Android图片加载

目前流行的Android图片加载库：

Picasso、 ImageLoader 、Fresco 、Glide

定义：Universal Image Loader 简称： UIL

**Picasso:**

* Picasso's fluent interface is nice. But jumping around with "with", "into", "load" you actually don't know what's behind the scene. It's confusing what's returned.
* Picasso简洁的接口定义非常好，但是有一些函数名字起的容易混淆，比如： "with", "into", "load" 实在是很难搞清楚使用场景。
* Picasso allows you to specify exact target size. It's useful when you have memory pressure or performance issues, you can trade off some image quality for speed.

Picasso 允许用户指定图片大小，这个功能对于内存压力问题和性能问题很有帮助；用户可以降低图片质量来达到减少使用内存和提高速度性能。

* Images are cached with size in its key, it's useful when you display images with different sizes.
* 图片是通过内部的key来缓存的，这一点对于使用不同大小图片现实很有用。
* You can customize the memory cache size. But its disc cache is only for http requests. For local images, if you care about loading speed, it's good to have a thumbnail disk cache so you don't have to read several MBs for an image every time. Picasso does not have this mechanism resizing and saving thumbnails on disk.
* 用户可以定制缓存的大小，但是磁盘缓存大小只有http请求的时候才有用。对于本地图片，可以把缩略图缓存在磁盘中，这应能提高读取速度，不用每次都读取好几MB的数据。Picasso 没有裁剪图片和保存缩略图的机制。
* Picasso does not expose the access to its cache instance. (You can get a hold of it when you first configure Picasso and keep it around...).
* Picasso没有暴露直接访问缓存的接口。
* Sometimes you want to asynchronously read image into a bitmap returned by a listener. Surprisingly Picasso doesn't have that. "fetch()" dose not pass back anything. "get()" is for synchronously read, and "load()" is for asynchronously draw a view.
* 有的时候我们想通过异步的方式来读取图片，然后再listener里面获得数据，非常意外的是Picasso 没有这样的接口。fetch()不传入任何参数；get() 同步读取，load()异步加载显示在view中。
* Picasso only has a few simple examples on the homepage, and you'll have to read through the unordered javadoc for advanced usages.
* Picasso 示例代码少，文档没有规律，比较乱。

**UIL:**

* UIL uses builders for customization. Almost everything can be configured.
* ImageLoader 使用定制化的方式构建的，几乎每种情况都能够通过配置满足。
* UIL does not allow you to specify the size you want to load into a view. It uses some rules based on the size of the view. It's not as flexible as Picasso. I have no way to load a lower resolution image to reduce memory footprint. (Edit: this behavior can be easily modified by adding an ImageSize argument in in the source code and bypass the view size checking)
* ImageLoader 不允许通过指定图片大小的方式来显示在view中，它通过基于view大小的一些规则来显示图片。这点不如picasso灵活。没有办法通过加载低分辨率的图像来降低内存的使用。
* UIL provides customizable disc cache, you can use this to cache the thumbnails with specified size. But it's not perfect. Here are the [details](http://stackoverflow.com/questions/21961505/android-universal-image-loader-by-pass-disc-memory-cache-check-when-loading-an). (Edit: if you care about speed and want multiple levels of thumbnail caching, like my case, you can modify the source code, let the disk cache use "memoryKey", and make it also size sensitive)
* ImageLoader 提供了定制化的磁盘缓存，你可以用这个来缓存指定大小的缩略图。但是不是很完美。
* UIL by default caches images of different sizes in memory, and it can be turned off in configuration.
* 默认的是在内存中缓存不同尺寸大小的图片，也可以通过配置来关闭这个功能。
* UIL exposes the backing memory and disk cache you can access.
* 暴露了内存缓存和磁盘缓存的接口，可以直接访问缓存。
* UIL provides flexible ways you can get a bitmap or load to a view.
* 提供了非常灵活的方法去得到和加载位图去显示到界面上。
* UIL is better in documentation. UIL gives the detailed usages on the Github page, and there's a linked tutorial.
* 文档比较好，有详细的用法示例代码。

