ListView заполение

Here's what I created to hold the images that my app is currently displaying. Please note that the "Log" object in use here is my custom wrapper around the final Log class inside Android.

package com.wilson.android.library;

/\*

Licensed to the Apache Software Foundation (ASF) under one

or more contributor license agreements. See the NOTICE file

distributed with this work for additional information

regarding copyright ownership. The ASF licenses this file

to you under the Apache License, Version 2.0 (the

"License"); you may not use this file except in compliance

with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,

software distributed under the License is distributed on an

"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

KIND, either express or implied. See the License for the

specific language governing permissions and limitations

under the License.

\*/

import java.io.IOException;

public class DrawableManager {

private final Map<String, Drawable> drawableMap;

public DrawableManager() {

drawableMap = new HashMap<String, Drawable>();

}

public Drawable fetchDrawable(String urlString) {

if (drawableMap.containsKey(urlString)) {

return drawableMap.get(urlString);

}

Log.d(this.getClass().getSimpleName(), "image url:" + urlString);

try {

InputStream is = fetch(urlString);

Drawable drawable = Drawable.createFromStream(is, "src");

if (drawable != null) {

drawableMap.put(urlString, drawable);

Log.d(this.getClass().getSimpleName(), "got a thumbnail drawable: " + drawable.getBounds() + ", "

+ drawable.getIntrinsicHeight() + "," + drawable.getIntrinsicWidth() + ", "

+ drawable.getMinimumHeight() + "," + drawable.getMinimumWidth());

} else {

Log.w(this.getClass().getSimpleName(), "could not get thumbnail");

}

return drawable;

} catch (MalformedURLException e) {

Log.e(this.getClass().getSimpleName(), "fetchDrawable failed", e);

return null;

} catch (IOException e) {

Log.e(this.getClass().getSimpleName(), "fetchDrawable failed", e);

return null;

}

}

public void fetchDrawableOnThread(final String urlString, final ImageView imageView) {

if (drawableMap.containsKey(urlString)) {

imageView.setImageDrawable(drawableMap.get(urlString));

}

final Handler handler = new Handler() {

@Override

public void handleMessage(Message message) {

imageView.setImageDrawable((Drawable) message.obj);

}

};

Thread thread = new Thread() {

@Override

public void run() {

//TODO : set imageView to a "pending" image

Drawable drawable = fetchDrawable(urlString);

Message message = handler.obtainMessage(1, drawable);

handler.sendMessage(message);

}

};

thread.start();

}

private InputStream fetch(String urlString) throws MalformedURLException, IOException {

DefaultHttpClient httpClient = new DefaultHttpClient();

HttpGet request = new HttpGet(urlString);

HttpResponse response = httpClient.execute(request);

return response.getEntity().getContent();

