Inducesmile

Mobile developer community for beginners and students

ANDROID REMOTE IMAGE DOWNLOAD USING ANDROID DOWNLOADMANAGER, ANDROID ASYNCTASK, VOLLEY, UNIVERSAL IMAGE LOADER, PICASSO AND OKHTTP

In this tutorial, we are going to learn different ways we can download images from a remote source in our android app. We will focus on android remote image download using Android DownloadManager,

ANDROID SOURCE CODE POLL

Which of the below listed source codes should we publish next month?

Multi Restaurant Food Ordering App (27%, 39 Votes)

Stock and Inventory App (18%, 26 Votes)

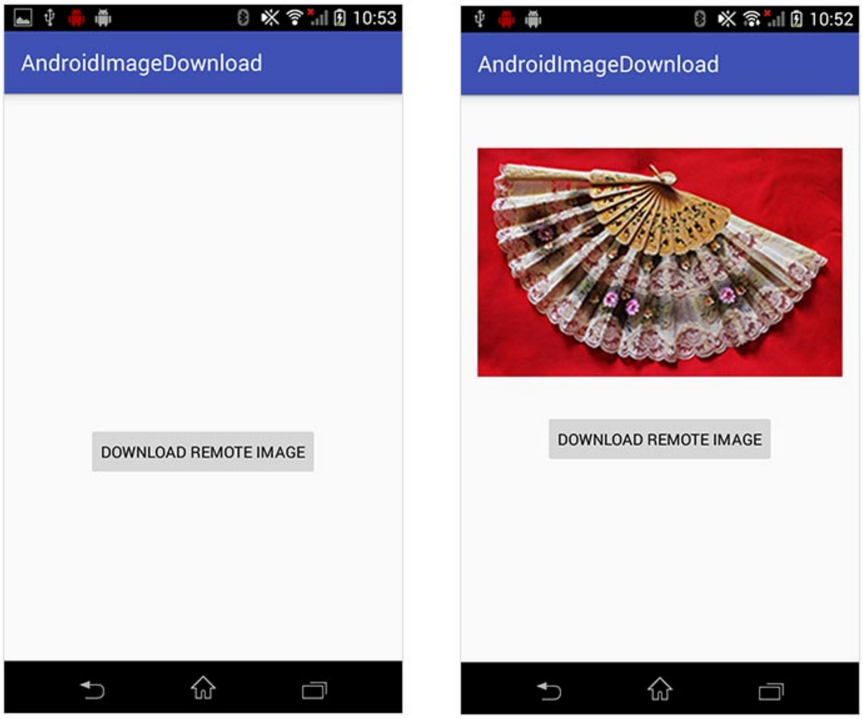
Android Dating App (16%, 23

image download and network calls but a way to demonstrate how we can use them to achieve the same result.

Beside the libraries and Android Service mentioned here, there are other libraries out there but these are the libraries you will always hear about in android image download.

We are going to create an Activity page that contains button views that is link to each of our test activity class.

Right now it is important for us to understand what we are planning to achieve in this tutorial. The screen-shot of our project is shown below.



Before we start, it is important for you to understand the tools and environment I used in this android tutorial. Feel free to use tools you are familiar with.

Windows 7

(10%, 15 votes)
□

Business & Service Provider Finder (6%, 9 Votes)
□

Total Voters: **147**

LIKE US ON FACEBOOK

Induce smile
5,305 likes

Like Page

Be the first of your friends to like this

RECENT POSTS

- Camera Translator All Language App
- Why MVVM and How to execute MVVM combined with Data Binding and LiveData in Four Simple Steps
- Camera Translator All Languages – Android Project idea source code for students
- Note Scan Android App -Project idea source code for students
- How To Build Android WhatsApp Clone Chat Application
- Android Car Rental App with Web Admin Panel – Project Idea Source Code For Students

To create a new android application project, following the steps as stipulated below.

