Glide学习笔记

## 简介

Glide 支持拉取，解码和展示视频快照，图片，和GIF动画。Glide的Api是如此的灵活，开发者甚至可以插入和替换成自己喜爱的任何网络栈。默认情况下，Glide使用的是一个定制化的基于HttpUrlConnection的栈，但同时也提供了与Google Volley和Square OkHttp快速集成的工具库。

虽然Glide 的主要目标是让任何形式的图片列表的滚动尽可能地变得更快、更平滑，但实际上，Glide几乎能满足你对远程图片的拉取/缩放/显示的一切需求。

V4 学习文档：<https://blog.csdn.net/u013005791/article/details/74532091>

## 常用到的方法：

Glide.with(this)  
 .load("url")//要加载的图片的url  
 .priority(Priority.HIGH)//设置优先级  
 .skipMemoryCache(true)//设置是否使用缓存 true ：不使用缓存 false ：使用缓存（默认使用缓存）  
 .asBitmap()//设置只显示 GIF动画的静态图片  
 .asGif()//设置是GIF动画  
 .diskCacheStrategy(DiskCacheStrategy.NONE)//设置硬盘缓存 NONE 不使用  
 .error(R.drawable.ic\_launcher)//加载失败时显示的图片  
 .placeholder(R.drawable.ic\_launcher)//加载过程时的占位图片  
 .crossFade()//加载过程中显示的动画  
 .override(200, 200)//裁剪图片的宽高  
 .centerCrop()//设置图片以中心缩放  
 .fitCenter()//设置图片缩放充满布局  
 .into(imageView);//使用的ImageView

## 1、load()方法：

load():支持远程图片，本地图片文件，图片资源，多媒体数据库的uri

load(Uri uri)，load(File file)，load(Integer resourceId)，load(URL url)，load(byte[] model)，load(T model)，loadFromMediaStore(Uri uri)。

## 2、crossFade()有几种重载的方法

crossFade(int duration):设置时间  
crossFade(Animation animation, int duration):设置自定义的动画和时间  
crossFade(int animationId, int duration): 加载动画资源和时间

## 硬盘缓存策略

DiskCacheStrategy.NONE: 不使用硬盘缓存

DiskCacheStrategy.SOURCE: 将原始图像缓存在硬盘中

DiskCacheStrategy.RESULT: 将显示出来大小的图像缓存在硬盘(默认缓存策略)

DiskCacheStrategy.ALL: 显示的图像和原始图像都会缓存

## 4、优先级

Priority.LOW

Priority.NORMAL

Priority.HIGH

Priority.IMMEDIATE

## 清除内存缓存

// 必须在UI线程中调用

Glide.get(context).clearMemory()

## 清除磁盘缓存

// 必须在后台线程中调用，建议同时clearMemory()

Glide.get(applicationContext).clearDiskCache();

## 获取缓存大小

//getCacheDir()缓存路径

new GetDiskCacheSizeTask(textView).execute(new File(getCacheDir(), DiskCache.Factory.DEFAULT\_DISK\_CACHE\_DIR));

## 处理图片的库

<https://github.com/wasabeef/glide-transformations>

依赖：compile 'jp.wasabeef:glide-transformations:2.0.0'

//圆形裁剪  
Glide.with(this)  
 .load("url")  
 .bitmapTransform(new CropCircleTransformation(this))  
 .into(imageView);

//圆角处理  
Glide.with(this)  
 .load("url")  
 .bitmapTransform(new RoundedCornersTransformation(this,30,0, RoundedCornersTransformation.CornerType.ALL))  
 .into(imageView);

//灰度处理  
Glide.with(this)  
 .load("url")  
 .bitmapTransform(new GrayscaleTransformation(this))  
 .into(imageView);

## Glide使用时遇到的问题

1、ImageView的setTag问题   
问题描述：如果使用Glide的into(imageView)为ImageView设置图片的同时使用ImageView的setTag(final Object tag)方法，将会导致java.lang.IllegalArgumentException: You must not call setTag() on a view Glide is targeting异常。因为Glide的ViewTarget中通过view.setTag(tag)和view.getTag()标记请求的，由于Android 4.0之前Tag存储在静态map里，如果Glide使用setTag(int key, final Object tag)方法标记请求则可能会导致内存泄露，所以Glide默认使用view.setTag(tag)标记请求，你就不能重复调用了。   
解决办法：如果你需要为ImageView设置Tag，必须使用setTag(int key, final Object tag)及getTag(int key)方法，其中key必须是合法的资源ID以确保key的唯一性，典型做法就是在资源文件中声明type="id"的item资源。