}

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 92 |  | | I think you should use SoftReferences so that your program will never cause OutOfMemoryException. As GC can clear softreferences when heap size is increasing... you can manage your own generation like after some seconds you can put your images to that list and before loading you should check that if image exists then don't download it again rather collect it from that list and also putting it back to your softref list and after sometime you can purge your hardlist :) – [AZ\_](http://stackoverflow.com/users/185022/az) [Jan 18 '11 at 8:08](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment5215402_559781) | | |  |  | | --- | --- | | 29 |  | | Google Shelves project is an excellent example look how they did [code.google.com/p/shelves](http://code.google.com/p/shelves/) – [AZ\_](http://stackoverflow.com/users/185022/az) [Jan 18 '11 at 8:09](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment5215415_559781) | | |  |  | | --- | --- | | 11 |  | | Don't you miss a return when drawableMap contains the image ... without starting the fetching-thread? – [Karussell](http://stackoverflow.com/users/194609/karussell) [Mar 29 '11 at 22:06](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment6213963_559781) | | |  |  | | --- | --- | | 3 |  | | This code has several problems. Firstly you should cache Drawables, that will cause a memory leak:[stackoverflow.com/questions/7648740/…](http://stackoverflow.com/questions/7648740/consequences-of-drawable-setcallbacknull" \o "consequences of drawable setcallbacknull) . Secondly the cache itself is never cleared so it will grow forever, that's another memory leak. – [satur9nine](http://stackoverflow.com/users/215266/satur9nine) [Nov 15 '11 at 5:35](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment9977556_559781) | | |  |  | | --- | --- | | 8 |  | | haven't any one heard about LRU Cache [developer.android.com/training/displaying-bitmaps/…](http://developer.android.com/training/displaying-bitmaps/cache-bitmap.html) – [Muhammad Babar](http://stackoverflow.com/users/1939564/muhammad-babar) [May 28 '13 at 7:26](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment24188393_559781) | | | |
| **Update: Note that this answer is pretty ineffective now. The Garbage Collector acts aggressively on SoftReference and WeakReference, so this code is NOT suitable for new apps.** (Instead, try libraries like [Universal Image Loader](https://github.com/nostra13/Android-Universal-Image-Loader) suggested in other answers.)  Thanks to James for the code, and Bao-Long for the suggestion of using SoftReference. I implemented the SoftReference changes on James' code. Unfortunately SoftReferences caused my images to be garbage collected too quickly. In my case it was fine without the SoftReference stuff, because my list size is limited and my images are small.  There's a discussion from a year ago regarding the SoftReferences on google groups: [link to thread](http://groups.google.com/group/android-developers/browse_thread/thread/ebabb0dadf38acc1). As a solution to the too-early garbage collection, they suggest the possibility of manually setting the VM heap size using dalvik.system.VMRuntime.setMinimumHeapSize(), which is not very attractive to me.  public DrawableManager() {  drawableMap = new HashMap<String, SoftReference<Drawable>>();  }  public Drawable fetchDrawable(String urlString) {  SoftReference<Drawable> drawableRef = drawableMap.get(urlString);  if (drawableRef != null) {  Drawable drawable = drawableRef.get();  if (drawable != null)  return drawable;  // Reference has expired so remove the key from drawableMap  drawableMap.remove(urlString);  }  if (Constants.LOGGING) Log.d(this.getClass().getSimpleName(), "image url:" + urlString);  try {  InputStream is = fetch(urlString);  Drawable drawable = Drawable.createFromStream(is, "src");  drawableRef = new SoftReference<Drawable>(drawable);  drawableMap.put(urlString, drawableRef);  if (Constants.LOGGING) Log.d(this.getClass().getSimpleName(), "got a thumbnail drawable: " + drawable.getBounds() + ", "  + drawable.getIntrinsicHeight() + "," + drawable.getIntrinsicWidth() + ", "  + drawable.getMinimumHeight() + "," + drawable.getMinimumWidth());  return drawableRef.get();  } catch (MalformedURLException e) {  if (Constants.LOGGING) Log.e(this.getClass().getSimpleName(), "fetchDrawable failed", e);  return null;  } catch (IOException e) {  if (Constants.LOGGING) Log.e(this.getClass().getSimpleName(), "fetchDrawable failed", e);  return null;  }  }  public void fetchDrawableOnThread(final String urlString, final ImageView imageView) {  SoftReference<Drawable> drawableRef = drawableMap.get(urlString);  if (drawableRef != null) {  Drawable drawable = drawableRef.get();  if (drawable != null) {  imageView.setImageDrawable(drawableRef.get());  return;  }  // Reference has expired so remove the key from drawableMap  drawableMap.remove(urlString);  }  final Handler handler = new Handler() {  @Override  public void handleMessage(Message message) {  imageView.setImageDrawable((Drawable) message.obj);  }  };  Thread thread = new Thread() {  @Override  public void run() {  //TODO : set imageView to a "pending" image  Drawable drawable = fetchDrawable(urlString);  Message message = handler.obtainMessage(1, drawable);  handler.sendMessage(message);  }  };  thread.start();  }   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/2773496)[improve this answer](http://stackoverflow.com/posts/2773496/edit) | [edited Jul 22 '13 at 0:17](http://stackoverflow.com/posts/2773496/revisions) | answered May 5 '10 at 13:16  [[https://www.gravatar.com/avatar/df91e299b5afa987f5882c36e0169a9a?s=32&d=identicon&r=PG](http://stackoverflow.com/users/57490/talklittle)](http://stackoverflow.com/users/57490/talklittle)  [TalkLittle](http://stackoverflow.com/users/57490/talklittle)  **5,452**33230 | | | |
|  | | | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 2 |  | | you can create generations like hard-generation and softgeneration. you can fix a time clear cache will clear all images that were not accessed in 3 sec.. you can have a look at google shelves project – [AZ\_](http://stackoverflow.com/users/185022/az) [Jan 18 '11 at 8:06](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment5215389_2773496) | | |  |  | | --- | --- | |  |  | | [developer.android.com/reference/java/lang/ref/…](http://developer.android.com/reference/java/lang/ref/SoftReference.html) SoftReference doc has a note about caching, see "Avoid Soft References for Caching" section. Most applications should use an android.util.LruCache instead of soft references. – [vokilam](http://stackoverflow.com/users/317928/vokilam) [Feb 26 '13 at 11:48](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment21222753_2773496) | | |  |  | | --- | --- | |  |  | | I admire your code but now in the new Android O/S there is 'aggressive ' garbage collecting. Holding a weak reference does not make any sense to me. – [j2emanue](http://stackoverflow.com/users/835883/j2emanue) [Jul 21 '13 at 17:54](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment25925182_2773496) | | |  |  | | --- | --- | |  |  | | @j2emanue You are right, as I tried to indicate at the top of my answer, SoftReferences are garbage collected too quickly. I'll try to edit this answer to make that even clearer. – [TalkLittle](http://stackoverflow.com/users/57490/talklittle) [Jul 22 '13 at 0:13](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment25930935_2773496) |   add a comment |
| up vote72down vote | | High performance loader - after examining the methods suggested here, I used [Ben's solution](http://negativeprobability.blogspot.com/2011/08/lazy-loading-of-images-in-listview.html) with some changes -   1. I realized that working with drawables is faster that with bitmaps so I uses drawables instead 2. Using SoftReference is great, but it makes the cached image to be deleted too often, so I added a Linked list that holds images references, preventing from the image to be deleted, until it reached a predefined size 3. To open the InputStream I used java.net.URLConnection which allows me to use web cache (you need to set a response cache first, but that's another story)   My code:  import java.util.Map;  import java.util.HashMap;  import java.util.LinkedList;  import java.util.Collections;  import java.util.WeakHashMap;  import java.lang.ref.SoftReference;  import java.util.concurrent.Executors;  import java.util.concurrent.ExecutorService;  import android.graphics.drawable.Drawable;  import android.widget.ImageView;  import android.os.Handler;  import android.os.Message;  import java.io.InputStream;  import java.net.MalformedURLException;  import java.io.IOException;  import java.net.URL;  import java.net.URLConnection;  public class DrawableBackgroundDownloader {  private final Map<String, SoftReference<Drawable>> mCache = new HashMap<String, SoftReference<Drawable>>();  private final LinkedList <Drawable> mChacheController = new LinkedList <Drawable> ();  private ExecutorService mThreadPool;  private final Map<ImageView, String> mImageViews = Collections.synchronizedMap(new WeakHashMap<ImageView, String>());  public static int MAX\_CACHE\_SIZE = 80;  public int THREAD\_POOL\_SIZE = 3;  /\*\*  \* Constructor  \*/  public DrawableBackgroundDownloader() {  mThreadPool = Executors.newFixedThreadPool(THREAD\_POOL\_SIZE);  }  /\*\*  \* Clears all instance data and stops running threads  \*/  public void Reset() {  ExecutorService oldThreadPool = mThreadPool;  mThreadPool = Executors.newFixedThreadPool(THREAD\_POOL\_SIZE);  oldThreadPool.shutdownNow();  mChacheController.clear();  mCache.clear();  mImageViews.clear();  }  public void loadDrawable(final String url, final ImageView imageView,Drawable placeholder) {  mImageViews.put(imageView, url);  Drawable drawable = getDrawableFromCache(url);  // check in UI thread, so no concurrency issues  if (drawable != null) {  //Log.d(null, "Item loaded from mCache: " + url);  imageView.setImageDrawable(drawable);  } else {  imageView.setImageDrawable(placeholder);  queueJob(url, imageView, placeholder);  }  }  private Drawable getDrawableFromCache(String url) {  if (mCache.containsKey(url)) {  return mCache.get(url).get();  }  return null;  }  private synchronized void putDrawableInCache(String url,Drawable drawable) {  int chacheControllerSize = mChacheController.size();  if (chacheControllerSize > MAX\_CACHE\_SIZE)  mChacheController.subList(0, MAX\_CACHE\_SIZE/2).clear();  mChacheController.addLast(drawable);  mCache.put(url, new SoftReference<Drawable>(drawable));  }  private void queueJob(final String url, final ImageView imageView,final Drawable placeholder) {  /\* Create handler in UI thread. \*/  final Handler handler = new Handler() {  @Override  public void handleMessage(Message msg) {  String tag = mImageViews.get(imageView);  if (tag != null && tag.equals(url)) {  if (imageView.isShown())  if (msg.obj != null) {  imageView.setImageDrawable((Drawable) msg.obj);  } else {  imageView.setImageDrawable(placeholder);  //Log.d(null, "fail " + url);  }  }  }  };  mThreadPool.submit(new Runnable() {  @Override  public void run() {  final Drawable bmp = downloadDrawable(url);  // if the view is not visible anymore, the image will be ready for next time in cache  if (imageView.isShown())  {  Message message = Message.obtain();  message.obj = bmp;  //Log.d(null, "Item downloaded: " + url);  handler.sendMessage(message);  }  }  });  }  private Drawable downloadDrawable(String url) {  try {  InputStream is = getInputStream(url);  Drawable drawable = Drawable.createFromStream(is, url);  putDrawableInCache(url,drawable);  return drawable;  } catch (MalformedURLException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  return null;  }  private InputStream getInputStream(String urlString) throws MalformedURLException, IOException {  URL url = new URL(urlString);  URLConnection connection;  connection = url.openConnection();  connection.setUseCaches(true);  connection.connect();  InputStream response = connection.getInputStream();  return response;  }  }   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/7861011)[improve this answer](http://stackoverflow.com/posts/7861011/edit) | [edited May 14 '12 at 9:37](http://stackoverflow.com/posts/7861011/revisions) | community wiki  [6 revs, 4 users 93%](http://stackoverflow.com/posts/7861011/revisions) [Pinhassi](http://stackoverflow.com/users/475472) | | |
