# 赵彦军

## 做一只快乐的程序猿!!

博客园 首页 新随笔 联系 管理

随笔-185 文章-0 评论-61

昵称: 赵彦军 园龄: 2年7个月 粉丝: 58 关注: 9 +加关注

## android 图片加载库 Glide 的使用介绍

## 一: 简介

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 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。

## https://github.com/bumptech/glide

## 二: 使用

```
dependencies {
   compile 'com.github.bumptech.glide:glide:3.7.0'
}
```

如何查看最新版本

http://search.maven.org/#search%7Cga%7C1%7Ca%3A%22glide%22

#### 三: 使用方法及简介

http://mrfu.me/2016/02/27/Glide Getting Started/

## 四、清除缓存

```
1
 2
         * 清除缓存
 3
         * @param context
4
 5
        public void clearCache( final Context context ){
 6
           clearMemoryCache( context );
           new Thread(new Runnable() {
8
                @Override
9
                public void run() {
10
                    clearDiskCache( context );
11
12
           }).start();
        }
13
14
15
         * 清除内存缓存
16
         * @param context
17
18
19
        public void clearMemoryCache( Context context ){
20
           Glide.get( context ).clearMemory();
        }
21
22
23
         * 清除磁盘缓存
24
         * @param context
25
26
         */
27
        public void clearDiskCache( Context context ){
28
           Glide.get( context ).clearDiskCache();
29
```

## 我的标签

```
android(26)
赵彦军(8)
zhaoyanjun(7)
markdown(6)
android studio(6)
rxandroid(5)
rxbinding(5)
rxjava(5)
rxjava操作符(5)
线程(3)
更多
```

## **随笔分类**(292)

```
Android(144)
Android Studio(81)
Android 错误(4)
android 框架(6)
CVS ( SVN 和 Git )(18)
IOS(7)
IOS 错误(1)
Java(11)
产品与设计(5)
生活(3)
网站与服务器(10)
运营与测试(2)
```

## 随笔档案(185)

```
2016年9月 (8)
2016年8月 (7)
2016年7月 (4)
2016年6月 (9)
2016年5月 (8)
2016年4月 (12)
2016年3月 (20)
2016年2月 (8)
2016年1月 (9)
2015年12月 (5)
2015年11月 (6)
2015年10月 (6)
2015年9月 (5)
2015年8月 (12)
2015年7月 (15)
2015年6月 (31)
2015年5月 (19)
2015年4月 (1)
```

# 积分与排名

积分 - 55444 排名 - 4350

# 最新评论

#### 五、注意事项

5.1、在使用时需要给app添加联网权限,没有权限不会报错,但是图片加载不出来,很坑 爹。

5.2、加载图片的方法要写在 UI 线程中, Glide会做异步处理。

#### 六、使用方法总结

6.1 加载网络图片

```
imageView = (ImageView) findViewById( R.id.image );
String url = "http://img5.jpg" ;
Glide.with( this ).load( url ).into( imageView ) ;
```

6.2 加载网络图片监听(下载完成后显示)

```
package lib.com.myapplication;
 1
 2
     import android.graphics.Bitmap;
 3
     import android.os.Bundle;
 4
     import android.support.v7.app.AppCompatActivity;
     import android.widget.ImageView;
 5
 6
     import com.bumptech.glide.Glide;
 7
     import com.bumptech.glide.request.animation.GlideAnimation;
8
     import com.bumptech.glide.request.target.SimpleTarget;
 9
     public class MainActivity extends AppCompatActivity {
10
11
         private ImageView imageView ;
         String url = "http://img5.imgtn.bdimg.com/it/u=2941079711,2736454066&fm=11&gp=0.jpg";
12
13
         @Override
14
15
         protected void onCreate(Bundle savedInstanceState) {
             super.onCreate(savedInstanceState);
16
17
             setContentView(R.layout.activity_main);
             imageView = (ImageView) findViewById( R.id.image );
18
             Glide.with( this ).load( url ).asBitmap().into( target );
19
20
         }
21
22
         private SimpleTarget target = new SimpleTarget<Bitmap>() {
23
             @Override
24
             public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
                 //图片加载完成
25
26
                 imageView.setImageBitmap(bitmap);
27
             }
28
         };
29
```

