**Loading Image Asynchronously in Android ListView**

* [1. Introduction](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#1-introduction)
* [2. What is Android AsyncTask](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#2-what-is-android-asynctask)
* [3. Downloading image using AsyncTask](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#3-downloading-image-using-asynctask)
* [4. Downloading image from web](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#4-downloading-image-from-web)
* [5. Creating custom ListView](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#5creating-custom-listview)
  + [5.1. Adding ListView to activity layout](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#51adding-listview-to-activity-layout)
  + [5.2. Create list view activity](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#52-createlist-view-activity)
  + [5.3. Create list row layout](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#53-create-list-row-layout)
  + [5.4. Creating custom list adapter](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#54-creating-custom-list-adapter)
  + [5.5. Using list view adapter](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#55-using-list-view-adapter)
* [5. Download Complete Example](http://stacktips.com/tutorials/android/loading-image-asynchronously-in-android-listview#5-download-complete-example)

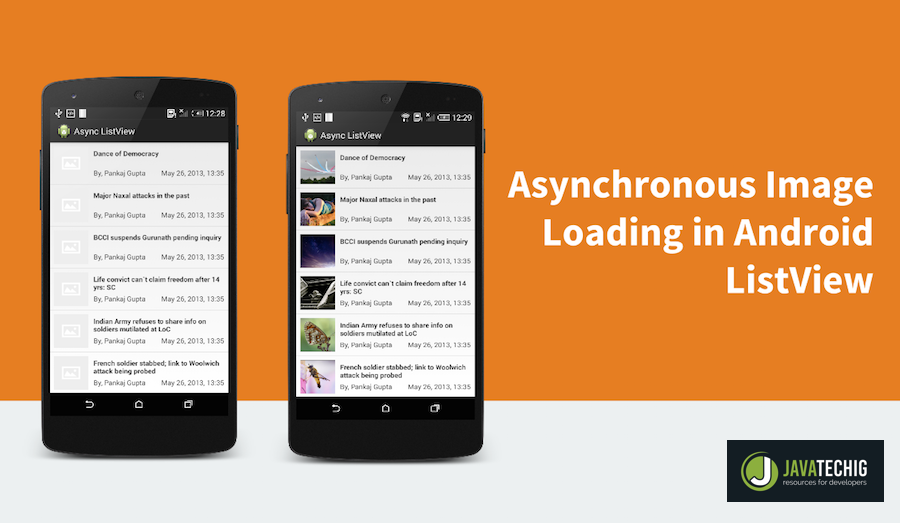
**1. Introduction**

As mobile devices are limited with memory, we must follow certain best practices to provide best performance and smooth user experience. Among set of best practices, the one holds priority is to take the long running heavy operations off the main thread.

Any long running tasks or heavy operations are usually performed in a different thread, to make sure your main thread does the minimum amount of work. Example of a typical long running tasks could be network operations, reading files form memory, animations, etc.

In this tutorial, we will create a simple [ListView](http://stacktips.com/android/android-listview-tutorial" \o "Android ListView Example) in Android that downloads data  asynchronously from the internet using a AsyncTask. As you can see in the screenshot below, the ListView contains a image thumbnails on each row, we will download the images asynchronously form server.

If you’re looking for downloading data from asynchronously from server, we recommend you to read through [Android networking tutorial.](http://stacktips.com/android/android-networking-tutorial)

[](http://st1.vchensubeswogfpjoq.netdna-cdn.com/wp-content/uploads/2013/06/Async-ListView-Android-Example.png)

**2. What is Android AsyncTask**

AsyncTask enables you to implement MultiThreading without getting your hands dirty into threads. AsyncTask is easy to use, and it allows performing background operation in dedicated thread and passing the results back UI thread. If you are doing something isolated related to UI, for example downloading data for List view, go ahead and use AsyncTask. Some of the basic characteristics of AsyncTask are as follows

1. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.
2. In onPreExecute you can define code, which need to be executed before background processing starts.
3. The doInBackground() method contains the code which needs to be executed in background, here in doInBackground we can send results to multiple times to event thread by publishProgress() method, to notify background processing has been completed we can return results simply.
4. The onProgressUpdate() method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update event thread
5. The onPostExecute() method handles results returned by doInBackground method.
6. If an async task not using any types, then it can be marked as Void type.
7. An running async task can be cancelled by calling cancel() method.

The generic types used by AsyncTask are

* Params, the type of the parameters sent to the task upon execution
* Progress, the type of the progress units published during the background computation.
* Result, the type of the result of the background computation.

**3. Downloading image using AsyncTask**

We had learnt the basics of AsyncTask. Let us take a glance at how to use it practically for downloading image asynchronously from web. To achieve this, let us create a new class and name it as ImageDownloaderTask.