|  | | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Works great! BTW there's a typo in the class name. – [Mullins](http://stackoverflow.com/users/752497/mullins) [Dec 7 '11 at 14:25](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment10396883_7861011) | | |  |  | | --- | --- | | 4 |  | | In case it saves someone else the time: import java.util.Map; import java.util.HashMap; import java.util.LinkedList; import java.util.Collections; import java.util.WeakHashMap; import java.lang.ref.SoftReference; import java.util.concurrent.Executors; import java.util.concurrent.ExecutorService; import android.graphics.drawable.Drawable; import android.widget.ImageView; import android.os.Handler; import android.os.Message; import java.io.InputStream; import java.net.MalformedURLException; import java.io.IOException; import java.net.URL; import java.net.URLConnection; – [Michael Reed](http://stackoverflow.com/users/381669/michael-reed) [Jan 8 '12 at 5:01](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment10940726_7861011) | | |  |  | | --- | --- | |  |  | | Thanks very much, this is a nice implementation. I also put a different placeholder for when the drawable is being loaded so the user can get some feedback. – [Juan Hernandez](http://stackoverflow.com/users/132029/juan-hernandez) [Feb 7 '12 at 15:53](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment11548447_7861011) | | |  |  | | --- | --- | |  |  | | Also I think is better to use a LIFO queue in the executorService (mThreadPool) instead of the default FIFO so last images requested (which probably are the visible ones) are loaded first. See[stackoverflow.com/questions/4620061/how-to-create-lifo-execu‌​tor](http://stackoverflow.com/questions/4620061/how-to-create-lifo-executor) – [Juan Hernandez](http://stackoverflow.com/users/132029/juan-hernandez) [Feb 7 '12 at 16:19](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment11549030_7861011) | | |  |  | | --- | --- | | 8 |  | | @MichaelReed, in case you're an Eclipse user, I recommend using Ctrl-Shift-O (that's the letter O, not the number 0). It automates the process of adding imports and organizes them for you. If you're on a Mac, use Command-Shift-O instead. – [SilithCrowe](http://stackoverflow.com/users/387781/silithcrowe) [Mar 20 '12 at 15:21](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment12463204_7861011) |   [show **6** more comments](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote65down vote | I have followed this Android Training and I think it does an excellent job at downloading images without blocking the main UI. It also handles caching and dealing with scrolling through many images: [Loading Large Bitmaps Efficiently](http://developer.android.com/training/displaying-bitmaps/load-bitmap.html)  http://i.stack.imgur.com/zqoX2.jpg   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/14062638)[improve this answer](http://stackoverflow.com/posts/14062638/edit) | [edited May 2 '13 at 21:01](http://stackoverflow.com/posts/14062638/revisions)  [[https://www.gravatar.com/avatar/a88177d67944af05c88c2907aa329558?s=32&d=identicon&r=PG](http://stackoverflow.com/users/1134705/jnthnjns)](http://stackoverflow.com/users/1134705/jnthnjns)  [jnthnjns](http://stackoverflow.com/users/1134705/jnthnjns)  **7,308**43052 | answered Dec 27 '12 at 23:27  [[https://www.gravatar.com/avatar/6fc6e07b4af580b1424c082536570dd4?s=32&d=identicon&r=PG](http://stackoverflow.com/users/502671/toobsco42)](http://stackoverflow.com/users/502671/toobsco42)  [toobsco42](http://stackoverflow.com/users/502671/toobsco42)  **2,165**114867 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | I'm sorry, I only pointed to a single class for the Google IO app (and I'm too late to edit). You should really study all their image loading and caching utility classes that you can find in the [same package as the cache class](http://goo.gl/1TGHi). – [mkuech](http://stackoverflow.com/users/297603/mkuech) [Jan 31 '13 at 5:28](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment20415999_14062638) | | |  |  | | --- | --- | |  |  | | Would anyone recommend grabbing the DiskLruCache, Image\*.java files from the iosched app's util folder to help with handling image loading/caching for list view? I mean it's definitely worth following the online Developer guides on the subject but these classes (from iosched) go a little further with the pattern. – [Gautam](http://stackoverflow.com/users/66044/gautam)[Apr 25 '13 at 16:58](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment23198597_14062638) |   add a comment |
| up vote59down vote | **Picasso**  Use Jake Wharton's Picasso Library. (A Perfect ImageLoading Library form the developer of ActionBarSherlock)  A powerful image downloading and caching library for Android.  Images add much-needed context and visual flair to Android applications. Picasso allows for hassle-free image loading in your application—often in one line of code!  Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);  Many common pitfalls of image loading on Android are handled automatically by Picasso:  Handling ImageView recycling and download cancellation in an adapter. Complex image transformations with minimal memory use. Automatic memory and disk caching.  [Picasso Jake Wharton's Library](http://square.github.io/picasso/)  **Glide**  Glide is a fast and efficient open source media management framework for Android that wraps media decoding, memory and disk caching, and resource pooling into a simple and easy to use interface.  Glide supports fetching, decoding, and displaying video stills, images, and animated GIFs. Glide includes a flexible api that allows developers to plug in to almost any network stack. By default Glide uses a custom HttpUrlConnection based stack, but also includes utility libraries plug in to Google's Volley project or Square's OkHttp library instead.  Glide.with(this).load("http://goo.gl/h8qOq7").into(imageView);  Glide's primary focus is on making scrolling any kind of a list of images as smooth and fast as possible, but Glide is also effective for almost any case where you need to fetch, resize, and display a remote image.  [Glide Image Loading Library](https://github.com/bumptech/glide)  **Fresco by Facebook**  Fresco is a powerful system for displaying images in Android applications.  Fresco takes care of image loading and display, so you don't have to. It will load images from the network, local storage, or local resources, and display a placeholder until the image has arrived. It has two levels of cache; one in memory and another in internal storage.  [Fresco Github](https://github.com/facebook/fresco)  In Android 4.x and lower, Fresco puts images in a special region of Android memory. This lets your application run faster - and suffer the dreaded OutOfMemoryError much less often.  [Fresco Documentation](http://frescolib.org/docs/index.html#_)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/22862991)[improve this answer](http://stackoverflow.com/posts/22862991/edit) | [edited Aug 11 '15 at 18:02](http://stackoverflow.com/posts/22862991/revisions) | answered Apr 4 '14 at 12:35  [[https://i.stack.imgur.com/rNeHp.jpg?s=32&g=1](http://stackoverflow.com/users/2767181/ashwin-s-ashok)](http://stackoverflow.com/users/2767181/ashwin-s-ashok)  [Ashwin S Ashok](http://stackoverflow.com/users/2767181/ashwin-s-ashok)  **1,703**11122 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote44down vote | I've written a tutorial that explains how to do lazy-loading of images in a listview. I go into some detail about the issues of recycling and concurrency. I also use a fixed thread pool to prevent spawning a lot of threads.  [Lazy loading of images in Listview Tutorial](http://negativeprobability.blogspot.com/2011/08/lazy-loading-of-images-in-listview.html)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/7214688)[improve this answer](http://stackoverflow.com/posts/7214688/edit) | [edited Jul 4 '13 at 20:40](http://stackoverflow.com/posts/7214688/revisions)  [[https://graph.facebook.com/645215287/picture?type=large](http://stackoverflow.com/users/1112120/trying-tobemyself)](http://stackoverflow.com/users/1112120/trying-tobemyself)  [Trying Tobemyself](http://stackoverflow.com/users/1112120/trying-tobemyself)  **2,738**21637 | answered Aug 27 '11 at 12:54  [[https://www.gravatar.com/avatar/c9414b15cfad914b3e8af1bf40f01c13?s=32&d=identicon&r=PG](http://stackoverflow.com/users/314784/ben-ruijl)](http://stackoverflow.com/users/314784/ben-ruijl)  [Ben Ruijl](http://stackoverflow.com/users/314784/ben-ruijl)  **2,445**11529 | |
|  | add a comment |
| up vote31down vote | The way I do it is by launching a thread to download the images in the background and hand it a callback for each list item. When an image is finished downloading it calls the callback which updates the view for the list item.  This method doesn't work very well when you're recycling views however.   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/543018)[improve this answer](http://stackoverflow.com/posts/543018/edit) | answered Feb 12 '09 at 20:07  [[https://www.gravatar.com/avatar/a8f25968cbb72db2d75a37ee7bd84c7b?s=32&d=identicon&r=PG](http://stackoverflow.com/users/24590/jasonhudgins)](http://stackoverflow.com/users/24590/jasonhudgins)  [jasonhudgins](http://stackoverflow.com/users/24590/jasonhudgins)  **2,171**11719 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | using a thread for each image is the approach I use as well. If you separate your model from your view you can persist the model outside the Activity (like in your 'application' class) to keep them cached. Beware of running out of resources if you have many images. – [James A Wilson](http://stackoverflow.com/users/13892/james-a-wilson) [Feb 15 '09 at 0:40](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment361774_543018) | | |  |  | | --- | --- | |  |  | | can you please elaborate. I am new to android development. Thanks for the tips though – [lostInTransit](http://stackoverflow.com/users/46297/lostintransit) [Feb 15 '09 at 14:23](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment362514_543018) | | |  |  | | --- | --- | | 12 |  | | Starting a new thread for each image is not an effective solution. You can end up with a lot of threads in memory and freezing UI. – [Fedor](http://stackoverflow.com/users/95313/fedor) [Jul 1 '10 at 0:04](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment3242775_543018) | | |  |  | | --- | --- | |  |  | | Fedor, agreed, I usually use a queue and a thread pool, that's the best way to go imo. – [jasonhudgins](http://stackoverflow.com/users/24590/jasonhudgins) [Jul 4 '10 at 22:48](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment3269578_543018) |   add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote27down vote | [**Picasso**](http://square.github.io/picasso/) allows for hassle-free image loading in your application—often in one line of code!  Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);  Many common pitfalls of image loading on Android are handled automatically by [**Picasso**](http://square.github.io/picasso/):   * *Handling ImageView recycling and download cancelation in an adapter.* * *Complex image transformations with minimal memory use.* * *Automatic memory and disk caching.*   enter image description here   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/23798872)[improve this answer](http://stackoverflow.com/posts/23798872/edit) | [edited Dec 10 '14 at 7:23](http://stackoverflow.com/posts/23798872/revisions) | answered May 22 '14 at 6:00  [[https://i.stack.imgur.com/fPHPI.jpg?s=32&g=1](http://stackoverflow.com/users/3535286/chiragkyada)](http://stackoverflow.com/users/3535286/chiragkyada)  [chiragkyada](http://stackoverflow.com/users/3535286/chiragkyada)  **1,326**716 | |
|  | add a comment |
| up vote25down vote | I just want to add one more good example, [*XML Adapters*](http://developer.android.com/resources/samples/XmlAdapters/index.html). As it's is used by Google and I am also using the same logic to avoid an OutOfMemory error.  Basically [this ImageDownloader](http://developer.android.com/resources/samples/XmlAdapters/src/com/example/android/xmladapters/ImageDownloader.html) is your answer (as it covers most of your requirements). Some you can also implement in that.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/8503340)[improve this answer](http://stackoverflow.com/posts/8503340/edit) | [edited Nov 22 '12 at 19:32](http://stackoverflow.com/posts/8503340/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Dec 14 '11 at 10:57  [[https://i.stack.imgur.com/SdRZF.jpg?s=32&g=1](http://stackoverflow.com/users/486139/arslan)](http://stackoverflow.com/users/486139/arslan)  [Arslan](http://stackoverflow.com/users/486139/arslan)  **12.1k**106390 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | ImageDownloader class not get complied: see the solution below [code.google.com/p/parleys-android-nextgen/issues/detail?id=1](http://code.google.com/p/parleys-android-nextgen/issues/detail?id=1) – [Sam](http://stackoverflow.com/users/322111/sam) [Feb 4 '12 at 4:48](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment11487098_8503340) | | |  |  | | --- | --- | |  |  | | Tried this code, it's giving me OOM – [Henrique de Sousa](http://stackoverflow.com/users/1420441/henrique-de-sousa) [Oct 4 '13 at 8:28](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment28370723_8503340) |   add a comment |