### 一. 四大图片缓存基本信息

  
Universal ImageLoader 是很早开源的图片缓存，在早期被很多应用使用。

Picasso 是 Square 开源的项目，且他的主导者是 JakeWharton，所以广为人知。

Glide 是 Google 员工的开源项目，被一些 Google App 使用。

Fresco 是 Facebook 在今年上半年开源的图片缓存，主要特点包括：  
(1) 两个内存缓存加上 Native 缓存构成了三级缓存

(2) 支持流式，可以类似网页上模糊渐进式显示图片

(3) 对多帧动画图片支持更好，如 Gif、WebP

### 二、基本概念

在正式对比前，先了解几个图片缓存通用的概念：  
(1) RequestManager：请求生成和管理模块

(2) Engine：引擎部分，负责创建任务(获取数据)，并调度执行

(3) GetDataInterface：数据获取接口，负责从各个数据源获取数据。  
比如 MemoryCache 从内存缓存获取数据、DiskCache 从本地缓存获取数据，下载器从网络获取数据等。

(4) Displayer：资源(图片)显示器，用于显示或操作资源。  
比如 ImageView，这几个图片缓存都不仅仅支持 ImageView，同时支持其他 View 以及虚拟的 Displayer 概念。

(5) Processor 资源(图片)处理器  
负责处理资源，比如旋转、压缩、截取等。

以上概念的称呼在不同图片缓存中可能不同，比如 Displayer 在 ImageLoader 中叫做 ImageAware，在 Picasso 和 Glide 中叫做 Target。

### 三、共同优点

**1. 使用简单**  
都可以通过一句代码可实现图片获取和显示。

**2. 可配置度高，自适应程度高**  
图片缓存的下载器(重试机制)、解码器、显示器、处理器、内存缓存、本地缓存、线程池、缓存算法等大都可轻松配置。

自适应程度高，根据系统性能初始化缓存配置、系统信息变更后动态调整策略。  
比如根据 CPU 核数确定最大并发数，根据可用内存确定内存缓存大小，网络状态变化时调整最大并发数等。

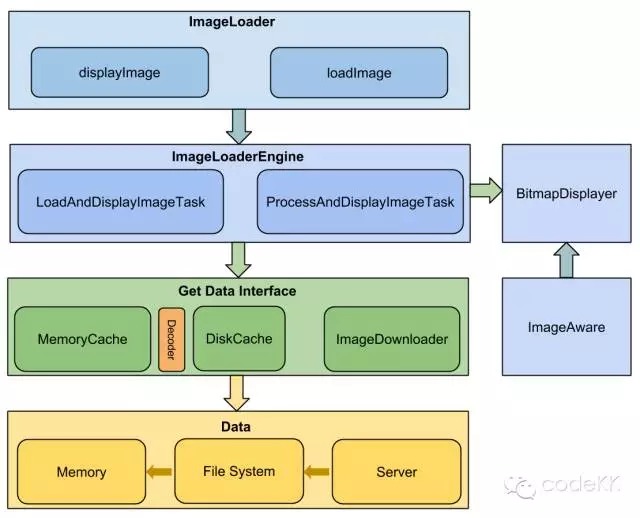
**3. 多级缓存**  
都至少有两级缓存、提高图片加载速度。

**4. 支持多种数据源**  
支持多种数据源，网络、本地、资源、Assets 等

**5. 支持多种 Displayer**  
不仅仅支持 ImageView，同时支持其他 View 以及虚拟的 Displayer 概念。

其他小的共同点包括支持动画、支持 transform 处理、获取 EXIF 信息等。

### 四、ImageLoader 设计及优点

1. 总体设计及流程

上面是 ImageLoader 的总体设计图。整个库分为 ImageLoaderEngine，Cache 及 ImageDownloader，ImageDecoder，BitmapDisplayer，BitmapProcessor 五大模块，其中 Cache 分为 MemoryCache 和 DiskCache 两部分。