The following code snippet expects the url of image as an parameter and initiate download image download request. Once download is over, it displays the bitmap on the image view.

class ImageDownloaderTask extends AsyncTask<String, Void, Bitmap> {

private final WeakReference<ImageView> imageViewReference;

public ImageDownloaderTask(ImageView imageView) {

imageViewReference = new WeakReference<ImageView>(imageView);

}

@Override

protected Bitmap doInBackground(String... params) {

return downloadBitmap(params[0]);

}

@Override

protected void onPostExecute(Bitmap bitmap) {

if (isCancelled()) {

bitmap = null;

}

if (imageViewReference != null) {

ImageView imageView = imageViewReference.get();

if (imageView != null) {

if (bitmap != null) {

imageView.setImageBitmap(bitmap);

} else {

Drawable placeholder = imageView.getContext().getResources().getDrawable(R.drawable.placeholder);

imageView.setImageDrawable(placeholder);

}

}

}

}

}

**4. Downloading image from web**

Notice that, in the above step we are calling downloadBitmap() method but haven’t declared it yet. Let us create declare the  downloadBitmap method which takes care of loading image and returning the bitmap. Here we are using HttpURLConnection to download the stream from given url. Learn more about HttpURLConnection from our[android networking tutorial](http://stacktips.com/android/android-networking-tutorial).

private Bitmap downloadBitmap(String url) {

HttpURLConnection urlConnection = null;

try {

URL uri = new URL(url);

urlConnection = (HttpURLConnection) uri.openConnection();

int statusCode = urlConnection.getResponseCode();

if (statusCode != HttpStatus.SC\_OK) {

return null;

}

InputStream inputStream = urlConnection.getInputStream();

if (inputStream != null) {

Bitmap bitmap = BitmapFactory.decodeStream(inputStream);

return bitmap;

}

} catch (Exception e) {

urlConnection.disconnect();

Log.w("ImageDownloader", "Error downloading image from " + url);

} finally {

if (urlConnection != null) {

urlConnection.disconnect();

}

}

return null;

}

**5. Creating custom ListView**

Now that we understand the basics of AsyncTask, let us proceed with creating the custom list view in android. The focus of this tutorial is tried to image download. If you not familiar with creating custom list view in android, you can read our [Android ListView tutorial](http://stacktips.com/android/android-listview-tutorial).

**5.1. Adding ListView to activity layout**

For sake of simplicity our activity layout contains a simple ListView that covers the total available width and height of the device. Create a new file activity\_main.xml in layout directory.

<?xml version="1.0" encoding="utf-8"?>

<ListView xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/custom\_list"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

</ListView>

**5.2. Create list view activity**

Let us now create a new activity class MainActivity.java in your project src directory and paste the following code. We will complete this activity in Section 3.5

public class MainActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

**5.3. Create list row layout**

Now let us focus on the layout for list view row item. As you can notice form the above screenshot, we will use [RelativeLayout](http://stacktips.com/android/android-relativelayout-example" \o "Android RelativeLayout Example) for building a simple list row view. Crete a new file list\_row\_layout.xml file in layout directory and paste the following code blocks.

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:minHeight="50dp"

android:padding="8dp">

<ImageView

android:id="@+id/thumbImage"

android:layout\_width="70dp"

android:layout\_height="70dp"

android:layout\_alignParentLeft="true"

android:layout\_centerVertical="true"

android:layout\_marginRight="10dp"

android:background="@drawable/placeholder" />

<TextView

android:id="@+id/title"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_toRightOf="@id/thumbImage"

android:minLines="2"

android:paddingTop="5dp"

android:textStyle="bold" />

<TextView

android:id="@+id/reporter"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/title"

android:layout\_marginTop="5dp"

android:layout\_toRightOf="@id/thumbImage" />

<TextView

android:id="@+id/date"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignBottom="@+id/reporter"

android:layout\_alignParentRight="true" />

</RelativeLayout>

**5.4. Creating custom list adapter**

Adapter is acts as a bridge between data source and adapter views such as ListView, GridView. Adapter iterates through the data set from beginning till the end and generate Views for each item in the list.

