picasso-强大的Android图片下载缓存库

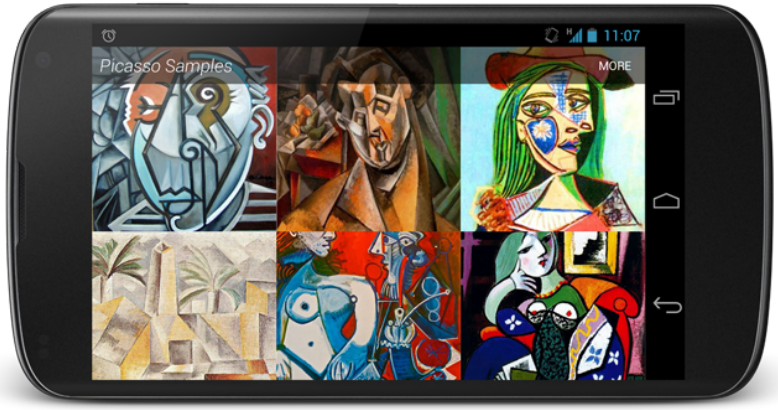
泡在网上的日子 / 文 发表于2014-07-31 10:06 第124508次阅读 android (/tags.php?/android/),开源 (/tags.php?/开源/),异步加载 (/tags.php?/异步加载/)

编辑推荐：稀土掘金 (https://juejin.im/)，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！

picasso是Square公司开源的一个Android图形缓存库，地址http://square.github.io/picasso/ (http://square.github.io/picasso/)，可以实现图片下载和缓存功能。仅仅只需要一行代码就能完全实现图片的异步加载：

```
1.  Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

Api看起来非常独特，是吧。



Picasso不仅实现了图片异步加载的功能，还解决了android中加载图片时需要解决的一些常见问题：

1.在adapter中需要取消已经不在视野范围的ImageView图片资源的加载，否则会导致图片错位，Picasso已经解决了这个问题。

2.使用复杂的图片压缩转换来尽可能的减少内存消耗

3.自带内存和硬盘二级缓存功能

特性以及示例代码：

ADAPTER 中的下载：Adapter的重用会被自动检测到，Picasso会取消上次的加载

```
1.  @Override public void getView(int position, View convertView, ViewGroup parent
2.    SquaredImageView view = (SquaredImageView) convertView;
3.    if (view == null) {
4.      view = new SquaredImageView(context);
5.    }
6.    String url = getItem(position);
7.    Picasso.with(context).load(url).into(view);
8.  }
```

图片转换：转换图片以适应布局大小并减少内存占用

```
Picasso.with(context)
  .load(url)
  .resize(50, 50)
  .centerCrop()
  .into(imageView);
```

你还可以自定义转换：

```
public class CropSquareTransformation implements Transformation {
  @Override public Bitmap transform(Bitmap source) {
    int size = Math.min(source.getWidth(), source.getHeight());
    int x = (source.getWidth() - size) / 2;
    int y = (source.getHeight() - size) / 2;
    Bitmap result = Bitmap.createBitmap(source, x, y, size, size);
    if (result != source) {
      source.recycle();
    }
    return result;
  }
  @Override public String key() { return "square()"; }
}
```

将CropSquareTransformation 的对象传递给**transform** 方法即可。

Place holders-空白或者错误占位图片：picasso提供了两种占位图片，未加载完成或者加载发生错误的时需要一张图片作为提示。

```
Picasso.with(context)
    .load(url)
    .placeholder(R.drawable.user_placeholder)
    .error(R.drawable.user_placeholder_error)
.into(imageView);
```

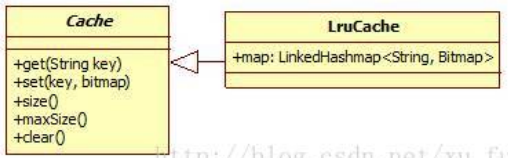
如果加载发生错误会重复三次请求，三次都失败才会显示erro **Place holder**

资源文件的加载：除了加载网络图片picasso还支持加载Resources, assets, files, content providers中的资源文件。