Go to File menu

Click on New menu

Click on Android Application

Enter Project name: AndroidImageDownload

Package: com.inducesmile.androidimagedownload

Select Empty Activity

Keep other default selections

Continue to click on next button until Finish button is active, then click on Finish Button.

If you are using Android Studio as your choice IDE, the new project will create a default

activity file and its corresponding layout file.

I have named my activity DownloadActivity.java. Before we go ahead and start adding codes inside these file, it is important for also to add all the dependency libraries we need.

Open your app build.gradle file located at **Gradle Script > build.gradle.** Copy and paste the following lines of code.

```
compile 'com.mcxiaoke.volley:library:1.0.19'
compile 'com.squareup.okhttp3:okhttp:3.1.2'
compile 'com.google.code.gson:gson:2.6.1'
compile 'com.nostra13.universalimageloader:universal-ima
```

- Android UI App for Food Ordering and Delivery from Multiple Restaurants

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorBlack">#000000</color>
</resources>
```

The strings.xml also will contain these lines of code. Feel free to change any of these values to fit your own project.

```
<resources>
    <string name="app_name">AndroidImageDownload</string
    <string name="download_image">Download remote image<
    <string name="download_options">Another Download Opt
    <string name="download_manager">Download image using
    <string name="async">Download image using AsyncTask<
    <string name="volley">Download image using Volley</s
    <string name="universal">Download image using Univer
    <string name="picasso">Download image using Picasso<
    <string name="okhttp">Download image using OkHttp</s
    <string name="retrofit">Download image using retrofi
</resources>
```

We will move ahead to create our first layout file for the launch activity – activity_download.xml. Open this file, copy and paste the below code in this layout file.

ACTIVITY_DOWNLOAD.XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
        android:scrollbars="none">
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <Button
                android:id="@+id/downloadmanager_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/downloaded_im
                android:layout_centerHorizontal="true"
                android:layout_marginTop="8dp"
                android:padding="16dp"
                android:text="@string/download_manager"
                android:textColor="@color/colorBlack" />
            <Button
                android:id="@+id/async_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/downloadmanag
                android:layout_centerHorizontal="true"
                android:layout_marginTop="16dp"
                android:padding="16dp"
                android:text="@string/async"
                android:textColor="@color/colorBlack" />
            <Button
                android:id="@+id/volley_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/async_button"
                android:layout_centerHorizontal="true"
                android:layout_marginTop="16dp"
                android:padding="16dp"
                android:text="@string/volley"
                android:textColor="@color/colorBlack" />
            <Button
                android:id="@+id/universal_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/universal_but
                android:layout_centerHorizontal="true"
                android:layout_marginTop="16dp"
                android:padding="16dp"
                android:text="@string/picasso"
                android:textColor="@color/colorBlack" />
            <Button
                android:id="@+id/okhttp_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@+id/picasso_butto
                android:layout_centerHorizontal="true"
                android:layout_marginTop="16dp"
                android:padding="16dp"
                android:text="@string/okhttp"
                android:textColor="@color/colorBlack" />
        </RelativeLayout>
    </ScrollView>
</RelativeLayout>
```

The layout file above is very simple. It includes six Button View controls link to different activity class for each image download.

Since the layout file is ready, we will proceed to the DownloadActivity.java file. We will get all the button instances and wire click events to them. The Intent class is use to move from the DownloadActivity page to another page. Open the DownloadActivity.java file. Copy and paste the following code inside this file.