|  |  |  |  |
| --- | --- | --- | --- |
| up vote18down vote | This is a common problem on Android that has been solved in many ways by many people. In my opinion the best solution I've seen is the relatively new library called [Picasso](http://square.github.io/picasso/). Here are the highlights:   * Open source, but headed up by Jake Wharton of [ActionBarSherlock](http://actionbarsherlock.com/) fame. * Asynchronously load images from network or app resources with one line of code * Automatic ListView detection * Automatic disk and memory caching * Can do custom transformations * Lots of configurable options * Super simple API * Frequently updated  |  |  | | --- | --- | | [share](http://stackoverflow.com/a/19392742)[improve this answer](http://stackoverflow.com/posts/19392742/edit) | answered Oct 15 '13 at 23:22  [[https://www.gravatar.com/avatar/3fa01cb646a69392d911bfd876ff884e?s=32&d=identicon&r=PG](http://stackoverflow.com/users/758458/howettl)](http://stackoverflow.com/users/758458/howettl)  [howettl](http://stackoverflow.com/users/758458/howettl)  **5,480**113876 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Picasso is the best solution. – [atifali](http://stackoverflow.com/users/3654735/atifali) [Jul 14 '15 at 8:46](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment50777993_19392742) |   add a comment |
| up vote17down vote | I have been using NetworkImageView from the new Android Volley Library com.android.volley.toolbox.NetworkImageView, and it seems to be working pretty well. Apparently, this is the same view that is used in [Google Play](http://en.wikipedia.org/wiki/Google_Play) and other new Google applications. Definitely worth checking out.   * [Google I/O 2013 volley image cache tutorial](http://blogs.captechconsulting.com/blog/raymond-robinson/google-io-2013-volley-image-cache-tutorial) * [Developers Google events](https://developers.google.com/events/io/sessions/325304728)  |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/17090817)[improve this answer](http://stackoverflow.com/posts/17090817/edit) | [edited Mar 16 '14 at 13:28](http://stackoverflow.com/posts/17090817/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Jun 13 '13 at 15:17  [[https://www.gravatar.com/avatar/92f3edd986d10f68997d1aad4d0fb250?s=32&d=identicon&r=PG](http://stackoverflow.com/users/439405/droidment)](http://stackoverflow.com/users/439405/droidment)  [droidment](http://stackoverflow.com/users/439405/droidment)  **395**36 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | I think this is the best solution - the other answers are very old - volley is realy fast and combined with jake warthons disklrucache its a perfekt solution - i tried a lot of others but not one is stable and fast as volley – [Alexander Sidikov](http://stackoverflow.com/users/1847899/alexander-sidikov) [Sep 7 '14 at 22:19](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment40199139_17090817) |   add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote15down vote | I think this issue is very popular among Android developers, and there are plenty of such libraries that claims to resolve this issue, but only a few of them seems to be on the mark. [AQuery](http://code.google.com/p/android-query/) is one such library, but it is better than most of them in all aspects and is worth trying for.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/13826316)[improve this answer](http://stackoverflow.com/posts/13826316/edit) | [edited Mar 16 '14 at 13:19](http://stackoverflow.com/posts/13826316/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Dec 11 '12 at 18:33  [[https://www.gravatar.com/avatar/b12c0507c70b136ff75e3b572ed8bd9a?s=32&d=identicon&r=PG](http://stackoverflow.com/users/1226111/ritesh-kumar-dubey)](http://stackoverflow.com/users/1226111/ritesh-kumar-dubey)  [Ritesh Kumar Dubey](http://stackoverflow.com/users/1226111/ritesh-kumar-dubey)  **401**518 | |
|  | add a comment |
| up vote14down vote | Well, image loading time from the Internet has many solutions. You may also use the library[Android-Query](https://code.google.com/p/android-query/wiki/ImageLoading). It will give you all the required activity. Make sure what you want to do and read the library wiki page. And solve the image loading restriction.  This is my code:  @Override  public View getView(int position, View convertView, ViewGroup parent) {  View v = convertView;  if (v == null) {  LayoutInflater vi = (LayoutInflater)getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  v = vi.inflate(R.layout.row, null);  }  ImageView imageview = (ImageView) v.findViewById(R.id.icon);  AQuery aq = new AQuery(convertView);  String imageUrl = "http://www.vikispot.com/z/images/vikispot/android-w.png";  aq.id(imageview).progress(this).image(imageUrl, true, true, 0, 0, new BitmapAjaxCallback() {  @Override  public void callback(String url, ImageView iv, Bitmap bm, AjaxStatus status) {  iv.setImageBitmap(bm);  }  ));  return v;  }  It should be solve your lazy loading problem.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/14827292)[improve this answer](http://stackoverflow.com/posts/14827292/edit) | [edited Apr 24 '15 at 21:10](http://stackoverflow.com/posts/14827292/revisions)  [[https://www.gravatar.com/avatar/cd27f10d8d60171d5a67306b7ccf94e0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/759007/ziem)](http://stackoverflow.com/users/759007/ziem)  [Ziem](http://stackoverflow.com/users/759007/ziem)  **3,032**43061 | answered Feb 12 '13 at 7:10  [[https://i.stack.imgur.com/dinhu.jpg?s=32&g=1](http://stackoverflow.com/users/1737833/rahul-rawat)](http://stackoverflow.com/users/1737833/rahul-rawat)  [Rahul Rawat](http://stackoverflow.com/users/1737833/rahul-rawat)  **306**1728 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote13down vote | Have a look at [Shutterbug](https://github.com/applidium/Shutterbug), Applidium's lightweight SDWebImage (a nice library on iOS) port to Android. It supports asynchronous caching, stores failed URLs, handles concurrency well, and helpful subclasses are included.  Pull requests (and bug reports) are welcome, too!   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/12915019)[improve this answer](http://stackoverflow.com/posts/12915019/edit) | [edited Nov 2 '12 at 17:42](http://stackoverflow.com/posts/12915019/revisions) | answered Oct 16 '12 at 12:43  [[https://www.gravatar.com/avatar/c98e7e3b538d8ff68a4f53f50e5771d4?s=32&d=identicon&r=PG](http://stackoverflow.com/users/767857/pnollet)](http://stackoverflow.com/users/767857/pnollet)  [pnollet](http://stackoverflow.com/users/767857/pnollet)  **1,387**922 | |
|  | add a comment |
| up vote10down vote | Novoda also has a great [lazy image loading library](http://androidimageloader.com/) and many apps like Songkick, Podio, SecretDJ and ImageSearch use their library.  Their library is hosted [here](https://github.com/novoda/ImageLoader) on Github and they have a pretty active [issues tracker](https://github.com/novoda/ImageLoader/issues) as well. Their project seems to be pretty active too, with over 300+ commits at the time of writing this reply.   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/14292671)[improve this answer](http://stackoverflow.com/posts/14292671/edit) | answered Jan 12 '13 at 11:00  [[https://www.gravatar.com/avatar/ad94879495e21b7362d0d2c845f44bc0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/614451/soham)](http://stackoverflow.com/users/614451/soham)  [Soham](http://stackoverflow.com/users/614451/soham)  **3,859**22141 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | Actually Novoda is a great library but...sometimes you don't need a huge library and only a simple approach of the solution. That is why LazyList in Github is so good, if your apps only shows an image in a listView and is not the main feature of your app, just another activity I would prefer to use something lightier. Otherwise if you know that you will have to use often and is part of the core, try Novoda. – [Nicolas Jafelle](http://stackoverflow.com/users/682216/nicolas-jafelle) [Apr 15 '13 at 15:34](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment22848433_14292671) |   add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote10down vote | [DroidParts](http://droidparts.org/) has [ImageFetcher](https://github.com/yanchenko/droidparts/blob/master/droidparts/src/org/droidparts/net/image/ImageFetcher.java) that requires zero configuration to get started.   * Uses a disk & in-memory [Least Recently Used](https://en.wikipedia.org/wiki/Cache_algorithms#Examples) (LRU) cache. * Efficiently decodes images. * Supports modifying bitmaps in background thread. * Has simple cross-fade. * Has image loading progress callback.   Clone [DroidPartsGram](https://github.com/yanchenko/droidparts/tree/master/droidparts-samples/DroidPartsGram) for an example:  Enter image description here   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/16396081)[improve this answer](http://stackoverflow.com/posts/16396081/edit) | [edited Mar 16 '14 at 13:26](http://stackoverflow.com/posts/16396081/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered May 6 '13 at 9:48  [[https://www.gravatar.com/avatar/6901887336e94dfc45e6208206dddbae?s=32&d=identicon&r=PG](http://stackoverflow.com/users/15187/yanchenko)](http://stackoverflow.com/users/15187/yanchenko)  [yanchenko](http://stackoverflow.com/users/15187/yanchenko)  **35.1k**28122147 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Hi, I've had a look at the code examples but i'm having issues using ImageFetcher with an ArrayAdapter, would you mind looking at my question? [stackoverflow.com/questions/21089147/…](http://stackoverflow.com/questions/21089147/recycling-issues-in-arrayadapter-using-imagefetcher) Thanks =] – [masha](http://stackoverflow.com/users/1614159/masha) [Jan 13 '14 at 10:37](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment31721259_16396081) |   add a comment |
