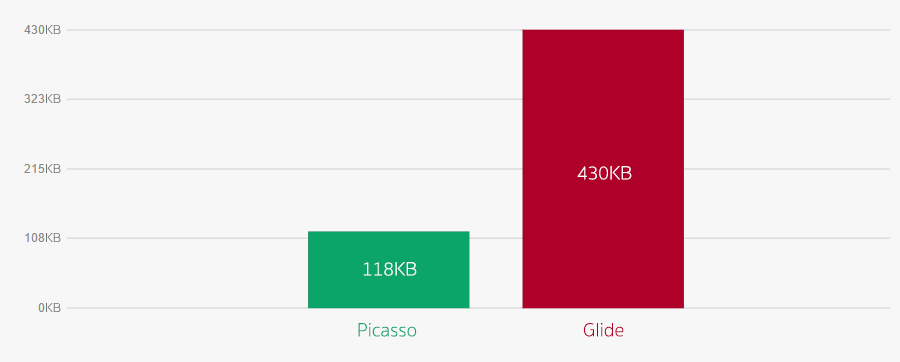
其实还有一些特性，不过不是非常重要，比如将图像转换成字节数组等。

配置

有许多可以配置的选项，比如大小，缓存的磁盘位置，最大缓存空间，位图格式等等。可以在这个页面查看这些配置[Configuration](https://github.com/bumptech/glide/wiki/Configuration) 。

**库的大小**

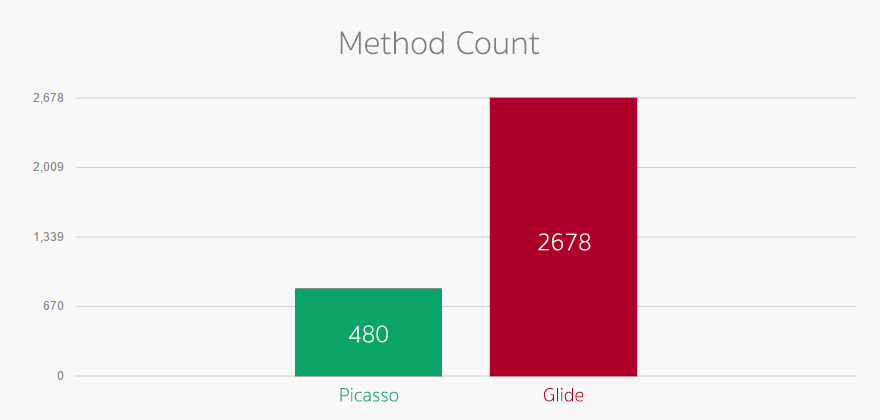
Picasso (v2.5.1)的大小约118kb，而Glide (v3.5.2)的大小约430kb。



Anyway 312KB difference might not be that significant.

不过312kb的差距并不是很重要。

Picasso和Glide的方法个数分别是840和2678个。



必须指出，对于DEX文件65535个方法的限制来说，2678是一个相当大的数字了。建议在使用Glide的时候开启ProGuard。

//获取本地缓存，并进行存贮

**new** AsyncTask<Void, Void, File>() {  
  
 @Override  
 **protected void** onPreExecute() {  
 }  
  
 @Override  
 **protected** File doInBackground(Void... params) {  
 **try** {  
 **return** Glide  
 .*with*(**context**)  
 .load(url)  
 .downloadOnly(Target.***SIZE\_ORIGINAL***, Target.***SIZE\_ORIGINAL***)  
 .get(); *// needs to be called on background thread* } **catch** (Exception ex) {  
 **return null**;  
 }  
 }  
  
 @Override  
 **protected void** onPostExecute(File aVoid) {  
 **if** (aVoid == **null**) {  
 **return**;  
 }  
 String path = aVoid.getPath();*//获取缓存地址* Bitmap bitmap = BitmapFactory.*decodeFile*(path);  
 ImageUtils.*saveBitmap*(bitmap, login\_name + **"\_"** + md5 + **".jpg"**);  
 }  
}.execute();

快速加载gif图片  
Glide.with(this).load(url).asGif().diskCacheStrategy(DiskCacheStrategy.SOURCE).into(imageView);

gif图片显示次数

Glide.with(this).load("url").diskCacheStrategy(DiskCacheStrategy.SOURCE).into(new GlideDrawableImageViewTarget(iv, 1));

**总结**

Glide和Picasso都是非常完美的库。Glide加载图像以及磁盘缓存的方式都要优于Picasso，速度更快，并且Glide更有利于减少OutOfMemoryError的发生，GIF动画是Glide的杀手锏。不过Picasso的图片质量更高。你更喜欢哪个呢？

虽然我使用了很长时间的Picasso，但是我得承认现在我更喜欢Glide。我的建议是使用Glide，但是将Bitmap格式换成 ARGB\_8888、让Glide缓存同时缓存全尺寸和改变尺寸两种。