```
1.  Picasso.with(context).load(R.drawable.landing_screen).into(imageView1);
2.  Picasso.with(context).load(new File(...)).into(imageView2);
```

下面是picasso源码的解析（不看不影响使用）

Cache，缓存类



Lrucacha，主要是get和set方法，存储的结构采用了LinkedHashMap，这种map内部实现了lru算法（Least Recently Used 近期最少使用算法）。

```
1.  this.map = new LinkedHashMap<String, Bitmap>(0, 0.75f, true);
```

最后一个参数的解释：

true if the ordering should be done based on the last access (from least-recently accessed to most-recently accessed), and false if the ordering should be the order in which the entries were inserted.

因为可能会涉及多线程，所以在存取的时候都会加锁。而且每次set操作后都会判断当前缓存区是否已满，如果满了就清掉最少使用的图形。代码如下

```
1.  private void trimToSize(int maxSize) {
2.          while (true) {
3.              String key;
4.              Bitmap value;
5.              synchronized (this) {
6.                  if (size < 0 || (map.isEmpty() && size != 0)) {
7.                      throw new IllegalStateException(getClass().getName()
8.                              + ".sizeOf() is reporting inconsistent results!");
9.                  }
10.
11.                 if (size <= maxSize || map.isEmpty()) {
12.                     break;
13.                 }
14.
15.                 Map.Entry<String, Bitmap> toEvict = map.entrySet().iterator()
16.                         .next();
17.                 key = toEvict.getKey();
18.                 value = toEvict.getValue();
19.                 map.remove(key);
20.                 size -= Utils.getBitmapBytes(value);
21.                 evictionCount++;
22.             }
23.         }
24.  }
```

Request，操作封装类

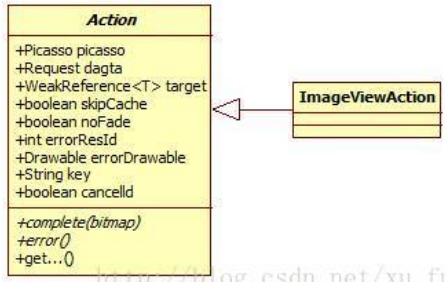
所有对图形的操作都会记录在这里，供之后图形的创建使用，如重新计算大小，旋转角度，也可以自定义变换，只需要实现Transformation，一个bitmap转换的接口。

```
1.  public interface Transformation {
2.    /**
3.     * Transform the source bitmap into a new bitmap. If you create a new bitmap
4.     * call {@link android.graphics.Bitmap#recycle()} on {@code source}. You may
5.     * if no transformation is required.
6.     */
7.    Bitmap transform(Bitmap source);
8.
9.    /**
10.    * Returns a unique key for the transformation, used for caching purposes. I
11.    * has parameters (e.g. size, scale factor, etc) then these should be part o
12.    */
13.   String key();
14. }
```

当操作封装好以后，会将Request传到另一个结构中Action。

Action

Action代表了一个具体的加载任务，主要用于图片加载后的结果回调，有两个抽象方法，complete和error，也就是当图片解析为bitmap后用户希望做什么。最简单的就是将bitmap设置给imageview，失败了就将错误通过回调通知到上层。

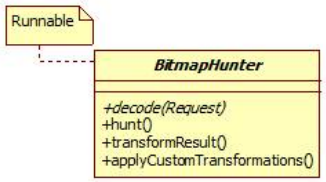


ImageViewAction实现了Action，在complete中将bitmap和imageview组成了一个PicassoDrawable，里面会实现淡出的动画效果。

```
1.  @Override
2.      public void complete(Bitmap result, Picasso.LoadedFrom from) {
3.          if (result == null) {
4.              throw new AssertionError(String.format(
5.                      "Attempted to complete action with no result!\n%s", this))
6.          }
7.
8.          ImageView target = this.target.get();
9.          if (target == null) {
10.             return;
11.         }
12.
13.         Context context = picasso.context;
14.         boolean debugging = picasso.debugging;
15.         PicassoDrawable.setBitmap(target, context, result, from, noFade,
16.                 debugging);
17.
18.         if (callback != null) {
19.             callback.onSuccess();
20.         }
21.     }
```

有了加载任务，具体的图片下载与解析是在哪里呢？这些都是耗时的操作，应该放在异步线程中进行，就是下面的BitmapHunter。

BitmapHunter



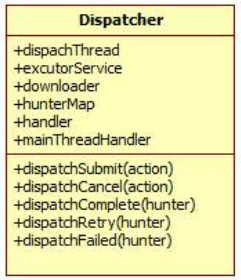
BitmapHunter是一个Runnable，其中有一个decode的抽象方法，用于子类实现不同类型资源的解析。