特别注意: .asBitmap() 一定要加, 否则可能会出错

**6.3** 占位符 **placeholder()** 方法: 在加载开始 **--** 加载完成,这段时间显示的图片。如果加载失败,则最终显示占位符。

```
imageView = (ImageView) findViewById( R.id.image );
Glide.with( this ).load( url ).placeholder( R.drawable.user ).into( imageView ) ;
```

**6.4** 占位符 **error()** 方法: 在加载失败的情况下,显示的图片。

```
imageView = (ImageView) findViewById( R.id.image );
Glide.with( this ).load( url ).placeholder( R.drawable.user ).error( R.drawable.default_error )
```

在加载开始--> 加载完成(失败),显示placeholder()图片;如果加载失败,则显示error() 里面的图片。

```
1. Re:Android 应用程序集成Google 登录
及二次封装
```

@

哦哦,就是server\_client\_id啊,现在已经解决了,多谢楼主关注

----无人小道

- 2. Re:Android 应用程序集成Google 登录 及二次封装
- @-无人小道你说的配置是什么? ...

--赵彦军

3. Re:Android 应用程序集成Google 登录 及二次封装

问题已解决

---无人小道

4. Re:Android 应用程序集成Google 登录 及二次封装

大神,我照着你封装的接口走了一遍,但是遇到一个问题,配置的时候需要google\_app\_id和server\_client\_id两个是填写google 凭据里面对应项目的客户端ID中的android cl......

---无人小道

5. Re:Android Builder模式在开发中的应用

hennao

--fulai\_xy

# 阅读排行榜

- 1. Android Http请求框架二: xUtils 框架 网络请求(4474)
- 2. RxJava 和 RxAndroid 四(RxBinding 的使用)(3080)
- 3. RxJava 和 RxAndroid 三 (生命周期控制和内存优化) (2234)
- 4. android 显示 PDF 文件(2001)
- 5. RxJava 和 RxAndroid (基础)(189

# 推荐排行榜

- 1. Android studio 使用Gradle发布Android开源项目到JCenter 总结(5)
- 2. Android 数据库 LiteOrm 的使用(2)
- 3. Android 开关按钮切换,类似于iphone 效果, view实现(2)
- 4. 【转载】安卓APP架构(2)
- 5. subversion SVN(2)

#### 6.5 加载 drawable 里面的图片

```
imageView = (ImageView) findViewById( R.id.image );
   Glide.with( this ).load( R.drawable.icon ).asBitmap().into( imageView );
6.6 加载 SD 卡里面的一张图片 1 load( String string)
   imageView = (ImageView) findViewById( R.id.image );
   //sd卡上的一张图片
2
   String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg";
4 Glide.with( this ).load( path ).into( imageView );
6.7 加载 SD 卡里面的一张图片 2 load(File file)
   imageView = (ImageView) findViewById( R.id.image );
2
   //sd卡上的一张图片
3
   String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg" ;
   File file = new File( path ) ;
5 Glide.with( this ).load( file ).into( imageView );
6.8 加载 SD 卡里面的一张图片 3 load( Uri uri )
   //sd卡上的一张图片
1
2
   String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg" ;
   File file = new File( path ) ;
3
   Uri uri = Uri.fromFile( file ) ;
```

参数	说明
.load(String string)	string可以为一个文件路径、uri或者url
.load(Uri uri)	uri类型
.load(File file)	文件
.load(Integer resourceld)	资源ld,R.drawable.xxx或者R.mipmap.xxx
.load(byte[] model)	byte[]类型
.load(T model)	自定义类型

## **6.9 Glide.with()** 生命周期控制

Glide.with( this ).load( uri ).into( imageView ) ;

- 1. with(Context context). 使用Application上下文,Glide请求将不受Activity/Fragment生命周期控制。
- 2. with(Activity activity). 使用Activity作为上下文,Glide的请求会受到Activity生命周期控制。
- 3. with(FragmentActivity activity).Glide的请求会受到FragmentActivity生命周期控制。
- 4. with(android.app.Fragment fragment).Glide的请求会受到Fragment 生命周期控制。
- 5. with(android.support.v4.app.Fragment fragment).Glide的请求会受到 Fragment生命周期控制。
- 请求会在onStop的时候自动暂停,
- 在onStart的时候重新启动,gif的动画也会在onStop的时候停止,以免在后台消耗电量。