| up vote8down vote | Check my fork of [LazyList](https://github.com/nicolasjafelle/LazyList). Basically, I improve the LazyList by delaying the call of the ImageView and create two methods:   1. When you need to put something like "Loading image..." 2. When you need to show the downloaded image.   I also improved the ImageLoader by implementing a [singleton](http://en.wikipedia.org/wiki/Singleton_pattern) in this object.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/14844000)[improve this answer](http://stackoverflow.com/posts/14844000/edit) | [edited Mar 16 '14 at 13:24](http://stackoverflow.com/posts/14844000/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Feb 12 '13 at 23:46  [[https://www.gravatar.com/avatar/752644e7386956fde1e497e0b27546f3?s=32&d=identicon&r=PG](http://stackoverflow.com/users/682216/nicolas-jafelle)](http://stackoverflow.com/users/682216/nicolas-jafelle)  [Nicolas Jafelle](http://stackoverflow.com/users/682216/nicolas-jafelle)  **833**1915 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote8down vote | Just a quick tip for someone who is in indecision regarding what library to use for lazy-loading images:  There are four basic ways.   1. DIY => Not the best solution but for a few images and if you want to go without the hassle of using others libraries 2. Volley's Lazy Loading library => From guys at android. It is nice and everything but is poorly documented and hence is a problem to use. 3. Picasso: A simple solution that just works, you can even specify the exact image size you want to bring in. It is very simple to use but might not be very "performant" for apps that has to deal with humongous amounts of images. 4. UIL: The best way to lazy load images. You can cache images(you need permission of course), initialize the loader once, then have your work done. The most mature asynchronous image loading library I have ever seen so far.  |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/25104563)[improve this answer](http://stackoverflow.com/posts/25104563/edit) | [edited Dec 2 '15 at 5:23](http://stackoverflow.com/posts/25104563/revisions) | answered Aug 3 '14 at 12:19  [[https://i.stack.imgur.com/cMJah.jpg?s=32&g=1](http://stackoverflow.com/users/2439443/bijay-koirala)](http://stackoverflow.com/users/2439443/bijay-koirala)  [Bijay Koirala](http://stackoverflow.com/users/2439443/bijay-koirala)  **320**28 | |
|  | add a comment |
| up vote6down vote | I can recommend a different way that works like a charm: Android Query.  You can download that [JAR](http://en.wikipedia.org/wiki/JAR_%28file_format%29) file from [here](http://code.google.com/p/android-query/downloads/list)  AQuery androidAQuery = new AQuery(this);  As an example:  androidAQuery.id(YOUR IMAGEVIEW).image(YOUR IMAGE TO LOAD, true, true, getDeviceWidth(), ANY DEFAULT IMAGE YOU WANT TO SHOW);  It's very fast and accurate, and using this you can find many more features like animation when loading, getting a bitmap (if needed), etc.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/17917897)[improve this answer](http://stackoverflow.com/posts/17917897/edit) | [edited Mar 16 '14 at 13:36](http://stackoverflow.com/posts/17917897/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Jul 29 '13 at 7:06  [[https://www.gravatar.com/avatar/478da2fe3e240a32c21865340932f9c6?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/644603/pratik-dasa)](http://stackoverflow.com/users/644603/pratik-dasa)  [Pratik Dasa](http://stackoverflow.com/users/644603/pratik-dasa)  **4,582**31532 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote6down vote | Give [Aquery](https://code.google.com/p/android-query/wiki/ImageLoading) a try. It has amazingly simple methods to load and cache images asynchronously.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/20611648)[improve this answer](http://stackoverflow.com/posts/20611648/edit) | [edited Mar 16 '14 at 13:37](http://stackoverflow.com/posts/20611648/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Dec 16 '13 at 13:04  [[https://www.gravatar.com/avatar/c8c4d0736a7ed9a3a52ec9d3d8299007?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/2779311/user2779311)](http://stackoverflow.com/users/2779311/user2779311)  [user2779311](http://stackoverflow.com/users/2779311/user2779311)  **347**717 | |
|  | add a comment |
| up vote6down vote | You must try this Universal Loader is best. I am using this after done many RnD on lazy loading .  [Universal Image Loader](https://github.com/nostra13/Android-Universal-Image-Loader)  **Features**   * Multithread image loading (async or sync) * Wide customization of ImageLoader's configuration (thread executors, downloader, decoder, memory and disk cache, display image options, etc.) * Many customization options for every display image call (stub images, caching switch, decoding options, Bitmap processing and displaying, etc.) * Image caching in memory and/or on disk (device's file system or SD card) * Listening loading process (including downloading progress)   Android 2.0+ support  enter image description here   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/28344352)[improve this answer](http://stackoverflow.com/posts/28344352/edit) | answered Feb 5 '15 at 12:45  [[https://www.gravatar.com/avatar/1f1190919aff7dba6fc69919ec77a44d?s=32&d=identicon&r=PG](http://stackoverflow.com/users/712868/girishce26)](http://stackoverflow.com/users/712868/girishce26)  [girishce26](http://stackoverflow.com/users/712868/girishce26)  **460**1822 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote6down vote | All above code have their own worth but with my personal experience just give a try with Picasso.  **Picasso** is a library specifically for this purpose, in-fact it will manage cache and all other network operations automatically.You will have to add library in your project and just write a single line of code to load image from remote URL.  Please visit here : <http://code.tutsplus.com/tutorials/android-sdk-working-with-picasso--cms-22149>   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/31322938)[improve this answer](http://stackoverflow.com/posts/31322938/edit) | [edited Sep 26 '15 at 18:54](http://stackoverflow.com/posts/31322938/revisions) | answered Jul 9 '15 at 16:16  [[https://i.stack.imgur.com/npNhT.png?s=32&g=1](http://stackoverflow.com/users/5076491/akbar)](http://stackoverflow.com/users/5076491/akbar)  [Akbar](http://stackoverflow.com/users/5076491/akbar)  **347**310 | |
|  | add a comment |
| up vote5down vote | [URLImageViewHelper](https://github.com/koush/UrlImageViewHelper) is an amazing library that helps you to do that.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/21373668)[improve this answer](http://stackoverflow.com/posts/21373668/edit) | [edited Mar 16 '14 at 13:37](http://stackoverflow.com/posts/21373668/revisions)  [[https://i.stack.imgur.com/RIZKi.png?s=32&g=1](http://stackoverflow.com/users/63550/peter-mortensen)](http://stackoverflow.com/users/63550/peter-mortensen)  [Peter Mortensen](http://stackoverflow.com/users/63550/peter-mortensen)  **10.3k**1369107 | answered Jan 27 '14 at 5:33  [[https://www.gravatar.com/avatar/a0316081e0ae776ba7e5e5cd749d01e3?s=32&d=identicon&r=PG](http://stackoverflow.com/users/2170871/diegoalt)](http://stackoverflow.com/users/2170871/diegoalt)  [DiegoAlt](http://stackoverflow.com/users/2170871/diegoalt)  **71**22 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote4down vote | public class ImageDownloader {  Map<String, Bitmap> imageCache;  public ImageDownloader() {  imageCache = new HashMap<String, Bitmap>();  }  // download function  public void download(String url, ImageView imageView) {  if (cancelPotentialDownload(url, imageView)) {  // Caching code right here  String filename = String.valueOf(url.hashCode());  File f = new File(getCacheDirectory(imageView.getContext()),  filename);  // Is the bitmap in our memory cache?  Bitmap bitmap = null;  bitmap = (Bitmap) imageCache.get(f.getPath());  if (bitmap == null) {  bitmap = BitmapFactory.decodeFile(f.getPath());  if (bitmap != null) {  imageCache.put(f.getPath(), bitmap);  }  }  // No? download it  if (bitmap == null) {  try {  BitmapDownloaderTask task = new BitmapDownloaderTask(  imageView);  DownloadedDrawable downloadedDrawable = new DownloadedDrawable(  task);  imageView.setImageDrawable(downloadedDrawable);  task.execute(url);  } catch (Exception e) {  Log.e("Error==>", e.toString());  }  } else {  // Yes? set the image  imageView.setImageBitmap(bitmap);  }  }  }  // cancel a download (internal only)  private static boolean cancelPotentialDownload(String url,  ImageView imageView) {  BitmapDownloaderTask bitmapDownloaderTask = getBitmapDownloaderTask(imageView);  if (bitmapDownloaderTask != null) {  String bitmapUrl = bitmapDownloaderTask.url;  if ((bitmapUrl == null) || (!bitmapUrl.equals(url))) {  bitmapDownloaderTask.cancel(true);  } else {  // The same URL is already being downloaded.  