```
@Override
    public void run() {
        try {
            Thread.currentThread()
                    .setName(Utils.THREAD_PREFIX + data.getName());

            result = hunt();

            if (result == null) {
                dispatcher.dispatchFailed(this);
            } else {
                dispatcher.dispatchComplete(this);
            }
        } catch (IOException e) {
            exception = e;
            dispatcher.dispatchRetry(this);
        } catch (Exception e) {
            exception = e;
            dispatcher.dispatchFailed(this);
        } finally {
            Thread.currentThread().setName(Utils.THREAD_IDLE_NAME);
        }
    }

    abstract Bitmap decode(Request data) throws IOException;

    Bitmap hunt() throws IOException {
        Bitmap bitmap;

        if (!skipMemoryCache) {
            bitmap = cache.get(key);
            if (bitmap != null) {
                stats.dispatchCacheHit();
                loadedFrom = MEMORY;
                return bitmap;
            }
        }

        bitmap = decode(data);

        if (bitmap != null) {
            stats.dispatchBitmapDecoded(bitmap);
            if (data.needsTransformation() || exifRotation != 0) {
                synchronized (DECODE_LOCK) {
                    if (data.needsMatrixTransform() || exifRotation != 0) {
                        bitmap = transformResult(data, bitmap, exifRotation);
                    }
                    if (data.hasCustomTransformations()) {
                        bitmap = applyCustomTransformations(
                                data.transformations, bitmap);
                    }
                }
                stats.dispatchBitmapTransformed(bitmap);
```

```
54.                 }
55.             }
56.
57.             return bitmap;
58.         }
```

可以看到，在decode生成原始bitmap，之后会做需要的转换transformResult和applyCustomTransformations。最后在将最终的结果传递到上层dispatcher.dispatchComplete(this)。
基本的组成元素有了，那这一切是怎么连接起来运行呢，答案是Dispatcher。
Dispatcher任务调度器

在bitmaphunter成功得到bitmap后，就是通过dispatcher将结果传递出去的，当然让bitmaphunter执行也要通过Dispatcher。



Dispatcher内有一个HandlerThread，所有的请求都会通过这个thread转换，也就是请求也是异步的，这样应该是为了Ui线程更加流畅，同时保证请求的顺序，因为handler的消息队列。
外部调用的是dispatchXXX方法，然后通过handler将请求转换到对应的performXXX方法。
例如生成Action以后就会调用dispather的dispatchSubmit()来请求执行，

```
1.   void dispatchSubmit(Action action) {
2.           handler.sendMessage(handler.obtainMessage(REQUEST_SUBMIT, action));
3.       }
```

handler接到消息后转换到performSubmit方法

```
1.   void performSubmit(Action action) {
2.           BitmapHunter hunter = hunterMap.get(action.getKey());
3.           if (hunter != null) {
4.               hunter.attach(action);
5.               return;
6.           }
7.
8.           if (service.isShutdown()) {
9.               return;
10.          }
11.
12.          hunter = forRequest(context, action.getPicasso(), this, cache, stats,
13.                  action, downloader);
14.          hunter.future = service.submit(hunter);
15.          hunterMap.put(action.getKey(), hunter);
16.      }
```

这里将通过action得到具体的BitmapHunder，然后交给ExecutorService执行。

下面是Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView)的过程，

