  这个图片异步加载并缓存的类已经被很多开发者所使用，是最常用的几个开源库之一，主流的应用，随便反编译几个火的项目，都可以见到它的身影。

       可是有的人并不知道如何去使用这库如何进行配置，网上查到的信息对于刚接触的人来说可能太少了，下面我就把我使用过程中所知道的写了下来，希望可以帮助自己和别人更深入了解这个库的使用和配置。

        GITHUB上的下载路径为：[https://github.com/nostra13/Android-Universal-Image-Loader](https://github.com/nostra13/Android-Universal-Image-Loader" \t "_blank) ，下载最新的库文件，并且导入到项目的LIB下便可以使用。

        该库以及DEMO本地下载链接：[下载地址](http://download.csdn.net/detail/vipzjyno1/7168691" \t "_blank)

一、介绍

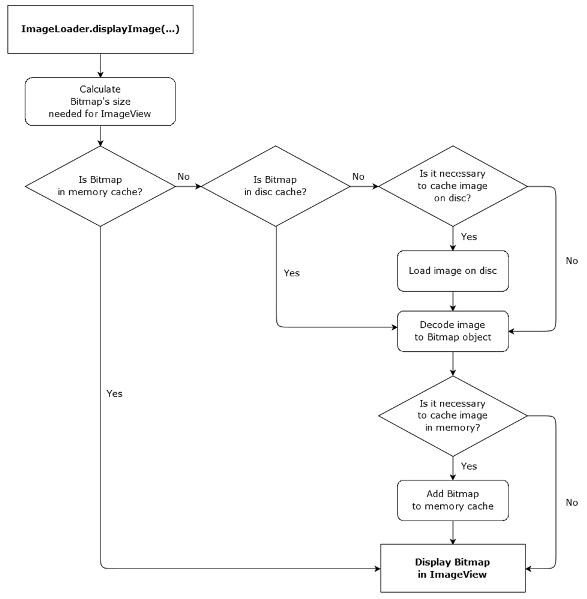
 Android-Universal-Image-Loader是一个开源的UI组件程序，该项目的目的是提供一个可重复使用的仪器为异步图像加载，缓存和显示。所以，如果你的程序里需要这个功能的话，那么不妨试试它。因为已经封装好了一些类和方法。我们 可以直接拿来用了。而不用重复去写了。其实，写一个这方面的程序还是比较麻烦的，要考虑多线程，缓存，内存溢出等很多方面。但是，你也可以参考这个例子来自己写出更好的程序。在此为大家介绍一下：

[](http://blog.csdn.net/wangjinyu501/article/details/8091623)

二、特点

多线程的图像加载  
的可能性的宽调谐对ImageLoader的配置（线程池的大小，HTTP选项，内存和光盘高速缓存，显示图像，以及其他）  
的图像的可能性中的缓存存储器和/或设备的文件器系统（或SD卡）  
可以“听”加载过程中  
可自定义每个显示的图像调用分隔的选项  
Widget支持  
Android 1.5以上支持  
简单描述一下这个项目的结构：每一个图片的加载和显示任务都运行在独立的线程中，除非这个图片缓存在内存中，这种情况下图片会立即显示。如果需要的图片缓存在本地，他们会开启一个独立的线程队列。如果在缓存中没有正确的图片，任务线程会从线程池中获取，因此，快速显示缓存图片时不会有明显的障碍。（别人那边借鉴的这段）

流程图：



三、使用方法

  这是 一个开源的Android关于下载显示图片的工具类，在这个下载包里面jar文件，用于我们导入项目使用，具体使用方法在包里面也含有。下面是一个例子：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. <uses-permission android:name="android.permission.INTERNET" />
2. <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

由于是使用过程中会图片获取要通过网络，并且有缓存设置，所以这2个权限必须要有。

**很多人想知道如何设置缓存的目录，可以通过以下方法：**

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. File cacheDir = StorageUtils.getOwnCacheDirectory(getApplicationContext(), "imageloader/Cache");

//这个是你希望的缓存文件的目录：imageloader/Cache  
  
之后在ImageLoaderConfiguration的配置文件中通过设置

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. .discCache(**new** UnlimitedDiscCache(cacheDir))//自定义缓存路径

方法来设置该应用的图片缓存路径。

下面我们就来开始看如何使用这个图片异步加载库把：

一.

