赵彦军

做一只快乐的程序猿！！

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

```
/**
    * 清除缓存
    * @param context
    */
   public void clearCache( final Context context ){
       clearMemoryCache( context );
       new Thread(new Runnable() {
           @Override
           public void run() {
               clearDiskCache(  context );
           }
       }).start();
   }

   /**
    * 清除内存缓存
    * @param context
    */
   public void clearMemoryCache( Context context ){
       Glide.get( context ).clearMemory();
   }

   /**
    * 清除磁盘缓存
    * @param context
    */
   public void clearDiskCache( Context context ){
       Glide.get( context ).clearDiskCache();
   }
```

昵称：赵彦军
园龄：2年7个月
粉丝：58
关注：9
+加关注

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。

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

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;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.ImageView;
import com.bumptech.glide.Glide;
import com.bumptech.glide.request.animation.GlideAnimation;
import com.bumptech.glide.request.target.SimpleTarget;

public class MainActivity extends AppCompatActivity {
    private ImageView imageView ;
    String url = "http://img5.imgtn.bdimg.com/it/u=2941079711,2736454066&fm=11&gp=0.jpg" ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = (ImageView) findViewById( R.id.image );
        Glide.with( this ).load( url ).asBitmap().into( target ) ;
    }

    private SimpleTarget target = new SimpleTarget<Bitmap>() {
        @Override
        public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
            //图片加载完成
            imageView.setImageBitmap(bitmap);
        }
    };
}
```

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。

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
String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg" ;
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 );
 //sd卡上的一张图片
String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg" ;
File file = new File( path ) ;
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
String path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp.jpg" ;
File file = new File( path ) ;
Uri uri = Uri.fromFile( file ) ;
Glide.with( this ).load( uri ).into( imageView ) ;
```

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 resourceId)	资源Id,R.drawable.xxx或者R.mipmap.xxx
.load(byte[] model)	byte[]类型
.load(T model)	自定义类型

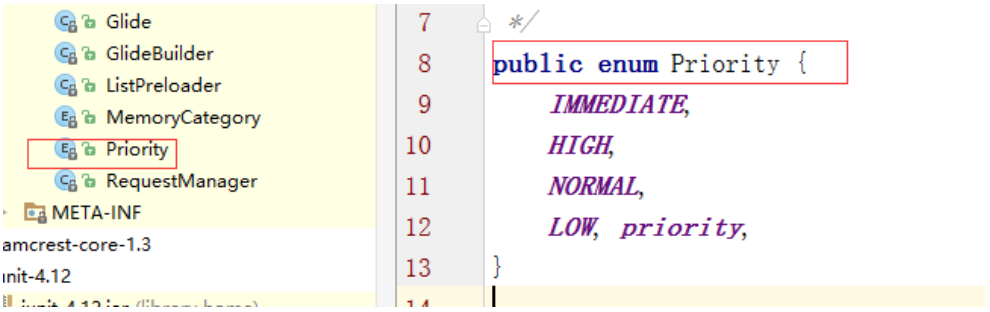
6.9 Glide.with() 生命周期控制

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。

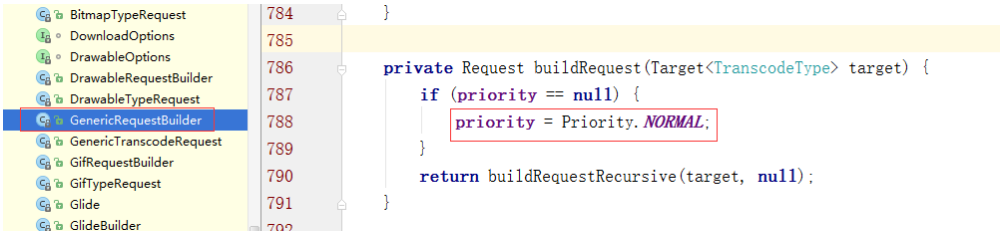
- 请求会在onStop的时候自动暂停，
- 在onStart的时候重新启动,gif的动画也会在onStop的时候停止，以免在后台消耗电量。

6.10 加载优先级设置 priority()

枚举类 Priority 提供了几种优先级等级 ，



默认的是 ： NORMAL



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

```
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
                    .with( ThumbnailActivity.this )
                    .load(R.mipmap.ic_launcher);
Glide.with( ThumbnailActivity.this ).load( ur2 ).thumbnail( thumbnailRequest ).into( imageView2
```

6.12 加载 Gif 动图

```
Glide.with( this ).load( url ).into( imageView1 ) ;
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