简单的讲就是 ImageLoader 收到加载及显示图片的任务，并将它交给 ImageLoaderEngine，ImageLoaderEngine 分发任务到具体线程池去执行，任务通过 Cache 及 ImageDownloader 获取图片，中间可能经过 BitmapProcessor 和 ImageDecoder 处理，最终转换为Bitmap 交给 BitmapDisplayer 在 ImageAware 中显示。

#### 2. ImageLoader 优点

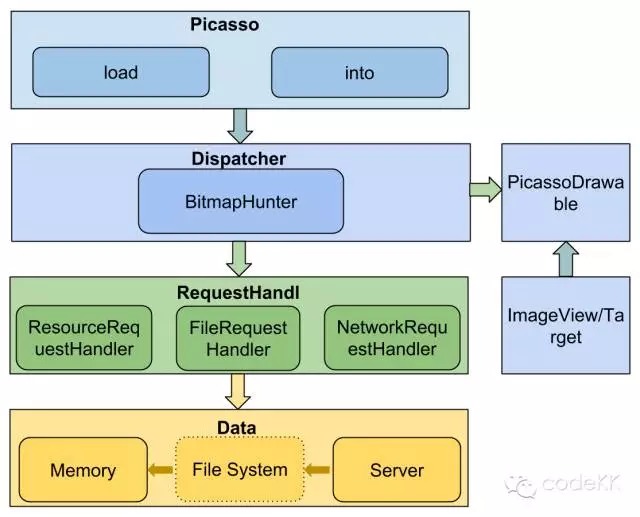
**(1) 支持下载进度监听**

**(2) 可以在 View 滚动中暂停图片加载**  
通过 PauseOnScrollListener 接口可以在 View 滚动中暂停图片加载。

**(3) 默认实现多种内存缓存算法** 这几个图片缓存都可以配置缓存算法，不过 ImageLoader 默认实现了较多缓存算法，如 Size 最大先删除、使用最少先删除、最近最少使用、先进先删除、时间最长先删除等。

**(4) 支持本地缓存文件名规则定义**

### 五、Picasso 设计及优点



#### 1. 总体设计及流程

上面是 Picasso 的总体设计图。整个库分为 Dispatcher，RequestHandler 及 Downloader，PicassoDrawable 等模块。

Dispatcher 负责分发和处理 Action，包括提交、暂停、继续、取消、网络状态变化、重试等等。

简单的讲就是 Picasso 收到加载及显示图片的任务，创建 Request 并将它交给 Dispatcher，Dispatcher 分发任务到具体 RequestHandler，任务通过 MemoryCache 及 Handler(数据获取接口) 获取图片，图片获取成功后通过 PicassoDrawable 显示到 Target 中。

需要注意的是上面 Data 的 File system 部分，Picasso 没有自定义本地缓存的接口，默认使用 http 的本地缓存，API 9 以上使用 okhttp，以下使用 Urlconnection，所以如果需要自定义本地缓存就需要重定义 Downloader。

#### 2. Picasso 优点

**(1) 自带统计监控功能**  
支持图片缓存使用的监控，包括缓存命中率、已使用内存大小、节省的流量等。

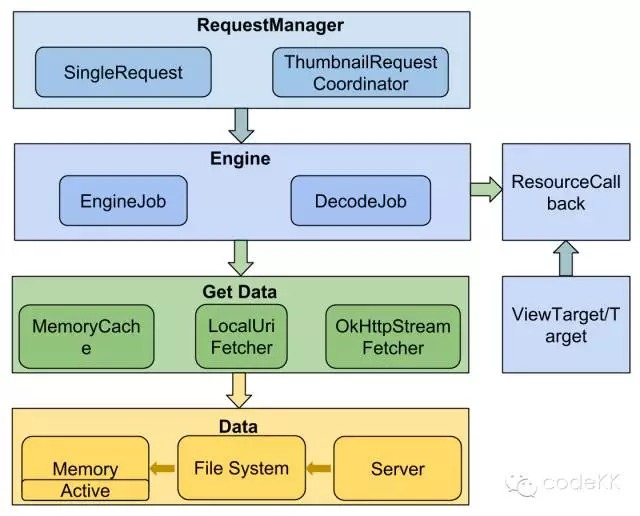
**(2) 支持优先级处理**  
每次任务调度前会选择优先级高的任务，比如 App 页面中 Banner 的优先级高于 Icon 时就很适用。