return false;  }  }  return true;  }  // gets an existing download if one exists for the imageview  private static BitmapDownloaderTask getBitmapDownloaderTask(  ImageView imageView) {  if (imageView != null) {  Drawable drawable = imageView.getDrawable();  if (drawable instanceof DownloadedDrawable) {  DownloadedDrawable downloadedDrawable = (DownloadedDrawable) drawable;  return downloadedDrawable.getBitmapDownloaderTask();  }  }  return null;  }  // our caching functions  // Find the dir to save cached images  private static File getCacheDirectory(Context context) {  String sdState = android.os.Environment.getExternalStorageState();  File cacheDir;  if (sdState.equals(android.os.Environment.MEDIA\_MOUNTED)) {  File sdDir = android.os.Environment.getExternalStorageDirectory();  // TODO : Change your diretcory here  cacheDir = new File(sdDir, "data/ToDo/images");  } else  cacheDir = context.getCacheDir();  if (!cacheDir.exists())  cacheDir.mkdirs();  return cacheDir;  }  private void writeFile(Bitmap bmp, File f) {  FileOutputStream out = null;  try {  out = new FileOutputStream(f);  bmp.compress(Bitmap.CompressFormat.PNG, 80, out);  } catch (Exception e) {  e.printStackTrace();  } finally {  try {  if (out != null)  out.close();  } catch (Exception ex) {  }  }  }  // download asynctask  public class BitmapDownloaderTask extends AsyncTask<String, Void, Bitmap> {  private String url;  private final WeakReference<ImageView> imageViewReference;  public BitmapDownloaderTask(ImageView imageView) {  imageViewReference = new WeakReference<ImageView>(imageView);  }  @Override  // Actual download method, run in the task thread  protected Bitmap doInBackground(String... params) {  // params comes from the execute() call: params[0] is the url.  url = (String) params[0];  return downloadBitmap(params[0]);  }  @Override  // Once the image is downloaded, associates it to the imageView  protected void onPostExecute(Bitmap bitmap) {  if (isCancelled()) {  bitmap = null;  }  if (imageViewReference != null) {  ImageView imageView = imageViewReference.get();  BitmapDownloaderTask bitmapDownloaderTask = getBitmapDownloaderTask(imageView);  // Change bitmap only if this process is still associated with  // it  if (this == bitmapDownloaderTask) {  imageView.setImageBitmap(bitmap);  // cache the image  String filename = String.valueOf(url.hashCode());  File f = new File(  getCacheDirectory(imageView.getContext()), filename);  imageCache.put(f.getPath(), bitmap);  writeFile(bitmap, f);  }  }  }  }  static class DownloadedDrawable extends ColorDrawable {  private final WeakReference<BitmapDownloaderTask> bitmapDownloaderTaskReference;  public DownloadedDrawable(BitmapDownloaderTask bitmapDownloaderTask) {  super(Color.WHITE);  bitmapDownloaderTaskReference = new WeakReference<BitmapDownloaderTask>(  bitmapDownloaderTask);  }  public BitmapDownloaderTask getBitmapDownloaderTask() {  return bitmapDownloaderTaskReference.get();  }  }  // the actual download code  static Bitmap downloadBitmap(String url) {  HttpParams params = new BasicHttpParams();  params.setParameter(CoreProtocolPNames.PROTOCOL\_VERSION,  HttpVersion.HTTP\_1\_1);  HttpClient client = new DefaultHttpClient(params);  final HttpGet getRequest = new HttpGet(url);  try {  HttpResponse response = client.execute(getRequest);  final int statusCode = response.getStatusLine().getStatusCode();  if (statusCode != HttpStatus.SC\_OK) {  Log.w("ImageDownloader", "Error " + statusCode  + " while retrieving bitmap from " + url);  return null;  }  final HttpEntity entity = response.getEntity();  if (entity != null) {  InputStream inputStream = null;  try {  inputStream = entity.getContent();  final Bitmap bitmap = BitmapFactory  .decodeStream(inputStream);  return bitmap;  } finally {  if (inputStream != null) {  inputStream.close();  }  entity.consumeContent();  }  }  } catch (Exception e) {  // Could provide a more explicit error message for IOException or  // IllegalStateException  getRequest.abort();  Log.w("ImageDownloader", "Error while retrieving bitmap from "  + url + e.toString());  } finally {  if (client != null) {  // client.close();  }  }  return null;  }  }   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/17716992)[improve this answer](http://stackoverflow.com/posts/17716992/edit) | [edited Oct 21 '13 at 12:21](http://stackoverflow.com/posts/17716992/revisions)  [[https://www.gravatar.com/avatar/3c6f1a94dff236a7cb6838022d621177?s=32&d=identicon&r=PG](http://stackoverflow.com/users/918472/rupesh-yadav)](http://stackoverflow.com/users/918472/rupesh-yadav)  [Rupesh Yadav](http://stackoverflow.com/users/918472/rupesh-yadav)  **6,963**43350 | answered Jul 18 '13 at 7:16  [[https://i.stack.imgur.com/L5zI0.jpg?s=32&g=1](http://stackoverflow.com/users/1647951/nikhil-gupta)](http://stackoverflow.com/users/1647951/nikhil-gupta)  [Nikhil Gupta](http://stackoverflow.com/users/1647951/nikhil-gupta)  **59**2 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 3 |  | | Any explanation? – [pivovarit](http://stackoverflow.com/users/2229438/pivovarit) [Jul 18 '13 at 7:47](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview/3068012#comment25823521_17716992) |   add a comment |
| up vote4down vote | You can try the ***Aquery* Android** library for lazy loading image and listview... The below code may help you..... [download library from here](http://code.google.com/p/android-query/downloads/detail?name=android-query-full.0.25.10.jar).  AQuery aq = new AQuery(mContext);  aq.id(R.id.image1).image("http://data.whicdn.com/images/63995806/original.jpg");   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/18592615)[improve this answer](http://stackoverflow.com/posts/18592615/edit) | [edited Mar 20 '14 at 4:22](http://stackoverflow.com/posts/18592615/revisions) | answered Sep 3 '13 at 12:41  [[https://www.gravatar.com/avatar/8df844054033768ad4c40784318f944f?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/2670249/chirag-ghori)](http://stackoverflow.com/users/2670249/chirag-ghori)  [Chirag Ghori](http://stackoverflow.com/users/2670249/chirag-ghori)  **3,176**2929 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote4down vote | I use [droidQuery](http://bit.ly/droidQuery). There are two mechanisms for loading an image from a URL. The first (shorthand) is simply:  $.with(myView).image(url);  This can be added into an ArrayAdapter's getView(...) method very easily.  The longhand method will give a lot more control, and has options not even discussed here (such as cacheing and callbacks), but a basic implementation that specifies the output size as 200px x 200px can be found here:  $.ajax(new AjaxOptions().url(url)  .type("GET")  .dataType("image")  .imageWidth(200).imageHeight(200)  .success(new Function() {  @Override  public void invoke($ droidQuery, Object... params) {  myImageView.setImageBitmap((Bitmap) params[0]);  }  })  .error(new Function() {  @Override  public void invoke($ droidQuery, Object... params) {  AjaxError e = (AjaxError) params[0];  Log.e("$", "Error " + e.status + ": " + e.error);  }  })  );   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/18281334)[improve this answer](http://stackoverflow.com/posts/18281334/edit) | [edited Apr 24 '15 at 20:51](http://stackoverflow.com/posts/18281334/revisions)  [[https://www.gravatar.com/avatar/cd27f10d8d60171d5a67306b7ccf94e0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/759007/ziem)](http://stackoverflow.com/users/759007/ziem)  [Ziem](http://stackoverflow.com/users/759007/ziem)  **3,032**43061 | answered Aug 16 '13 at 20:06  [[https://www.gravatar.com/avatar/76f3c302f6b92b09af41cb0c46de064e?s=32&d=identicon&r=PG](http://stackoverflow.com/users/763080/phil)](http://stackoverflow.com/users/763080/phil)  [Phil](http://stackoverflow.com/users/763080/phil)  **21k**1486129 | |
|  | add a comment |
| up vote3down vote | I had this issue and implemented lruCache. I believe you need API 12 and above or use the compatiblity v4 library. lurCache is fast memory, but it also has a budget, so if you're worried about that you can use a diskcache... It's all described in [*Caching Bitmaps*](http://developer.android.com/training/displaying-bitmaps/cache-bitmap.html).  I'll now provide my implementation which is a [singleton](http://en.wikipedia.org/wiki/Singleton_pattern) I call from anywhere like this:  //Where the first is a string and the other is a imageview to load.  DownloadImageTask.getInstance().loadBitmap(avatarURL, iv\_avatar);  Here's the ideal code to cache and then call the above in getView of an adapter when retrieving the web image:  public class DownloadImageTask {  private LruCache<String, Bitmap> mMemoryCache;  /\* Create a singleton class to call this from multiple classes \*/  private static DownloadImageTask instance = null;  public static DownloadImageTask getInstance() {  if (instance == null) {  instance = new DownloadImageTask();  }  return instance;  }  //Lock the constructor from public instances  private DownloadImageTask() {  // Get max available VM memory, exceeding this amount will throw an  // OutOfMemory exception. Stored in kilobytes as LruCache takes an  // int in its constructor.  final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);  // Use 1/8th of the available memory for this memory cache.  final int cacheSize = maxMemory / 8;  mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {  @Override  protected int sizeOf(String key, Bitmap bitmap) {  // The cache size will be measured in kilobytes rather than  // number of items.  return bitmap.getByteCount() / 1024;  }  };  }  public void loadBitmap(String avatarURL, ImageView imageView) {  final String imageKey = String.valueOf(avatarURL);  final Bitmap bitmap = getBitmapFromMemCache(imageKey);  if (bitmap != null) {  imageView.setImageBitmap(bitmap);  } else {  imageView.setImageResource(R.drawable.ic\_launcher);  new DownloadImageTaskViaWeb(imageView).execute(avatarURL);  }  }  private void addBitmapToMemoryCache(String key, Bitmap bitmap) {  if (getBitmapFromMemCache(key) == null) {  mMemoryCache.put(key, bitmap);  }  }  private Bitmap getBitmapFromMemCache(String key) {  return mMemoryCache.get(key);  }  /\* A background process that opens a http stream and decodes a web image. \*/  class DownloadImageTaskViaWeb extends AsyncTask<String, Void, Bitmap> {  ImageView bmImage;  public DownloadImageTaskViaWeb(ImageView bmImage) {  this.bmImage = bmImage;  }  protected Bitmap doInBackground(String... urls) {  String urldisplay = urls[0];  Bitmap mIcon = null;  try {  InputStream in = new java.net.URL(urldisplay).openStream();  mIcon = BitmapFactory.decodeStream(in);  }  catch (Exception e) {  Log.e("Error", e.getMessage());  e.printStackTrace();  }  addBitmapToMemoryCache(String.valueOf(urldisplay), mIcon);  return mIcon;  }  /\* After decoding we update the view on the main UI. \*/  protected void onPostExecute(Bitmap result) {  bmImage.setImageBitmap(result);  }  }  }   |  | | --- | | [share](http://stackoverflow.com/a/17775153)[improve this answer](http://stackoverflow.com/posts/17775153/edit) | |