DOWNLOADACTIVITY.JAVA

```
import android.content.Intent;
```

```
        Button downloadManagerButton = (Button)findViewB
        Button asyncButton = (Button)findViewById(R.id.a
        Button volleyButton = (Button)findViewById(R.id.
        Button universalButton = (Button)findViewById(R.
        Button picassoButton = (Button)findViewById(R.id
        Button okhttpButton = (Button)findViewById(R.id.
        Button retrofitButton = (Button)findViewById(R.i
        downloadManagerButton.setOnClickListener(this);
        asyncButton.setOnClickListener(this);
        volleyButton.setOnClickListener(this);
        universalButton.setOnClickListener(this);
        picassoButton.setOnClickListener(this);
        okhttpButton.setOnClickListener(this);
        retrofitButton.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        int id = v.getId();
        switch (id){
            case R.id.downloadmanager_button:
                Intent managerIntent = new Intent(this,
                startActivity(managerIntent);
                break;
            case R.id.async_button:
                Intent asyncIntent = new Intent(this, Do
                startActivity(asyncIntent);
                break;
            case R.id.volley_button:
                Intent volleyIntent = new Intent(this, V
                startActivity(volleyIntent);
                break;
            case R.id.universal_button:
                Intent universalIntent = new Intent(this
                startActivity(universalIntent);
                break;
            case R.id.picasso_button:
                Intent picassoIntent = new Intent(this,
                startActivity(picassoIntent);
```

USING ANDROID DOWNLOADMANAGER FOR REMOTE IMAGE DOWNLOAD

When you click on the first button, it will take you to a new activity page where you will implement the use of Android DownloadManager to download image from a remote source in your android application.

According to android user guide – The download manager is a system service that handles long-running HTTP downloads. Clients may request that a URI be downloaded to a particular destination file.

The download manager will conduct the download in the background, taking care of HTTP interactions and retrying downloads after failures or across connectivity changes and system reboots.

Instances of this class should be obtained through getSystemService(String) by passing DOWNLOAD_SERVICE. Apps that request downloads through this API should register a broadcast receiver for ACTION_NOTIFICATION_CLICKED to appropriately handle when the user clicks on a running download in a notification or from the downloads UI.

Note that the application must have the INTERNET permission to use this class.

Android Download Manager might not be the best option when you what to download an image and display it in an ImageView.

Right click in your project java > package folder, select activity in the fly-out menu and click to create and empty activity. Name this Activity file – **DownloadManagerActivity.java.**

Open the activity layout file location at **res > layout.** We are going to

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_marg
    android:paddingLeft="@dimen/activity_horizontal_marg
    android:paddingRight="@dimen/activity_horizontal_mar
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.inducesmile.androidimagedownload.
    <ImageView
        android:id="@+id/downloaded_image"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:adjustViewBounds="true"
        android:scaleType="fitXY"
        android:src="@drawable/trans" />
    <Button
        android:id="@+id/download_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/downloaded_image"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="@string/download_image"
        android:textColor="@color/colorBlack" />
</RelativeLayout>
```

In the DownloadManagerActivity.java file, we will get the instances of our View controls and wire the button event click.

Request an instance of the DownloadManager through android system service call.

```
request = new DownloadManager.Request(uri);
request.setTitle("Downloading");
request.setDescription("Download in process");
request.setDestinationInExternalFilesDir(MainActivity.th
request.setVisibleInDownloadsUi(true);
```

We will register a BroadcastReceiver that will get notify when the image download is completed by using an action DownloadManager.ACTION_DOWNLOAD_COMPLETE.

Finally, we will enqeue the request object. The complete code for this class is shown below.

```
package com.inducesmile.androidimagedownload;
import android.app.DownloadManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
    private DownloadManager downloadManager;
    private Uri uri;
    private DownloadManager.Request request;
    private long downloadReferenceId;
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    private BroadcastReceiver receiver;
```