```
1.  public static Picasso with(Context context) {
2.          if (singleton == null) {
3.              singleton = new Builder(context).build();
4.          }
5.          return singleton;
6.      }
7.
8.      public Picasso build() {
9.              Context context = this.context;
10.
11.             if (downloader == null) {
12.                 downloader = Utils.createDefaultDownloader(context);
13.             }
14.             if (cache == null) {
15.                 cache = new LruCache(context);
16.             }
17.             if (service == null) {
18.                 service = new PicassoExecutorService();
19.             }
20.             if (transformer == null) {
21.                 transformer = RequestTransformer.IDENTITY;
22.             }
23.
24.             Stats stats = new Stats(cache);
25.
26.             Dispatcher dispatcher = new Dispatcher(context, service, HANDLER,
27.                     downloader, cache, stats);
28.
29.             return new Picasso(context, dispatcher, cache, listener,
30.                     transformer, stats, debugging);
31.         }
```

在Picasso.with()的时候会将执行所需的所有必备元素创建出来，如缓存cache、执行executorService、调度dispatch等，在load()时创建Request，在into()中创建action、bitmapHunter，并最终交给dispatcher执行。

参考：http://blog.csdn.net/xu_fu/article/details/17043231 (http://blog.csdn.net/xu_fu/article/details/17043231)

(/member/index.php?uid=Colin)

(/member/index.php?uid=stefory)

(/member/index.php?uid=zl3693372)

(/member/index.php?uid=西北狼)

(/member/index.php?uid=willchun)
(/member/index.php?uid=LinKen)


(/member/index.php?uid=倔强的骨头)

(/member/index.php?uid=钓鱼的猫)


(/member/index.php?uid=小熊叮当猫)
(/member/index.php?uid=oHeHeHou)


(/member/index.php?uid=vivaYao)
(/member/index.php?uid=vamayoumo)


(/member/index.php?uid=恨自己不能小清新)
(/member/index.php?uid=clhxlhm)

(/member/index.php?uid=cswpuc)
(/member/index.php?uid=sindyue)

(/member/index.php?uid=ccll)
(/member/index.php?uid=Mr.LONG)

(/member/index.php?uid=Bryan)
(/member/index.php?uid=lcx)

(/member/index.php?uid=tuogusa)
(/member/index.php?uid=gaozb)


(/member/index.php?uid=Caihong.zhou)
(/member/index.php?uid=WadeThere)

(/member/index.php?uid=qq920020062)
(/member/index.php?uid=Mr.jiang)

(/member/index.php?uid=aq15756005983)

(/member/index.php?uid=seamless)


(/member/index.php?uid=android小强)
(/member/index.php?uid=marcous)


(/member/index.php?uid=xtl1889)
(/member/index.php?uid=qizdyyt)

(/member/index.php?uid=speedy)

(/member/index.php?uid=赵海博)

(/member/index.php?uid=Hannibal)
(/member/index.php?uid=策马啸西风)

 (/member/index.php?uid=逍遥)  (/member/index.php?uid=balabala)

 (/member/index.php?uid=Hope.320)  (/member/index.php?uid=最纯真的回忆cg)

 (/member/index.php?uid=DavidLiang)  (/member/index.php?uid=VeiZhang)

 (/member/index.php?uid=wangzunbin)  (/member/index.php?uid=深灰色)

 (/member/index.php?uid=15881313220)  (/member/index.php?uid=xiaorulou)

相关文章

上一篇：**Android文字测量与绘制的两个注意点 (/a/anzhuokaifa/androidkaifa/2014/0731/1638.html)**

1、在用canvas绘制文字时需要测量文字的绘制范围，常用的方法是使用paint.getTextBound()，然后通过返回的Rect得到长宽，不过这个测量的宽度有些误差，导致计算位置坐标是偏移，使用paint.measureText()可以得到文字准确的宽度。 2、canvas.drawText(text, x

下一篇：**OnGlobalLayoutListener获得一个视图的高度 (/a/anzhuokaifa/androidkaifa/2014/0731/1640.html)**

当一个视图树的布局发生改变时，可以被ViewTreeObserver监听到， 这是一个注册监听视图树的观察者(observer)，在视图树的全局事件改变时得到通知。ViewTreeObserver不能直接实例化，而是通过getViewTreeObserver()获得。ViewTreeObserver 有如下内部类： in

发表评论

axuan2015 (/member/index.php?uid=axuan2015) . 2017-10-13

(/member/index.php?uid=axuan2015)

招聘多名Android 工程师:
任职要求：
1.计算机相关专业本科学历3年以上工作经验;
2.熟悉Android的基本组件与自定义控件;
3.有过聊天或办公协同相关开发经验加分;
薪资:20-35k

官网:http://www.icourt.cc/
公众号:
iCourt法秀;
iCourt技术团队

地点:
北京市朝阳区建国路58号 晋商博物馆4a

吸引您的不是高薪;而是指数倍增长的未来! 推荐与自荐都有礼相送;
联系方式:电话：15001099630 qq:2767356582 微信:alienware_2015

0 0 回复

Aria (/member/index.php?uid=Aria) . 2016-07-13

(/member/index.php?uid=Aria)

然而我最终选择了glide

1 14 回复

网友218.17.161.76 . 2016-03-07

> eling13 的原帖：
> 如果用picasso来加载大图，效果会如何？picasso有提供一些函数给用户让用户设置如果遇到大图时该如何，遇到小图时该如何，不然当一个列表里有大图有小图时，单单固定一个resize()，resize大了，小图容易失真啊，小了大图就不能大显示了

onlyScaleDown().

0 1 回复

网友218.17.161.76 . 2016-03-07

> eling13 的原帖：
> 如果用picasso来加载大图，效果会如何？picasso有提供一些函数给用户让用户设置如果遇到大图时该如何，遇到小图时该如何，不然当一个列表里有大图有小图时，单单固定一个resize()，resize大了，小图容易失真啊，小了大图就不能大显示了

onlyScaleDown()方法

0 0 回复

网友183.194.90.242 . 2015-11-20