        先要配置ImageLoaderConfiguration这个类实现全局ImageLoader的实现情况。

可以选择在Application中初始化设置该类。

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. ImageLoaderConfiguration config = **new** ImageLoaderConfiguration
2. .Builder(context)
3. .memoryCacheExtraOptions(480, 800) // max width, max height，即保存的每个缓存文件的最大长宽
4. .discCacheExtraOptions(480, 800, CompressFormat.JPEG, 75, **null**) // Can slow ImageLoader, use it carefully (Better don't use it)/设置缓存的详细信息，最好不要设置这个
5. .threadPoolSize(3)//线程池内加载的数量
6. .threadPriority(Thread.NORM\_PRIORITY - 2)
7. .denyCacheImageMultipleSizesInMemory()
8. .memoryCache(**new** UsingFreqLimitedMemoryCache(2 \* 1024 \* 1024)) // You can pass your own memory cache implementation/你可以通过自己的内存缓存实现
9. .memoryCacheSize(2 \* 1024 \* 1024)
10. .discCacheSize(50 \* 1024 \* 1024)
11. .discCacheFileNameGenerator(**new** Md5FileNameGenerator())//将保存的时候的URI名称用MD5 加密
12. .tasksProcessingOrder(QueueProcessingType.LIFO)
13. .discCacheFileCount(100) //缓存的文件数量
14. .discCache(**new** UnlimitedDiscCache(cacheDir))//自定义缓存路径
15. .defaultDisplayImageOptions(DisplayImageOptions.createSimple())
16. .imageDownloader(**new** BaseImageDownloader(context, 5 \* 1000, 30 \* 1000)) // connectTimeout (5 s), readTimeout (30 s)超时时间
17. .writeDebugLogs() // Remove for release app
18. .build();//开始构建
19. // Initialize ImageLoader with configuration.

以上的配置看个人需求进行选择，不是所有都要进行配置。

配置好ImageLoaderConfiguration后，调用以下方法来实现初始化：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. ImageLoader.getInstance().init(config);//全局初始化此配置

注：ImageLoaderConfiguration 配置中的.discCacheFileNameGenerator()方法是将缓存下来的文件以什么方式命名

          里面可以调用的方法有  1.new Md5FileNameGenerator() //使用MD5对UIL进行加密命名

                                                2.new HashCodeFileNameGenerator()//使用HASHCODE对UIL进行加密命名

二.

        使用ImageLoader进行图片加载的时候，先要实例化ImageLoader，调用以下方法进行实例化，在每个布局里面都要实例化后再使用。

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. **protected** ImageLoader imageLoader = ImageLoader.getInstance();

之后进行显示的图片的各种格式DisplayImageOptions 的设置：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. DisplayImageOptions options;
2. options = **new** DisplayImageOptions.Builder()
3. .showImageOnLoading(R.drawable.ic\_launcher) //设置图片在下载期间显示的图片
4. .showImageForEmptyUri(R.drawable.ic\_launcher)//设置图片Uri为空或是错误的时候显示的图片
5. .showImageOnFail(R.drawable.ic\_launcher)  //设置图片加载/解码过程中错误时候显示的图片
6. .cacheInMemory(**true**)//设置下载的图片是否缓存在内存中
7. .cacheOnDisc(**true**)//设置下载的图片是否缓存在SD卡中
8. .considerExifParams(**true**)  //是否考虑JPEG图像EXIF参数（旋转，翻转）
9. .imageScaleType(ImageScaleType.EXACTLY\_STRETCHED)//设置图片以如何的编码方式显示
10. .bitmapConfig(Bitmap.Config.RGB\_565)//设置图片的解码类型//
11. .decodingOptions(android.graphics.BitmapFactory.Options decodingOptions)//设置图片的解码配置
12. //.delayBeforeLoading(int delayInMillis)//int delayInMillis为你设置的下载前的延迟时间
13. //设置图片加入缓存前，对bitmap进行设置
14. //.preProcessor(BitmapProcessor preProcessor)
15. .resetViewBeforeLoading(**true**)//设置图片在下载前是否重置，复位
16. .displayer(**new** RoundedBitmapDisplayer(20))//是否设置为圆角，弧度为多少
17. .displayer(**new** FadeInBitmapDisplayer(100))//是否图片加载好后渐入的动画时间
18. .build();//构建完成

按照你所需要的配置去设置，如果不需要的就可以不做配置。  
  
注：  
  
  以上配置中的：

 1）.imageScaleType(ImageScaleType imageScaleType)  是设置 图片的缩放方式  
     缩放类型mageScaleType:  
  