|  |  |  |  |
| --- | --- | --- | --- |
| Another way to do it, is through your adapter in a thread in your getView() method :  Thread pics\_thread = new Thread(new Runnable() {  @Override  public void run() {  Bitmap bitmap = getPicture(url);  if(bitmap != null) {  runOnUiThread(new Runnable() {  @Override  public void run() {  holder.imageview.setImageBitmap(bitmap);  adapter.notifyDataSetChanged();  }  });  }  }  });  pics\_thread.start();  of course, you should always cache your images to avoid extra operations, you could put your images in a HashMap array, check if the image exists in the array, if not, proceed with the thread or else load the image from you HashMap array. Also always check that you are not leaking memory, bitmaps and drawables are often heavy on memory. It is up to you to optimize your code.   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/24339989)[improve this answer](http://stackoverflow.com/posts/24339989/edit) | [edited Apr 24 '15 at 21:01](http://stackoverflow.com/posts/24339989/revisions)  [[https://www.gravatar.com/avatar/cd27f10d8d60171d5a67306b7ccf94e0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/759007/ziem)](http://stackoverflow.com/users/759007/ziem)  [Ziem](http://stackoverflow.com/users/759007/ziem)  **3,032**43061 | answered Jun 21 '14 at 8:55  [[https://i.stack.imgur.com/nAQhj.jpg?s=32&g=1](http://stackoverflow.com/users/1079001/samet)](http://stackoverflow.com/users/1079001/samet)  [Samet](http://stackoverflow.com/users/1079001/samet)  **628**823 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | I like getting the bitmap in a different thread, of course. But the only issue I have with having this code in getView() is there will be many threads running for several images. And getView may try to load many or several images at one time. – [The Original Android](http://stackoverflow.com/users/1378963/the-original-android) [Jun 7 '15 at 8:31](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview?page=2&tab=votes#comment49441939_24339989) |   add a comment |
| up vote1down vote | You can use some third party library such as Piccaso or Volley for effective lazy loading. You can also create your own by implementing the below   1. Implement code for downloading the image from the url 2. Implement caching mechanism for storing and retrieving image(Use LruCache of android for caching)  |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/34873683)[improve this answer](http://stackoverflow.com/posts/34873683/edit) | [edited Sep 27 at 8:16](http://stackoverflow.com/posts/34873683/revisions)  [[https://i.stack.imgur.com/2jBjo.png?s=32&g=1](http://stackoverflow.com/users/1844392/piyush)](http://stackoverflow.com/users/1844392/piyush)  [Piyush](http://stackoverflow.com/users/1844392/piyush)  **18.3k**52055 | answered Jan 19 at 9:59  [[https://www.gravatar.com/avatar/bf8ba2c9b6a2b96af1b5ffddf4d02715?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/4286151/balaramnayak)](http://stackoverflow.com/users/4286151/balaramnayak)  [BalaramNayak](http://stackoverflow.com/users/4286151/balaramnayak)  **170**17 | |
|  | add a comment |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote0down vote | use below class to download and load images in listview.It caches every images once download. Also loads images ad lazy loading.  package com.fudiyoxpress.images;  import java.io.File;  import java.io.FileInputStream;  import java.io.FileNotFoundException;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;  import java.net.HttpURLConnection;  import java.net.URL;  import java.util.Collections;  import java.util.Map;  import java.util.WeakHashMap;  import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;  import android.content.Context;  import android.graphics.Bitmap;  import android.graphics.BitmapFactory;  import android.os.Handler;  import android.widget.ImageView;  import com.fudiyoxpress.R;  import com.fudiyoxpress.config.Config;  import com.fudiyoxpress.twitter.ScaleBitmap;  public class ImageLoader {  // Initialize MemoryCache  MemoryCache memoryCache = new MemoryCache();  FileCache fileCache;  Context C;  // Create Map (collection) to store image and image url in key value pair  private Map<ImageView, String> imageViews = Collections  .synchronizedMap(new WeakHashMap<ImageView, String>());  ExecutorService executorService;  // handler to display images in UI thread  Handler handler = new Handler();  public ImageLoader(Context context) {  C = context;  fileCache = new FileCache(context);  // Creates a thread pool that reuses a fixed number of  // threads operating off a shared unbounded queue.  executorService = Executors.newFixedThreadPool(5);  }  // default image show in list (Before online image download)  final int stub\_id = R.drawable.restlogoplaceholder;  public void DisplayImage(String url, ImageView imageView, Context context,  boolean header\_flag) {  Bitmap largeIcon = BitmapFactory.decodeResource(context.getResources(),  R.drawable.restlogoplaceholder);  header\_flag = false;  // Store image and url in Map  imageViews.put(imageView, url);  // Check image is stored in MemoryCache Map or not (see  // MemoryCache.java)  Bitmap bitmap = memoryCache.get(url);  if (bitmap != null) {  // if image is stored in MemoryCache Map then  // Show image in listview row  Bitmap b = ScaleBitmap  .getScaledBitmap(context, bitmap, header\_flag);  imageView.setImageBitmap(b);  } else {  // queue Photo to download from url  queuePhoto(url, imageView, header\_flag);  // Before downloading image show default image  imageView.setImageBitmap(ScaleBitmap.getScaledBitmap(context,  largeIcon, header\_flag));  }  }  private void queuePhoto(String url, ImageView imageView, boolean header\_flag) {  // Store image and url in PhotoToLoad object  PhotoToLoad p = new PhotoToLoad(url, imageView, header\_flag);  // pass PhotoToLoad object to PhotosLoader runnable class  // and submit PhotosLoader runnable to executers to run runnable  // Submits a PhotosLoader runnable task for execution  executorService.submit(new PhotosLoader(p));  }  // Task for the queue  private class PhotoToLoad {  public String url;  public ImageView imageView;  public boolean b;  public PhotoToLoad(String u, ImageView i, boolean header\_flag) {  url = u;  imageView = i;  b = header\_flag;  }  }  class PhotosLoader implements Runnable {  PhotoToLoad photoToLoad;  PhotosLoader(PhotoToLoad photoToLoad) {  this.photoToLoad = photoToLoad;  }  @Override  public void run() {  try {  // Check if image already downloaded  if (imageViewReused(photoToLoad))  return;  // download image from web url  Bitmap bmp = getBitmap(photoToLoad.url);  // set image data in Memory Cache  memoryCache.put(photoToLoad.url, bmp);  if (imageViewReused(photoToLoad))  return;  // Get bitmap to display  BitmapDisplayer bd = new BitmapDisplayer(bmp, photoToLoad);  // Causes the Runnable bd (BitmapDisplayer) to be added to the  // message queue.  // The runnable will be run on the thread to which this handler  // is attached.  // BitmapDisplayer run method will call  handler.post(bd);  } catch (Throwable th) {  // th.printStackTrace();  }  }  }  private Bitmap getBitmap(String url) {  File f = fileCache.getFile(url);  // from SD cache  // CHECK : if trying to decode file which not exist in cache return null  Bitmap b = decodeFile(f);  if (b != null)  return b;  // Download image file from web  try {  // // download the image  Bitmap bitmap = null;  URL imageURL = null;  try {  imageURL = new URL(Config.WEB\_URL + "/ServeBlob?id=" + url);  HttpURLConnection connection = (HttpURLConnection) imageURL  .openConnection();  connection.setDoInput(true);  connection.connect();  // if(!