## **6.10** 加载优先级设置 **priority()**

枚举类 Priority 提供了几种优先级等级 ,

```
G Glide
    😘 🚡 GlideBuilder
                                      8
                                             public enum Priority
    😘 🚡 ListPreloader
                                      9
                                                  IMMEDIATE.
    ♠ ™ MemoryCategory
    🛂 🚡 Priority
                                     10
                                                  HIGH,
     😘 🖥 RequestManager
                                                  NORMAL,
                                     11
META-INF
                                     12
                                                  LOW, priority,
amcrest-core-1.3
                                     13
init-4.12
II toute 4.10 tou /01----- Lu
```

默认的是: NORMAL

```
784

<sup>™</sup> BitmapTypeRequest

    DownloadOptions

                              785
📭 o DrawableOptions
                              786
                                             private Request buildRequest(Target TranscodeType target) {
😭 🚡 DrawableRequestBuilder
                              787
                                                  if (priority == null) {
  DrawableTypeRequest
                              788
                                                      priority = Priority. NORMAL;
😘 🖰 GenericTranscodeRequest
                              789
😘 🖰 GifRequestBuilder
                              790
                                                  return buildRequestRecursive(target, null);
😭 🖥 GifTypeRequest
Ga a Glide
                              791
G GlideBuilder
```

实例:

```
Glide.with( this).load( url2).priority(Priority.LOW ).into( imageView2 );
Glide.with( this).load( url3).priority(Priority.HIGH ).into( imageView3 );
```

但是这里的优先级只是在加载的过程中起一个参考作用, 并不决定真正的加载顺序。

#### 6.11 缩略图的支持

(一) 先加载原图的十分之一作为缩略图,再加载原图

```
1 | Glide.with(thi).load(url).thumbnail(0.1f).into(imageview);
(二)用本地的图片作为缩略图,然后再加载原图
```

```
DrawableRequestBuilder<Integer> thumbnailRequest = Glide

uith( ThumbnailActivity.this )

load(R.mipmap.ic_launcher);

Glide.with( ThumbnailActivity.this ).load( ur2 ).thumbnail( thumbnailRequest ).into( imageView2
```

## 6.12 加载 Gif 动图

```
1 | Glide.with( this ).load( url ).into( imageView1 );
2 | Glide.with( this ).load( url ).asGif().into( imageView2 );
```

注意: 如果把asGif 换成 asBitmap 则会显示一张静态图。

## 6.13 加载本地视频,相当于一张缩略图

```
imageView = (ImageView) findViewById( R.id.image_video );

String files = Environment.getExternalStorageDirectory().getAbsolutePath() + "/yueyu.mkv";

Glide.with( this ).load( files ).into( imageView );
```

- (1) 只能加载本地视频, 网络视频无法加载。
- (2) 加载本地视频显示只是视频的第一帧图像,相当于一张缩略图。不能播放视频。

## 6.14 加载动画

- .crossFade() 淡入淡出, 也是默认动画
- .crossFade(int duration) 定义淡入淡出的时间间隔
- .dontAnimate() 不使用任何动画

## **6.15** glide 内存缓存

glide 默认启用内存缓存,如果想要禁止内存缓存,使用 .skipMemoryCache(true)

## 七、自定义 GlideModule

自定义 GlideModule 的好处:

- 1、可以全局的改变 glide 的加载策略
- 2、可以自定义磁盘缓存目录
- 3、可以设置图片加载的质量

## 7.1 首先定义一个类实现 GlideModule

```
1
     public class SimpleGlideModule implements GlideModule {
 2
3
        @Override
        public void applyOptions(Context context, GlideBuilder builder) {
4
 5
        @Override
8
         public void registerComponents(Context context, Glide glide) {
9
10
11
         }
12
    }
```

可以看到重写了两个方法,applyOptions(),registerComponents().两个方法都没有返回值。 我们着重于第一个方法,重点研究 GlideBuilder 。

## 7.2 然后在 AndroidManifest.xml 去申明你写的 SimpleGlideModule

## 7.3 GlideBuilder

- .setMemoryCache(MemoryCache memoryCache)
- .setBitmapPool(BitmapPool bitmapPool)
- .setDiskCache(DiskCache.Factory diskCacheFactory)
- .setDiskCacheService(ExecutorService service)
- .setResizeService(ExecutorService service)
- .setDecodeFormat(DecodeFormat)

```
public class SimpleGlideModule implements GlideModule {
   @Override
   public void applyOptions(Context context, GlideBuilder builder) {
          builder.
              • setBitmapPool (BitmapPool bitmapPool)
                                                        GlideBuild
              ■ setDecodeFormat (DecodeFormat de…
                                                     GlideBuilder
   @Override

■ setDiskCache (Factory diskCacheFa… GlideBuilder)

   public voi 👦 🚡 setDiskCacheService (ExecutorSer… GlideBuilder
              GlideBuilder

    setMemorvCache (MemorvCache memo···

                                                     GlideBuilder

■ setResizeService (ExecutorServic…

                                                     GlideBuilder

<sup>™</sup> equals (Object o)

                                                          hoolean

    hashCode ()
```