              EXACTLY :图像将完全按比例缩小的目标大小  
  
              EXACTLY\_STRETCHED:图片会缩放到目标大小完全  
  
              IN\_SAMPLE\_INT:图像将被二次采样的整数倍  
  
              IN\_SAMPLE\_POWER\_OF\_2:图片将降低2倍，直到下一减少步骤，使图像更小的目标大小  
  
              NONE:图片不会调整  
  2）.displayer(BitmapDisplayer displayer)   是设置 图片的显示方式  
  
      显示方式displayer：  
  
              RoundedBitmapDisplayer（int roundPixels）设置圆角图片  
  
              FakeBitmapDisplayer（）这个类什么都没做  
  
              FadeInBitmapDisplayer（int durationMillis）设置图片渐显的时间  
  
　　　　　　　  SimpleBitmapDisplayer()正常显示一张图片

之后按照需求调用

1.纯粹为了加载默认配置的一个图片的  
  
  方法:

  public void displayImage(String uri, ImageView imageView) {}

  具体实现：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. ImageLoader.getInstance().displayImage(imageUrl, imageView); // imageUrl代表图片的URL地址，imageView代表承载图片的IMAGEVIEW控件

2.加载自定义配置的一个图片的  
  
  方法:

  public void displayImage(String uri, ImageView imageView, DisplayImageOptions options) {}

  具体实现：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. ImageLoader.getInstance().displayImage(imageUrl, imageView，options); // imageUrl代表图片的URL地址，imageView代表承载图片的IMAGEVIEW控件 ， options代表DisplayImageOptions配置文件

3.图片加载时候带加载情况的监听  
  
  方法：

 public void displayImage(String uri, ImageAware imageAware, DisplayImageOptions options,ImageLoadingListener listener) {｝

  ImageLoadingListener 用于监听图片的下载情况。

  具体实现：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. imageLoader.displayImage(imageUrl, imageView, options, **new** ImageLoadingListener() {
2. @Override
3. **public** **void** onLoadingStarted() {
4. //开始加载的时候执行
5. }
6. @Override
7. **public** **void** onLoadingFailed(FailReason failReason) {
8. //加载失败的时候执行
9. }
10. @Override
11. **public** **void** onLoadingComplete(Bitmap loadedImage) {
12. //加载成功的时候执行
13. }
14. @Override
15. **public** **void** onLoadingCancelled() {
16. //加载取消的时候执行
18. }});

4.图片加载时候，带监听又带加载进度条的情况

调用：  
public void loadImage(String uri, ImageSize targetImageSize, DisplayImageOptions options,  
                    ImageLoadingListener listener, ImageLoadingProgressListener progressListener) {}  
   具体实现：

**[java]** [view plaincopy](http://blog.csdn.net/vipzjyno1/article/details/23206387)[在CODE上查看代码片](https://code.csdn.net/snippets/283481)

1. imageLoader.displayImage(imageUrl, imageView, options, **new** ImageLoadingListener() {
2. @Override
3. **public** **void** onLoadingStarted() {
4. //开始加载的时候执行
5. }
6. @Override
7. **public** **void** onLoadingFailed(FailReason failReason) {
8. //加载失败的时候执行
9. }
10. @Override
11. **public** **void** onLoadingComplete(Bitmap loadedImage) {
12. //加载成功的时候执行
13. }
14. @Override
15. **public** **void** onLoadingCancelled() {
16. //加载取消的时候执行
17. },**new** ImageLoadingProgressListener() {
18. @Override
19. **public** **void** onProgressUpdate(String imageUri, View view, **int** current,**int** total) {
20. //在这里更新 ProgressBar的进度信息
21. }
22. });

四、注意事项  
  1.上述提到的2个权限必须加入，否则会出错  
  2.ImageLoaderConfiguration必须配置并且全局化的初始化这个配置ImageLoader.getInstance().init(config);  否则也会出现错误提示  
  3.ImageLoader是根据ImageView的height，width确定图片的宽高。  
  4.如果经常出现OOM（别人那边看到的，觉得很有提的必要）  
   ①减少配置之中线程池的大小，(.threadPoolSize).推荐1-5；  
   ②使用.bitmapConfig(Bitmap.config.RGB\_565)代替ARGB\_8888;  
   ③使用.imageScaleType(ImageScaleType.IN\_SAMPLE\_INT)或者        try.imageScaleType(ImageScaleType.EXACTLY)；  
   ④避免使用RoundedBitmapDisplayer.他会创建新的ARGB\_8888格式的Bitmap对象；  
   ⑤使用.memoryCache(new WeakMemoryCache())，不要使用.cacheInMemory();  
  
下面有人问到如何加载本地图片什么的，之后把那位亲的回答补充上来，谢谢各位大大提出的不足（那时候用的时候只想到用网络图片了，所以也没考虑这么多）。

1. String imageUri = "http://site.com/image.png"; // from Web
2. String imageUri = "file:///mnt/sdcard/image.png"; // from SD card
3. String imageUri = "content://media/external/audio/albumart/13"; // from content provider
4. String imageUri = "assets://image.png"; // from assets
5. String imageUri = "drawable://" + R.drawable.image; // from drawables (only images, non-9patch)

    大致就说到这里，可能有的地方不足欢迎提出，希望大家可以更加容易入手、使用这个库。