Create a new class named CustomListAdapter and extend it from BaseAdapter. [Visit here](http://stacktips.com/android/android-listview-tutorial)to learn more about android adapters.

public class CustomListAdapter extends BaseAdapter {

private ArrayList listData;

private LayoutInflater layoutInflater;

public CustomListAdapter(Context context, ArrayList listData) {

this.listData = listData;

layoutInflater = LayoutInflater.from(context);

}

@Override

public int getCount() {

return listData.size();

}

@Override

public Object getItem(int position) {

return listData.get(position);

}

@Override

public long getItemId(int position) {

return position;

}

public View getView(int position, View convertView, ViewGroup parent) {

ViewHolder holder;

if (convertView == null) {

convertView = layoutInflater.inflate(R.layout.list\_row\_layout, null);

holder = new ViewHolder();

holder.headlineView = (TextView) convertView.findViewById(R.id.title);

holder.reporterNameView = (TextView) convertView.findViewById(R.id.reporter);

holder.reportedDateView = (TextView) convertView.findViewById(R.id.date);

holder.imageView = (ImageView) convertView.findViewById(R.id.thumbImage);

convertView.setTag(holder);

} else {

holder = (ViewHolder) convertView.getTag();

}

NewsItem newsItem = (NewsItem) listData.get(position);

holder.headlineView.setText(newsItem.getHeadline());

holder.reporterNameView.setText("By, " + newsItem.getReporterName());

holder.reportedDateView.setText(newsItem.getDate());

if (holder.imageView != null) {

new ImageDownloaderTask(holder.imageView).execute(newsItem.getUrl());

}

return convertView;

}

static class ViewHolder {

TextView headlineView;

TextView reporterNameView;

TextView reportedDateView;

ImageView imageView;

}

}

**5.5. Using list view adapter**

Now that we have the list view and adapter class ready. Let us proceed to complete the MainActivity class. The following code snippet is used to initialize the list view and assign the custom adapter to it.

public class MainActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ArrayList<ListItem> listData = getListData();

final ListView listView = (ListView) findViewById(R.id.custom\_list);

listView.setAdapter(new CustomListAdapter(this, listData));

listView.setOnItemClickListener(new OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> a, View v, int position, long id) {

ListItem newsData = (ListItem) listView.getItemAtPosition(position);

Toast.makeText(MainActivity.this, "Selected :" + " " + newsData, Toast.LENGTH\_LONG).show();

}

});

}

private ArrayList<ListItem> getListData() {

ArrayList<ListItem> listMockData = new ArrayList<ListItem>();

String[] images = getResources().getStringArray(R.array.images\_array);

String[] headlines = getResources().getStringArray(R.array.headline\_array);

for (int i = 0; i < images.length; i++) {

ListItem newsData = new ListItem();

newsData.setUrl(images[i]);

newsData.setHeadline(headlines[i]);

newsData.setReporterName("Pankaj Gupta");

newsData.setDate("May 26, 2013, 13:35");

listMockData.add(newsData);

}

return listMockData;

}

}

Notice that, getListData() method in the activity is used to create some dummy list data for the list view. To make this example simple, we are using the string arrays defined in the strings.xml resource file. But in realtime you might download the data from server or get it from any other sources.

Add the following string array declarations to string.xml file.

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">Async ListView</string>

<array name="images\_array">

<item>http://lh5.ggpht.com/\_hepKlJWopDg/TB-\_WXikaYI/AAAAAAAAElI/715k4NvBM4w/s144-c/IMG\_0075.JPG</item>

<item>http://lh4.ggpht.com/\_4f1e\_yo-zMQ/TCe5h9yN-TI/AAAAAAAAXqs/8X2fIjtKjmw/s144-c/IMG\_1786.JPG</item>

<item>http://lh3.ggpht.com/\_GEnSvSHk4iE/TDSfmyCfn0I/AAAAAAAAF8Y/cqmhEoxbwys/s144-c/\_MG\_3675.jpg</item>

<item>http://lh6.ggpht.com/\_ZN5zQnkI67I/TCFFZaJHDnI/AAAAAAAABVk/YoUbDQHJRdo/s144-c/P9250508.JPG</item>

<item>http://lh4.ggpht.com/\_XjNwVI0kmW8/TCOwNtzGheI/AAAAAAAAC84/SxFJhG7Scgo/s144-c/0014.jpg</item>

<item>http://lh6.ggpht.com/\_Nsxc889y6hY/TBp7jfx-cgI/AAAAAAAAHAg/Rr7jX44r2Gc/s144-c/IMGP9775a.jpg</item>

<item>http://lh6.ggpht.com/\_ZN5zQnkI67I/TCFFZaJHDnI/AAAAAAAABVk/YoUbDQHJRdo/s144-c/P9250508.JPG</item>

</array>

<array name="headline\_array">

<item>Dance of Democracy</item>

<item>Major Naxal attacks in the past</item>

<item>BCCI suspends Gurunath pending inquiry </item>

<item>Life convict can`t claim freedom after 14 yrs: SC</item>

<item>Indian Army refuses to share info on soldiers mutilated at LoC</item>

<item>French soldier stabbed; link to Woolwich attack being probed</item>

<item>Life convict can`t claim freedom after 14 yrs: SC</item>

</array>

</resources>

**5. Download Complete Example**

[**Download**](https://github.com/javatechig/Async-ListView-Image-Loader)