可以看到 setBitmapPool() 是设置bitmap池的; setDecodeFormat() 是设置解码方式的; setDiskCache() 是设置磁盘缓存的;

#### 7.4 DecodeFormat

Android里有两个方法去解析图片: ARGB8888和RGB565。第一个为每个像素采用4 byte 表示,后面一个则用2 byte表示。ARG8888有更高的图片质量,并且能够存储一个alpha通道。Glide默认使用低质量的RGB565。你可以通过使用Glide module方法改变解析格式。

```
📭 DecodeFormat.java 🗴
30
                  {@link android.graphics.BitmapFactory} does not allow us to guaran
                 be of a requested config without resorting to expensive copying. A
31
                  Most GIFs, for example, will still produce {@link android.graphic.
32
33
                  {@link android.graphics.Bitmap.Config}s.
34
35
          PREFER_ARGB_8888,
36
37
38
39
           * Bitmaps decoded from image formats that support and/or use alpha (some
           * return {@link android.graphics.Bitmap.Config#ARGB_8888} for {@link an
40
           * decoded from formats that don't support or use alpha should return
41
           * {@link android.graphics.Bitmap.Config#RGB_565} for {@link android.gra
42
43
44
           */
          PREFER RGB 565;
45
46
47
          /** The default value for DecodeFormat.
          public static final DecodeFormat | DEFAULT = PREFER_RGB_565;
48
49
```

## 最后一个完整的自定义glideModule

```
//自定义缓存 路径 和 缓存大小
8
9
            String diskCachePath = Environment.getExternalStorageDirectory().getAbsolutePath() + "
10
11
12
            builder.setDecodeFormat(DecodeFormat.PREFER_ARGB_8888);
13
            //自定义磁盘缓存:这种缓存只有自己的app才能访问到
14
            // builder.setDiskCache( new InternalCacheDiskCacheFactory( context , diskCacheSize ))
15
            // builder.setDiskCache( new InternalCacheDiskCacheFactory( context , diskCachePath ,
16
17
            //自定义磁盘缓存: 这种缓存存在SD卡上, 所有的应用都可以访问到
18
19
            builder.setDiskCache(new DiskLruCacheFactory( diskCachePath , diskCacheSize ));
20
21
        }
22
23
        @Override
24
        public void registerComponents(Context context, Glide glide) {
25
26
        }
27
    }
```

## 八、缓存管理

## 7.1、默认缓存目录和缓存大小

在Glide源码中有一个DiskCache接口,里面的Factory类定义了默认的磁盘缓存大小为: 250~M,缓存路径在: image\_manager\_disk\_cache 目录下



在模拟器上可以查看缓存目录的位置,有些真机看不到这个目录,有可能数据库没有刷新的原因:



#### 7.2、缓存模式

源码中有枚举类 DiskCacheStrategy 定义了四种缓存类型

- DiskCacheStrategy.SOURCE 缓存原图
- DiskCacheStrategy.RESULT 缓存和imageview大小匹配的图
- DiskCacheStrategy.ALL 既缓存原图,有缓存和imageview大小匹配的图
- DiskCacheStrategy.NONE 不做任何缓存

```
*/

**Set of available caching strategies for media.

**

**public enum DiskCacheStrategy {

    /** Caches with both {@link #SOURCE} and {@link #RESULT}. *,

    ALL(true, true),

    /** Saves no data to cache. */

    NONE(false, false),

    /** Saves just the original data to cache. */

    SOURCE(true, false),

    /** Saves the media item after all transformations to cache.

    RESULT(false, true);

private final boolean cacheSource;

private final boolean cacheResult;
```