```
public class SimpleGlideModule implements GlideModule {

    @Override
    public void applyOptions(Context context, GlideBuilder builder) {

    }

    @Override
    public void registerComponents(Context context, Glide glide) {

    }
}
```

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 。

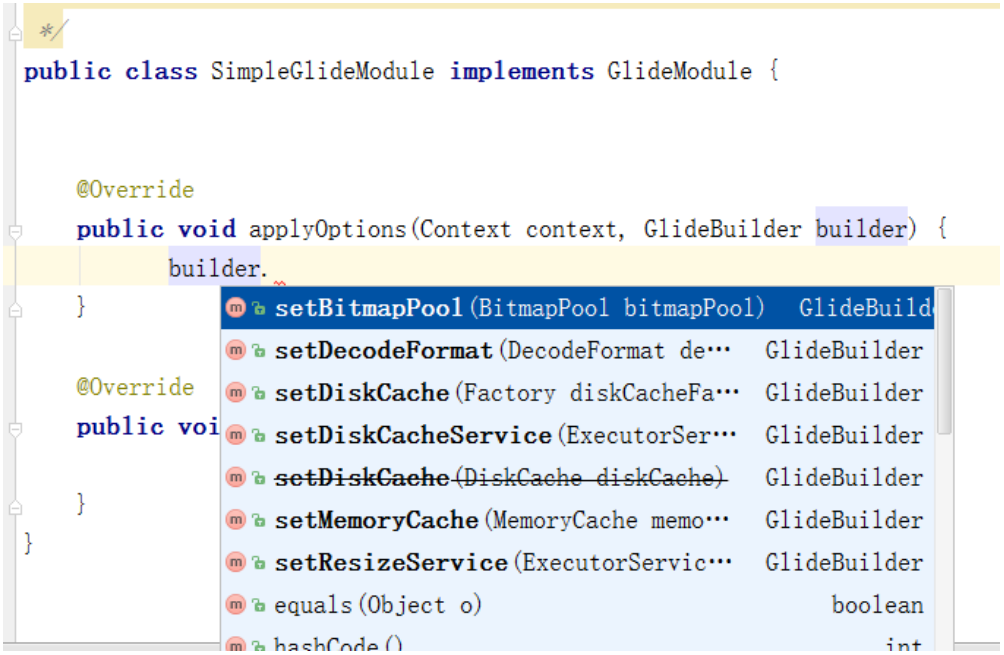
7.2 然后在 AndroidManifest.xml 去申明你写的 SimpleGlideModule

```
<meta-data
    android:name="app.zuil.com.glidedemo.util.SimpleGlideModule"
    android:value="GlideModule" />
```

name是：包名 + 类名

7.3 GlideBuilder

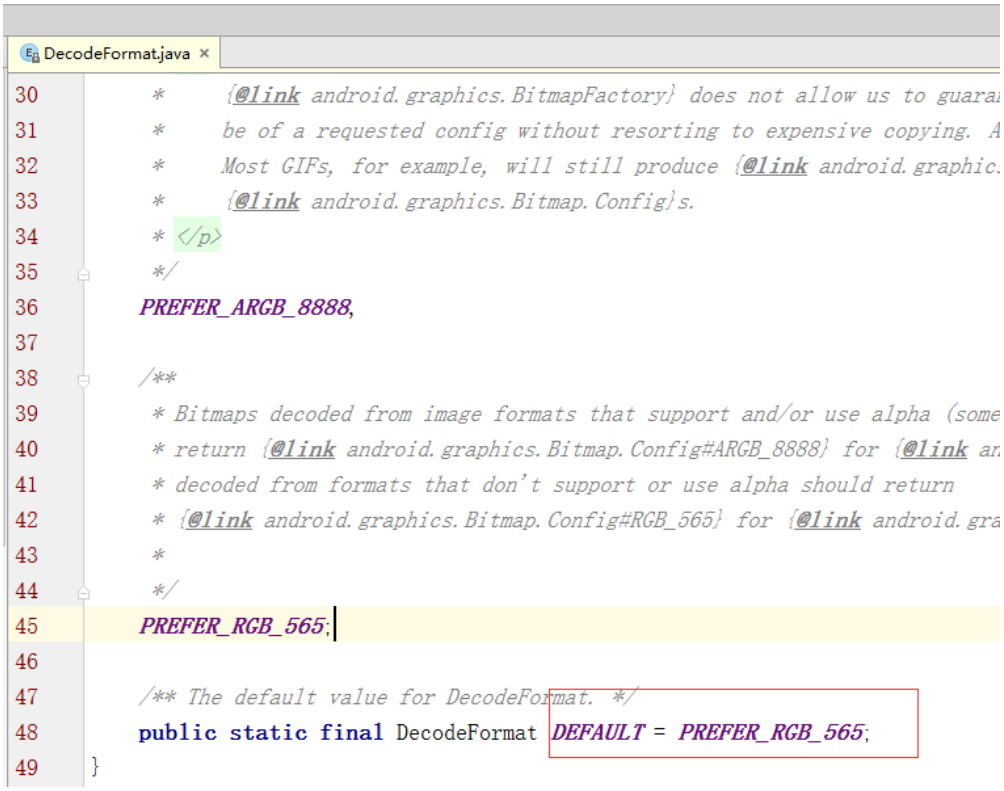
- .setMemoryCache(MemoryCache memoryCache)
- .setBitmapPool(BitmapPool bitmapPool)
- .setDiskCache(DiskCache.Factory diskCacheFactory)
- .setDiskCacheService(ExecutorService service)
- .setResizeService(ExecutorService service)
- .setDecodeFormat(DecodeFormat decodeFormat)



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。



最后一个完整的自定义**glideModule**

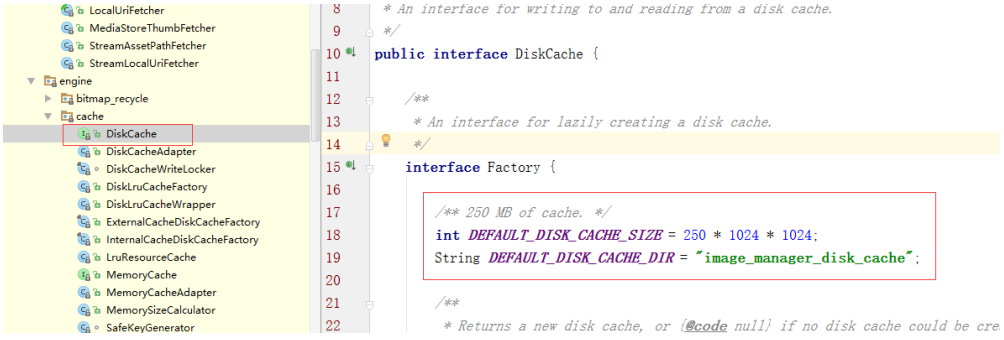
```
public class SimpleGlideModule implements GlideModule {

