# 配置

下载类库：

<https://github.com/nostra13/Android-Universal-Image-Loader/tree/master/downloads>

参考使用文章

<http://blog.csdn.net/vipzjyno1/article/details/23206387>

## 1：配置初始化imageloader

/\*\* 初始化ImageLoader \*/

**public** **static** **void** initImageLoader(Context context) {

File cacheDir = StorageUtils.*getOwnCacheDirectory*(context, "topnews/Cache");//获取到缓存的目录地址

Log.*d*("cacheDir", cacheDir.getPath());

//创建配置ImageLoader(所有的选项都是可选的,只使用那些你真的想定制)，这个可以设定在APPLACATION里面，设置为全局的配置参数

ImageLoaderConfiguration config = **new** ImageLoaderConfiguration

.Builder(context)

//.memoryCacheExtraOptions(480, 800) // max width, max height，即保存的每个缓存文件的最大长宽

//.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

.threadPoolSize(3)//线程池内加载的数量

.threadPriority(Thread.*NORM\_PRIORITY* - 2)

.denyCacheImageMultipleSizesInMemory()

//.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

//.memoryCacheSize(2 \* 1024 \* 1024)

///.discCacheSize(50 \* 1024 \* 1024)

.discCacheFileNameGenerator(**new** Md5FileNameGenerator())//将保存的时候的URI名称用MD5 加密

//.discCacheFileNameGenerator(new HashCodeFileNameGenerator())//将保存的时候的URI名称用HASHCODE加密

.tasksProcessingOrder(QueueProcessingType.*LIFO*)

//.discCacheFileCount(100) //缓存的File数量

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

//.defaultDisplayImageOptions(DisplayImageOptions.createSimple())

//.imageDownloader(new BaseImageDownloader(context, 5 \* 1000, 30 \* 1000)) // connectTimeout (5 s), readTimeout (30 s)超时时间

.writeDebugLogs() // Remove for release app

.build();

// Initialize ImageLoader with configuration.

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

}

## 2：权限配置

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

# 使用

## 1配置option

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

：

## 2：使用

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

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. }});