通过查看源码我们发现,Glide默认缓存策略是: DiskCacheStrategy.RESULT

```
private boolean isCacheable = true:

    DownloadOptions
    DrawableOptions

                       73
                                 private GlideAnimationFactory(TranscodeType) animationFactory = NoAnimation. setFactory(
    😭 a DrawableRequestBuilder
                       74
                                 private int overrideHeight = -1;
    n DrawableTypeRequest
                                private int overrideWidth = -1;
                       75
   Ga a GenericRequestBuilder
                               private DiskCacheStrategy diskCacheStrategy = DiskCacheStrategy.RESULT;
                       76
    G a GenericTranscodeRequest
G a GifRequestBuilder
                       77
                                 private Transformation<ResourceType> transformation = UnitTransformation.get();

<sup>™</sup> GifTypeRequest

                                 private boolean isTransformationSet;
    Cara Glida
       7.3 四种缓存模式对比
           比如: 网络图片我们叫做 big1.jpg 宽高: 3000 x 2000 大小: 2.15 M 。
   1 http://o7rvuansr.bkt.clouddn.com/big1.jpg
          客户端的 imageview 大小: 300 x 300
   1
       <ImageView</pre>
          android:id="@+id/image"
   2
   3
          android:layout_width="300dp"
   4
          android:layout_height="300dp" />
           (1) DiskCacheStrategy.SOURCE : 只会缓存一张图片, 大小: 2M
          Glide. with (Activity2. this). load (url ). diskCacheStrategy (
DiskCacheStrategy.SOURCE ).into( imageView ) ;
           (2) DiskCacheStrategy.RESULT: 只缓存了一张图片, 大小: 14 KB
          Glide.with(Activity2.this).load(url).diskCacheStrategy(
DiskCacheStrategy.RESULT ).into( imageView ) ;
           (3) DiskCacheStrategy.ALL :缓存了两张图片,一张大小: 2 M , 一张大小: 14 KB
          Glide. with (Activity2. this). load (url
).diskCacheStrategy(DiskCacheStrategy.ALL).into(imageView);
           (4) DiskCacheStrategy.NONE: 没有缓存图片
              Glide.with( Activity2.this).load( url
```

## 7.4、缓存场景测试

两个imageView ,一个 100 x 100 , 一个 300 x 300 ; 先加载第一张,再加载第二张

• 测试一: 两张图片都在 **DiskCacheStrategy.SOURCE** 的情况下:

).diskCacheStrategy(DiskCacheStrategy.NONE ).into( imageView );

```
//加载第一张图
 1
           findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
 2
 3
               @Override
 4
              public void onClick(View v) {
 5
                   Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOU
          });
 8
          //加载第二张图
9
10
          findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
               @Override
11
              public void onClick(View v) {
12
                  Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOU
13
14
15
          });
```

通过测试发现,在加载第一张图片的时候,缓存了2M 的原始图,在加载第二张的时候,就不会再请求网络,直接从缓存中加载。

• 测试二:第一张图在 DiskCacheStrategy.SOURCE,第二张在DiskCacheStrategy.RESULT 情况下

```
1 //加载第一张图
2 findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
3 @Override
4 public void onClick(View v) {
```

```
Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.S
 5
 6
                 }
 7
             });
 8
             //加载第二张图
 9
             findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
10
11
12
                 public void onClick(View v) {
13
                     Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.R
14
                 }
15
             });
4
```

通过测试发现,加载第一张图片的情况下,缓存了2M的原始图。在加载第二张图片的时候,又请求网络,下载了14 KB的缓存。这说明,即使本地存在缓存,缓存策略不一样,缓存就不会被重用。

- 测试三: 两张图都在 DiskCacheStrategy.RESULT 情况下,第一个Imageview大小: 100 x100 ,第二个imageview大小: 300 x 300
- (1) 先加载  $100 \times 100$ , 加载出来后,缓存了  $2 \times 100$  的图片,然后加载  $100 \times 100$  ,又重新请求网络,缓存了  $14 \times 100$  的图片。缓存没有复用。
- (2) 先加载  $300 \times 300$ , 加载出来后,缓存了  $14 \times 100$  KB的图片。然后加载  $100 \times 100$ , 又重新请求网络,缓存了  $2 \times 100$  KB的图片,缓存没有复用。

```
1
     //加载第一张图
 2
         findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
 3
             @Override
 4
             public void onClick(View v) {
 5
                 Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.RESUL
 6
             }
 7
         });
 8
 9
         //加载第二张图
10
         findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
             @Override
11
             public void onClick(View v) {
12
13
                 Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.RESUL
14
             }
15
         });
4
```