    @Override
    public void applyOptions(Context context, GlideBuilder builder) {
        //定义缓存大小为100M
        int  diskCacheSize =  100 * 1024 * 1024;

```

```
        //自定义缓存 路径 和 缓存大小
        String diskCachePath = Environment.getExternalStorageDirectory().getAbsolutePath() + "

        //提高图片质量
        builder.setDecodeFormat(DecodeFormat.PREFER_ARGB_8888);

        //自定义磁盘缓存:这种缓存只有自己的app才能访问到
        // builder.setDiskCache( new InternalCacheDiskCacheFactory( context , diskCacheSize ))
        // builder.setDiskCache( new InternalCacheDiskCacheFactory( context , diskCachePath ,

        //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
        builder.setDiskCache(new DiskLruCacheFactory( diskCachePath , diskCacheSize ));

    }

    @Override
    public void registerComponents(Context context, Glide glide) {

    }
}
```

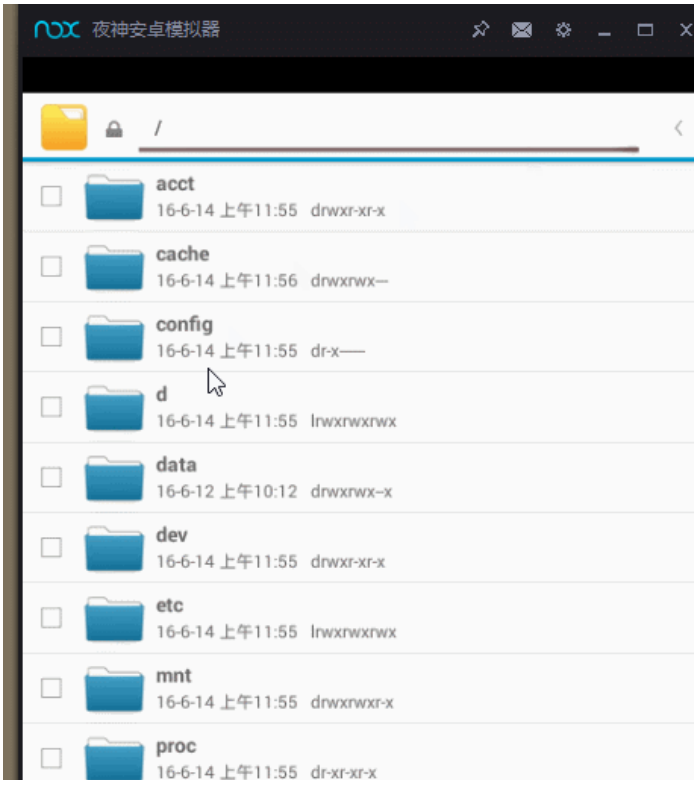
八、缓存管理

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：



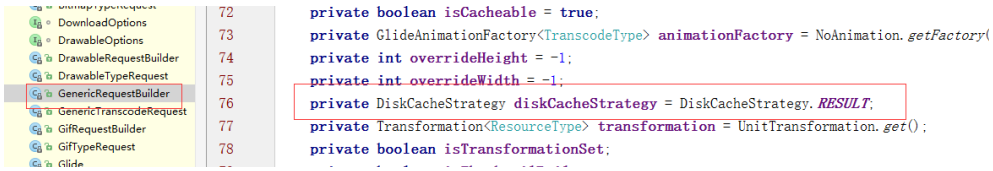
7.2、缓存模式

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**



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 。