```
void performSubmit(Action action) {
        BitmapHunter hunter = hunterMap.get(action.getKey());


          return;
        }
                                                                &nb
```

0 0 回复

网友183.194.90.242 . 2015-11-20

然并卵

0 5 回复

eling13 (/member/index.php?uid=eling13) . 2015-11-20

不知道picasso这框架，对于大分辨的图，比如1280*1044，加载出来的bitmap内存占很大啊

0 0 回复

eling13 (/member/index.php?uid=eling13) . 2015-11-20

如果用picasso来加载大图，效果会如何？picasso有提供一些函数给用户让用户设置如果遇到大图时该如何，遇到小图时该如何，不然当一个列表里有大图有小图时，单单固定一个resize()，resize大了，小图容易失真啊，小了大图就不能大显示了

2 0 回复

网友125.70.175.231 . 2015-07-23

> bluesofy 的原帖：
>
> > 网友60.255.0.17 的原帖：
> > shit 没有一个说设置缓存路径
>
> disk缓存么？貌似没提供设置方法，
> 默认是放在应用目录的cache/picasso-cache/，
> 实在想换位置就修改picasso源码里的Utils.PICASSO_CACHE的值咯~

泡网终于出现会思考的程序员了

9 0 回复

bluesofy (/member/index.php?uid=bluesofy) . 2015-07-23

> 网友60.255.0.17 的原帖：
> shit 没有一个说设置缓存路径

disk缓存么？貌似没提供设置方法，
默认是放在应用目录的cache/picasso-cache/，
实在想换位置就修改picasso源码里的Utils.PICASSO_CACHE的值咯~

0 0 回复

总: 2 页/11 条评论 **1** 2 **下一页**

推荐文章

Android插件化原理解析——概要 (/a/anzhuokaifa/androidkaifa/2016/0227/4005.html)

NumberProgressBar：一个简约性感的数字ProgressBar (/a/anzhuokaifa/androidkaifa/2014/0813/1645.html)

终于等到你Depth-LIB-Android (/a/anzhuokaifa/androidkaifa/2016/0429/4200.html)

App开发架构指南（谷歌官方文档译文） (/a/anzhuokaifa/androidkaifa/2017/0523/7963.html)

将Eclipse代码导入到Android Studio的两种方式 (/a/anzhuokaifa/androidkaifa/2015/0104/2259.html)

SQLBrite:一个响应式的数据查询框架 (/a/anzhuokaifa/androidkaifa/2015/0306/2552.html)

赞助商