• 测试四: imageView1 宽高: 100 x 100 ; imageView2 宽高: 300 x 300 ; imageView 3 宽高: 600 x 600

```
//加载第二张图
 1
    findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
 2
 3
        public void onClick(View v) {
 4
 5
            Glide.with( Activity2.this).load( url ).into( imageView1 ) ;
 6
        }
 7
8
    });
9
10
     //加载第二张图
11
     findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
12
        @Override
13
        public void onClick(View v) {
14
            Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.ALL ).int
15
         }
    });
16
17
18
     //加载第三张图
19
     findViewById( R.id.bt3 ).setOnClickListener(new View.OnClickListener() {
        @Override
20
21
        public void onClick(View v) {
22
            Glide.with( Activity2.this).load( url ).into( imageView3 ) ;
```

```
23 }
24 });
```

先加载 imageView2 ,请求网络, 缓存策略是 DiskCacheStrategy.ALL , 缓存了两张图 , 一张是 2~M , 一张是 14~kB 。

再加载 imageView1,请求网络,缓存策略是 DiskCacheStrategy.RESULT,缓存了一张图,2kB。可见 imageview2 的缓存对image1不起作用。

最后加载 imageView3,请求网络,缓存策略是 DiskCacheStrategy.RESULT,缓存了一张图, 47 KB,可见 imageview1 和 imageview2的缓存对imageView3 不起作用

- 测试五: SimpleTarget 的使用 , imageview 大小: 300 x 300 ; 网络图片 宽高: 3000 x 2000 大小: 2.15 M;
  - (1) 这种情况的缓存 2 M

```
1 Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOURCE ).into( ima
```

(2) 这种情况的缓存 14 KB

```
1 Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.RESULT ).into( ima
```

(3) 这种情况的缓存 777 KB

```
1
    package app.zuil.com.glidedemo;
    import android.graphics.Bitmap;
    import android.os.Bundle;
    import android.support.v7.app.AppCompatActivity;
    import android.view.View;
    import android.widget.ImageView:
     import com.bumptech.glide.Glide;
8
     import com.bumptech.glide.load.engine.DiskCacheStrategy;
     import com.bumptech.glide.request.animation.GlideAnimation;
9
     import com.bumptech.glide.request.target.SimpleTarget;
10
11
12
     public class Activity2 extends AppCompatActivity {
13
14
        String url = "http://o7rvuansr.bkt.clouddn.com/big1.jpg" ;
         ImageView imageView2;
15
16
17
         private SimpleTarget target ;
18
19
         @Override
20
         protected void onCreate(Bundle savedInstanceState) {
21
             super.onCreate(savedInstanceState);
22
             setContentView(R.layout.activity1);
23
24
             imageView2 = (ImageView) findViewById( R.id.image2 );
25
26
             //加载第二张图
27
             findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
28
                 @Override
29
                 public void onClick(View v) {
                     Glide.with( Activity2.this).load( url ).asBitmap().diskCacheStrategy( DiskCach
30
31
32
             });
33
             target = new SimpleTarget<Bitmap>() {
35
                 @Override
                 public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
36
37
                     //图片加载完成
                     imageView2.setImageBitmap( bitmap );
38
39
40
             };
41
         }
42
    }
```

(4) 这种缓存 67 KB

```
1
     package app.zuil.com.glidedemo;
 2
 3
     import android.graphics.Bitmap;
4
     import android.os.Bundle;
 5
     import android.support.v7.app.AppCompatActivity;
 6
     import android.view.View;
 7
     import android.view.ViewGroup;
     import android.widget.ImageView;
9
10
     import com.bumptech.glide.Glide;
11
     import com.bumptech.glide.load.engine.DiskCacheStrategy;
12
     import com.bumptech.glide.request.animation.GlideAnimation;
13
     import com.bumptech.glide.request.target.SimpleTarget;
14
15
     public class Activity2 extends AppCompatActivity {
16
17
        String url = "http://o7rvuansr.bkt.clouddn.com/big1.jpg" ;
18
         ImageView imageView2 ;
19
         private static int width ;
20
         private static int height ;
21
        private SimpleTarget target ;
22
23
         private ViewGroup.LayoutParams params ;
25
        @Override
        protected void onCreate(Bundle savedInstanceState) {
26
27
             super.onCreate(savedInstanceState);
28
             setContentView(R.layout.activity1);
29
30
             imageView2 = (ImageView) findViewById( R.id.image2 );
31
             params = imageView2.getLayoutParams();
32
             //测量图片的宽高
33
             width = params.width ;
34
             height = params.height ;
35
36
37
             //加载第二张图
             findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
38
39
                 @Override
40
                 public void onClick(View v) {
41
                     Glide.with( Activity2.this).load( url ).asBitmap().diskCacheStrategy( DiskCach
42
                 }
             });
43
44
45
             target = new SimpleTarget<Bitmap>( width , height ) {
46
47
                 public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
48
                     //图片加载完成
                     imageView2.setImageBitmap(bitmap);
49
50
                 }
51
             };
52
         }
53
    }
```