**(3) 支持延迟到图片尺寸计算完成加载**

**(4) 支持飞行模式、并发线程数根据网络类型而变**  
手机切换到飞行模式或网络类型变换时会自动调整线程池最大并发数，比如 wifi 最大并发为 4， 4g 为 3，3g 为 2。  
这里 Picasso 根据网络类型来决定最大并发数，而不是 CPU 核数。

**(5) “无”本地缓存**  
无”本地缓存，不是说没有本地缓存，而是 Picasso 自己没有实现，交给了 Square 的另外一个网络库 okhttp 去实现，这样的好处是可以通过请求 Response Header 中的 Cache-Control 及 Expired 控制图片的过期时间。

### 六、Glide 设计及优点



#### 1. 总体设计及流程

上面是 Glide 的总体设计图。整个库分为 RequestManager(请求管理器)，Engine(数据获取引擎)、 Fetcher(数据获取器)、MemoryCache(内存缓存)、DiskLRUCache、Transformation(图片处理)、Encoder(本地缓存存储)、Registry(图片类型及解析器配置)、Target(目标) 等模块。

简单的讲就是 Glide 收到加载及显示资源的任务，创建 Request 并将它交给RequestManager，Request 启动 Engine 去数据源获取资源(通过 Fetcher )，获取到后 Transformation 处理后交给 Target。

Glide 依赖于 DiskLRUCache、GifDecoder 等开源库去完成本地缓存和 Gif 图片解码工作。

#### 2. Glide 优点

**(1) 图片缓存->媒体缓存**  
Glide 不仅是一个图片缓存，它支持 Gif、WebP、缩略图。甚至是 Video，所以更该当做一个媒体缓存。

**(2) 支持优先级处理**

**(3) 与 Activity/Fragment 生命周期一致，支持 trimMemory**  
Glide 对每个 context 都保持一个 RequestManager，通过 FragmentTransaction 保持与 Activity/Fragment 生命周期一致，并且有对应的 trimMemory 接口实现可供调用。

**(4) 支持 okhttp、Volley**  
Glide 默认通过 UrlConnection 获取数据，可以配合 okhttp 或是 Volley 使用。实际 ImageLoader、Picasso 也都支持 okhttp、Volley。

**(5) 内存友好**  
① Glide 的内存缓存有个 active 的设计  
从内存缓存中取数据时，不像一般的实现用 get，而是用 remove，再将这个缓存数据放到一个 value 为软引用的 activeResources map 中，并计数引用数，在图片加载完成后进行判断，如果引用计数为空则回收掉。

② 内存缓存更小图片  
Glide 以 url、view\_width、view\_height、屏幕的分辨率等做为联合 key，将处理后的图片缓存在内存缓存中，而不是原始图片以节省大小

③ 与 Activity/Fragment 生命周期一致，支持 trimMemory

④ 图片默认使用默认 RGB\_565 而不是 ARGB\_888  
虽然清晰度差些，但图片更小，也可配置到 ARGB\_888。

其他：Glide 可以通过 signature 或不使用本地缓存支持 url 过期

### 七、汇总

|  |  |
| --- | --- |
|  | **Fresco** (-) - Huge size of library - No Callback with View, Bitmap parameters - SimpleDraweeView doesn't support wrap\_content - Huge size of cache (+) - Pretty fast image loader (for small && medium images) - A lot of functionality(streaming, drawing tools, memory management, etc) - Possibility to setup directly in xml (for example round corners) - GIF support - WebP support  **Picasso** (-) - Slow loading big images from internet into ListView (+) - Tinny size of library - Small size of cache - Simple in use - UI is not freeze - WebP support  **Glide** (-) - Big size of library (+) - Tinny size of cache - Simple in use - GIF support - WebP support - Fast loading big images from internet into ListView - UI is not freeze - BitmapPool to re-use memory and thus lesser GC events |

建议： 使用**Picasso**

