LRU:最近最少使用算法。

LruCache:内存缓存

DiskLruCache:磁盘缓存

1. 初始化ImagLoader时，创建LruCache,DiskLruCache

LruCache mMemoryCache=new LruCache<StringKey,Bitmap>(cacheSize){

Protected int size of(String key,Bitmap bitmap){

//该方法中计算图片的大小

}

}

DiskLruCache mDiskCache=new DisLruCache.open(disCacheDir,1,1,Disk\_Cache\_Size);

1. 内存缓存的添加和获取

//添加mMemoryCache.put(key,bitmap)

//获取mMemoryCache,get(key,bitmap)

1. 磁盘缓存的添加和获取

//DiskLruCache.Editor 的commit/abort方法来进行文件的提交和撤销操作

//DiskLruCache.Snapshot的FileInputStream获取文件流，然后进行压缩，再将bitmap

添加到缓存中显示

Glide.with().load().into();

第一个with()方法的源码还是比较好理解的。其实就是为了得到一个RequestManager对象而已，然后Glide会根据我们传入with()方法的参数来确定图片加载的生命周期

传入图片地址，创建用于加载图片的ModelLoader对象，再创建用于请求图片的RequestBuilder类并返回。DrawableTypeRequest类提供了asBitmap()和asGif()这两个方法，用于强制指定加载静态图片和动态图片

将ImageView转换成一个Target，将图片加载的Request和Target绑定，这样就可以通过

setTag()、getTag()进行view的复用

然后开始Request.begin(),先去设置图片的宽高，在去加载图片，首先是一级缓存使用

LruCache(内部是LinkHashMap)算法将资源缓存在内存中，接着是使用软引用进行二级缓存，二级缓存的资源为正在显示的图片，如果一二级缓存没有读取到则开启网络请求