• 测试六:以上的测试都是在两个imageView 大小不一样的情况下。现在来测试 两个imageview大小相同的时候。imageview1,宽高: 400 x 400; imageview2,宽高: 400 x 400

```
1 Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.ALL ).into( imageV
2 Glide.with( Activity2.this).load( url ).into( imageView2 );
```

4

- (1) 分析 imageView1,采用 DiskCacheStrategy.ALL: imageView2,采用默认的缓存策略,也就是DiskCacheStrategy.RESULT。
- (2) 先加载 imageview1, 缓存了2个文件,一个是2 M,一个事 22 KB; 再加载 imageview2,发现没有请求网络, 直接使用了imageview1 的缓存。
- (3) 先加载 imageview2 ,缓存了1个文件,22 KB;然后再加载 imageview1 ,发现没有请求网络,直接使用imageview2 的缓存。而且在缓存文件夹中一直只有1个22 KB 缓存,这时候我们就发现,即使 imageview1 的缓存模式是 DiskCacheStrategy.ALL ,在这里似乎缓存原图的功能失效了。
  - 测试七: imageview1, 宽高: 400 x 400; imageview2, 宽高: 400 x 400
- Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOURCE ).into( ima Glide.with( Activity2.this).load( url ).into( imageView2 );

(1) 先加载 imageview1,缓存了1个文件, 2M; 然后再加载 imageview2,发现没有请求网络,直接使用 imageview1 的缓存。两个图片完全加载出来,缓存文件夹一共只有1个文件, 2M;

(2) 先加载 Imageview2,缓存了 1个文件, 22 KB; 然后再加载 imageview1,发现没有请求网咯, 直接用 imageview2 的缓存。两个图片完全加载出来,缓存文件夹一共只有 1 个文件 , 22 KB。

博客中的代码示例都在 https://github.com/zyj1609wz/GlideDemo

## 分类: Android, Android Studio

标签: <u>赵彦军</u>, <u>zhaoyanjun</u>, <u>android</u>, <u>google</u>, <u>Glide</u>, <u>图片加载</u>, <u>android 图片加载</u>, <u>android glide</u>





0

## +加关注

- « 上一篇: 如何处理 android 方法总数超过 65536 . the number of method references in a .dex file exceed 64k
- » 下一篇: Android 视频播放器, 在线播放

posted @ 2016-04-18 13:43 赵彦军 阅读(592) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

0

注册用户登录后才能发表评论,请 <u>登录</u> 或 <u>注册</u>,<u>访问</u>网站首页。

- 【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【活动】优达学城正式发布"无人驾驶车工程师"课程
- 【推荐】移动直播百强八成都在用融云即时通讯云
- 【推荐】别再闷头写代码!找对工具,事半功倍,全能开发工具包用起来



#### 最新IT新闻:

- · Google开源SLAM地图软件:用于机器人、无人驾驶、无人机
- ·春雨医生正式发布讣告: CEO张锐去世 业务暂交由联合创始人
- ·春雨医生创始人张锐因病意外辞世 他曾说创业是一次18岁的远行
- · Google收购一年后: Pixate Studio宣布将于10月31号被终结

- · 重新专注于研发: Theranos将关闭实验室和解雇340名员工
- » 更多新闻...



90%的开发者选择极光推送 不仅是集成简单、24小时一对一技术支持

## 最新知识库文章:

- · 陈皓: 什么是工程师文化?
- ·没那么难,谈CSS的设计模式
- ·程序猿媳妇儿注意事项
- ·可是姑娘,你为什么要编程呢?
- · 知其所以然(以算法学习为例)
- » 更多知识库文章...

Copyright ©2016 赵彦军