```
http://o7rvuansr.bkt.clouddn.com/big1.jpg
```

客户端的 imageview 大小：300 x 300

```
<ImageView
   android:id="@+id/image"
   android:layout_width="300dp"
   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).diskCacheStrategy(DiskCacheStrategy.NONE).into(imageView) ;

7.4、 缓存场景测试

两个imageView ,一个 100 x 100 , 一个 300 x300 ； 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：

```
//加载第一张图
      findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
              Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOU
          }
      });

      //加载第二张图
      findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
              Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.SOU
          }
      });
```

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

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

```
                Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.S
            }
        });

        //加载第二张图
        findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.R
            }
        });
```

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 x 100 ,** 加载出来后，缓存了 **2 kB** 的图片，然后加载 **300 x 300**，又重新请求网络，缓存了 **14 KB** 的图片 。缓存没有复用。

（**2**）先加载 **300 x 300 ,** 加载出来后，缓存了 **14 KB**的图片。然后加载 **100 x 100**，又重新请求网络，缓存了 **2 KB** 的图片，缓存没有复用。

```
//加载第一张图
    findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.RESUL
        }
    });

    //加载第二张图
    findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.RESUL
        }
    });
```

- 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
findViewById( R.id.bt1 ).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Glide.with( Activity2.this).load( url ).into( imageView1 ) ;
    }

});

//加载第二张图
findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Glide.with( Activity2.this).load( url ).diskCacheStrategy( DiskCacheStrategy.ALL ).int
    }
});

//加载第三张图
findViewById( R.id.bt3 ).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Glide.with( Activity2.this).load( url ).into( imageView3 ) ;
```

```
    }
});
```

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，2 kB 。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

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

（2）这种情况的缓存 14 KB

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

（3）这种情况的缓存 777 KB

```
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;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.bumptech.glide.request.animation.GlideAnimation;
import com.bumptech.glide.request.target.SimpleTarget;

public class Activity2 extends AppCompatActivity {

    String url = "http://o7rvuansr.bkt.clouddn.com/big1.jpg" ;
    ImageView  imageView2 ;

    private SimpleTarget target ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity1);

        imageView2 = (ImageView) findViewById( R.id.image2 );

        //加载第二张图
        findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Glide.with( Activity2.this).load( url ).asBitmap().diskCacheStrategy( DiskCach
            }
        });

        target = new SimpleTarget<Bitmap>() {
            @Override
            public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
                //图片加载完成
                imageView2.setImageBitmap( bitmap );
            }
        };
    }
}
```

（4）这种缓存 67 KB

```
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.view.ViewGroup;
import android.widget.ImageView;

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.bumptech.glide.request.animation.GlideAnimation;
import com.bumptech.glide.request.target.SimpleTarget;

public class Activity2 extends AppCompatActivity {

    String url = "http://o7rvuansr.bkt.clouddn.com/big1.jpg" ;
    ImageView imageView2 ;
    private static int width  ;
    private static int height ;

    private SimpleTarget target ;
    private ViewGroup.LayoutParams params ;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity1);

        imageView2 = (ImageView) findViewById( R.id.image2 );
        params = imageView2.getLayoutParams() ;

        //测量图片的宽高
        width = params.width ;
        height = params.height ;

        //加载第二张图
        findViewById( R.id.bt2 ).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Glide.with( Activity2.this).load( url ).asBitmap().diskCacheStrategy( DiskCach
            }
        });

        target = new SimpleTarget<Bitmap>( width , height ) {
            @Override
            public void onResourceReady(Bitmap bitmap, GlideAnimation glideAnimation) {
                //图片加载完成
                imageView2.setImageBitmap(bitmap);
            }
        };
    }
}
```

- 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**

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

（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，发现没有请求网络， 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: 赵彦军, zhaoyanjun, android, google, Glide, 图片加载, android 图片加载, android glide

好文要顶 关注我 收藏该文

赵彦军
关注 - 9
粉丝 - 58
+加关注

0 0

« 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

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

· 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...

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...