2、placeholder()导致的图片变形问题   
问题描述：使用.placeholder()方法在某些情况下会导致图片显示的时候出现图片变形的情况。这是因为Glide默认开启的crossFade动画导致的TransitionDrawable绘制异常，详细描述和讨论可以看一下这个[#363](https://github.com/bumptech/glide/issues/363" \t "https://blog.csdn.net/shangmingchao/article/details/51125554/_blank) issue。根本原因就是你的placeholder图片和你要加载显示的图片宽高比不一样，而Android的TransitionDrawable无法很好地处理不同宽高比的过渡问题，这的确是个Bug，是Android的也是Glide的。   
解决办法：使用.dontAnimate()方法禁用过渡动画，或者使用animate()方法自己写动画，再或者自己修复TransitionDrawable的问题。

3、ImageView的资源回收问题   
问题描述：默认情况下，Glide会根据with()使用的Activity或Fragment的生命周期自动调整资源请求以及资源回收。但是如果有很占内存的Fragment或Activity不销毁而仅仅是隐藏视图，那么这些图片资源就没办法及时回收，即使是GC的时候。   
解决办法：可以考虑使用WeakReference，如：

final WeakReference<ImageView> imageViewWeakReference = new WeakReference<>(imageView);  
ImageView target = imageViewWeakReference.get();  
if (target != null) {  
 Glide.with(context).load(uri).into(target);  
}

1. ImageView的scaleType问题   
   scaleType默认为fitCenter模式，如果你想设置成centerInside，不好意思，3.x还没有这个方法，参见这个[#591](https://github.com/bumptech/glide/issues/591" \t "https://blog.csdn.net/shangmingchao/article/details/51125554/_blank) issue，折中的解决办法就是放弃使用centerInside，或者结合android:scaleType="centerInside"和.dontTransform()使用以禁止Glide对资源进行转换。   
   如果你想要ImageView的宽高根据图片资源的大小而定(即使用wrap\_comtent)，那么你就必须明确告诉Glide我想加载原始资源：使用android:scaleType="center"，或者.dontTransform()，或者.override(Target.SIZE\_ORIGINAL, Target.SIZE\_ORIGINAL)。   
   不推荐使用fitXY，因为这样Glide会加载全尺寸图像到内存中而造成不必要的内存占用。
2. 异步线程完成后加载图片的崩溃问题   
   问题描述：通常情况下，异步线程会被约束在Activity生命周期内，所以异步线程完成后使用Glide加载图片是没有问题的。但如果你的异步线程在Activity销毁时没有取消掉，那么异步线程完成后就Glide就无法为一个已销毁的Activity加载图片资源，抛出的异常如下（在with()方法中就进行判断并抛出异常）：
3. java.lang.IllegalArgumentException: You cannot start a load for a destroyed activity
4. at com.bumptech.glide.manager.RequestManagerRetriever.assertNotDestroyed(RequestManagerRetriever.java:134)
5. at com.bumptech.glide.manager.RequestManagerRetriever.get(RequestManagerRetriever.java:102)
6. at com.bumptech.glide.Glide.with(Glide.java:653)
7. at com.frank.glidedemo.TestActivity.onGetDataCompleted(TestActivity.java:23)
8. at com.frank.glidedemo.TestActivity.access$000(TestActivity.java:10)
9. at com.frank.glidedemo.TestActivity$BackgroundTask.onPostExecute(TestActivity.java:46)
10. at com.frank.glidedemo.TestActivity$BackgroundTask.onPostExecute(TestActivity.java:28)
11. at android.os.AsyncTask.finish(AsyncTask.java:632)
12. at android.os.AsyncTask.access$600(AsyncTask.java:177)
13. at android.os.AsyncTask$InternalHandler.handleMessage(AsyncTask.java:645)
14. at android.os.Handler.dispatchMessage(Handler.java:102)
15. at android.os.Looper.loop(Looper.java:157)
16. at android.app.ActivityThread.main(ActivityThread.java:5356)
17. at java.lang.reflect.Method.invokeNative(Native Method)
18. at java.lang.reflect.Method.invoke(Method.java:515)
19. at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:1265)
20. at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1081)
21. at dalvik.system.NativeStart.main(Native Method)
22. }

解决办法：正确管理Background Threads(异步线程)，当Activity停止或销毁时，停止所有相关的异步线程，停止所有后续的UI操作。或者加载前使用isFinishing()或isDestroyed()进行限制（不建议这种处理方式）。