(new File(imageURL.toString())).exists())  // {  // imageURL=new URL("");  // }  InputStream inputStream = connection.getInputStream();  // Constructs a new FileOutputStream that writes to  // file  // if file not exist then it will create file  OutputStream os = new FileOutputStream(f);  // See Utils class CopyStream method  // It will each pixel from input stream and  // write pixels to output stream (file)  Utils.CopyStream(inputStream, os);  os.close();  BitmapFactory.Options options = new BitmapFactory.Options();  options.inSampleSize = 8;  bitmap = BitmapFactory.decodeStream(inputStream, null, options);  } catch (IOException e) {  // e.printStackTrace();  }  // Now file created and going to resize file with defined height  // Decodes image and scales it to reduce memory consumption  bitmap = decodeFile(f);  return bitmap;  } catch (Throwable ex) {  ex.printStackTrace();  if (ex instanceof OutOfMemoryError)  memoryCache.clear();  return null;  }  }  // Decodes image and scales it to reduce memory consumption  private Bitmap decodeFile(File f) {  try {  // Decode image size  BitmapFactory.Options o = new BitmapFactory.Options();  o.inJustDecodeBounds = true;  FileInputStream stream1 = new FileInputStream(f);  BitmapFactory.decodeStream(stream1, null, o);  stream1.close();  // Find the correct scale value. It should be the power of 2.  // Set width/height of recreated image  final int REQUIRED\_SIZE = 85;  int width\_tmp = o.outWidth, height\_tmp = o.outHeight;  int scale = 1;  while (true) {  if (width\_tmp / 2 < REQUIRED\_SIZE  || height\_tmp / 2 < REQUIRED\_SIZE)  break;  width\_tmp /= 2;  height\_tmp /= 2;  scale \*= 2;  }  // decode with current scale values  BitmapFactory.Options o2 = new BitmapFactory.Options();  o2.inSampleSize = scale;  FileInputStream stream2 = new FileInputStream(f);  Bitmap bitmap = BitmapFactory.decodeStream(stream2, null, o2);  stream2.close();  return bitmap;  } catch (FileNotFoundException e) {  } catch (IOException e) {  e.printStackTrace();  }  return null;  }  boolean imageViewReused(PhotoToLoad photoToLoad) {  String tag = imageViews.get(photoToLoad.imageView);  // Check url is already exist in imageViews MAP  if (tag == null || !tag.equals(photoToLoad.url))  return true;  return false;  }  // Used to display bitmap in the UI thread  class BitmapDisplayer implements Runnable {  Bitmap bitmap;  PhotoToLoad photoToLoad;  public BitmapDisplayer(Bitmap b, PhotoToLoad p) {  bitmap = b;  photoToLoad = p;  }  public void run() {  if (imageViewReused(photoToLoad))  return;  // Show bitmap on UI  if (bitmap != null) {  photoToLoad.imageView.setImageBitmap(ScaleBitmap  .getScaledBitmap(C, bitmap, photoToLoad.b));  } else {  }  // photoToLoad.imageView.setImageResource(stub\_id);  }  }  public void clearCache() {  // Clear cache directory downloaded images and stored data in maps  memoryCache.clear();  fileCache.clear();  }  }  package com.fudiyoxpress.images;  import java.util.Collections;  import java.util.Iterator;  import java.util.LinkedHashMap;  import java.util.Map;  import java.util.Map.Entry;  import android.graphics.Bitmap;  import android.util.Log;  public class MemoryCache {  private static final String TAG = "MemoryCache";  //Last argument true for LRU ordering  private Map<String, Bitmap> cache = Collections.synchronizedMap(  new LinkedHashMap<String, Bitmap>(10,1.5f,true));  //current allocated size  private long size=0;  //max memory cache folder used to download images in bytes  private long limit = 1000000;  public MemoryCache(){  //use 25% of available heap size  setLimit(Runtime.getRuntime().maxMemory()/4);  }  public void setLimit(long new\_limit){  limit=new\_limit;  Log.i(TAG, "MemoryCache will use up to "+limit/1024./1024.+"MB");  }  public Bitmap get(String id){  try{  if(!cache.containsKey(id))  return null;  //NullPointerException sometimes happen here http://code.google.com/p/osmdroid/issues/detail?id=78  return cache.get(id);  }catch(NullPointerException ex){  ex.printStackTrace();  return null;  }  }  public void put(String id, Bitmap bitmap){  try{  if(cache.containsKey(id))  size-=getSizeInBytes(cache.get(id));  cache.put(id, bitmap);  size+=getSizeInBytes(bitmap);  checkSize();  }catch(Throwable th){  th.printStackTrace();  }  }  private void checkSize() {  Log.i(TAG, "cache size="+size+" length="+cache.size());  if(size>limit){  Iterator<Entry<String, Bitmap>> iter=cache.entrySet().iterator();//least recently accessed item will be the first one iterated  while(iter.hasNext()){  Entry<String, Bitmap> entry=iter.next();  size-=getSizeInBytes(entry.getValue());  iter.remove();  if(size<=limit)  break;  }  Log.i(TAG, "Clean cache. New size "+cache.size());  }  }  public void clear() {  try{  //NullPointerException sometimes happen here http://code.google.com/p/osmdroid/issues/detail?id=78  cache.clear();  size=0;  }catch(NullPointerException ex){  ex.printStackTrace();  }  }  long getSizeInBytes(Bitmap bitmap) {  if(bitmap==null)  return 0;  return bitmap.getRowBytes() \* bitmap.getHeight();  }  }  package com.fudiyoxpress.images;  import java.io.InputStream;  import java.io.OutputStream;  public class Utils {  public static void CopyStream(InputStream is, OutputStream os)  {  final int buffer\_size=1024;  try  {  byte[] bytes=new byte[buffer\_size];  for(;;)  {  //Read byte from input stream  int count=is.read(bytes, 0, buffer\_size);  if(count==-1)  break;  //Write byte from output stream  os.write(bytes, 0, count);  }  }  catch(Exception ex){}  }  }   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/34061650)[improve this answer](http://stackoverflow.com/posts/34061650/edit) | [edited Dec 3 '15 at 9:19](http://stackoverflow.com/posts/34061650/revisions)  [[https://i.stack.imgur.com/gnoUL.jpg?s=32&g=1](http://stackoverflow.com/users/2725435/bart%c5%82omiej-sema%c5%84czyk)](http://stackoverflow.com/users/2725435/bart%c5%82omiej-sema%c5%84czyk)  [Bartłomiej Semańczyk](http://stackoverflow.com/users/2725435/bart%c5%82omiej-sema%c5%84czyk)  **13.9k**1584123 | answered Dec 3 '15 at 9:01  [[https://www.gravatar.com/avatar/0becbfe9438984d98821cd1aca2d3cbb?s=32&d=identicon&r=PG](http://stackoverflow.com/users/1988348/jotiram-chavan)](http://stackoverflow.com/users/1988348/jotiram-chavan)  [Jotiram Chavan](http://stackoverflow.com/users/1988348/jotiram-chavan)  **170**8 | |
|  | add a comment |
| up vote0down vote | Except load the data cache asynchronously, you may require the UI cache  Except the the loading visible item data, you may require to load the approximity-visible item data  Example: Suppose the listview visible item is [6,7,8,9,10], you may require to load [6,7,8,9,10] AND pre-load the item [1, 2, 3, 4, 5] & [11, 12, 13, 14, 15], because user probably scroll to the pre-page or post-page   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/37816686)[improve this answer](http://stackoverflow.com/posts/37816686/edit) | answered Jun 14 at 15:49  [[https://www.gravatar.com/avatar/b8e2b29547a9a45b2feca76fbec9e833?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/6456129/yessy)](http://stackoverflow.com/users/6456129/yessy)  [Yessy](http://stackoverflow.com/users/6456129/yessy)  **81**14 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 1 |  | | In addition to your description, please include some code to further improve your answer. – [buczek](http://stackoverflow.com/users/2291915/buczek) [Jun 14 at 16:09](http://stackoverflow.com/questions/541966/lazy-load-of-images-in-listview?page=2&tab=votes#comment63098499_37816686) |   add a comment |

|  |  |
| --- | --- |
| up vote0down vote | I found the [Glide](https://github.com/bumptech/glide) as better option than Picasso. I was using picasso to load around 32 images of size around 200-500KB each and I was always getting OOM. But the Glide solved my all OOMissues. |