```
request = new DownloadManager.Request(uri);
request.setTitle("Downloading");
request.setDescription("Download in process");
request.setDestinationInExternalFilesDir(MainAct
request.setVisibleInDownloadsUi(true);
downloadButton.setOnClickListener(new View.OnCli
    @Override
    public void onClick(View v) {
        downloadReferenceId = downloadManager.en
    }
});
receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Inten
        String mAction = intent.getAction();
        if(DownloadManager.ACTION_DOWNLOAD_COMPL
            long returnedId = intent.getLongExtr
            if(returnedId == downloadReferenceId
                DownloadManager.Query mQuery = n
                mQuery.setFilterById(returnedId)
                Cursor cursor = downloadManager.
                if(cursor.moveToFirst()){
                    int statusIndex = cursor.get
                    if (DownloadManager.STATUS_S
                        Toast.makeText(MainActiv
                        return;
                    }
                    int uriIndex = cursor.getCol
                    String downloadedPackageUriS
                    Uri mUri = Uri.parse(downloa
                    imagePlaceHolder.setImageURI
                }
            }
        }
    }
};
IntentFilter mIntentFilter = new IntentFilter(Do
registerReceiver(receiver, mIntentFilter);
```

Create a new java file in your package folder and name it Helper.java. Open the file and copy and paste this single line of code to it. It holds the path to the image file will want to download.

```
public class Helper {
    public static final String imageDownloadPath = "path
}
```

ANDROID ASYNCTASK FOR REMOTE IMAGE DOWNLOAD

We look at how to achieve the same feature using Android AsyncTask class. Create a new empty activity file. Name the file DownloadAsyncActivity.java.

Open the layout file and include the same layout we used in activity_download_manager.xml.

Please take note, we are going to use the same layout content in all our layout file. That is – one ImageView and Button.

Our class will inherit from Android AsyncTask class. We will going to override two methods from the parent class – **Bitmap doInBackground(String… urls) and onPostExecute(Bitmap result).**

The Bitmap doInBackground(String… urls) method runs a background thread which is use for the image download. This is important since android does not allow process intensive tasks to be run in the UI Thread.

The onPostExecute(Bitmap result) return the result of the download as an image bitmap. The bitmap will be set to our ImageView.

```
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import java.io.InputStream;
public class DownloadAsync extends AppCompatActivity {
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_download_async)
        imagePlaceHolder = (ImageView)findViewById(R.id.
        downloadButton = (Button)findViewById(R.id.downl
        downloadButton.setOnClickListener(new View.OnCli
            @Override
            public void onClick(View v) {
                DownloadImageWithURLTask downloadTask =
                downloadTask.execute(Helper.imageDownloa
            }
        });
    }
    private class DownloadImageWithURLTask extends Async
        ImageView bmImage;
        public DownloadImageWithURLTask(ImageView bmImag
            this.bmImage = bmImage;
        }
        protected Bitmap doInBackground(String... urls)
            String pathToFile = urls[0];
            Bitmap bitmap = null;
            try {
                InputStream in = new java.net.URL(pathTo
                bitmap = BitmapFactory.decodeStream(in);
            } catch (Exception e) {
                Log.e("Error", e.getMessage());
                e.printStackTrace();
            }
            return bitmap;
        }
```

ANDROID VOLLEY FOR REMOTE IMAGE DOWNLOAD

Volley is an HTTP library that makes networking for Android apps easier and most importantly, faster. Volley is available through the open AOSP repository.

Volley offers the following benefits:

Automatic scheduling of network requests.

Multiple concurrent network connections.

Transparent disk and memory response caching with standard HTTP cache coherence.

Support for request prioritization.

Cancellation request API. You can cancel a single request, or you can set blocks or scopes of requests to cancel.

Ease of customization, for example, for retry and backoff.

Strong ordering that makes it easy to correctly populate your UI with data fetched asynchronously from the network.

Debugging and tracing tools.

We are focus on a simple remote image download. If you have not used Volley before in your android application for network task, consider using it.

Now, let create a new empty activity file called VolleyDownloadActivity.java. Open the layout file and paste the layout code we used in the proceeding layout for AsyncTask download.

```
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.ImageRequest;

public class VolleyDownloadActivity extends AppCompatAct
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    private ImageRequest request;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_volley_download
        imagePlaceHolder = (ImageView)findViewById(R.id.
        downloadButton = (Button)findViewById(R.id.downl
        downloadButton.setOnClickListener(new View.OnCli
            @Override
            public void onClick(View v) {
                VolleySingleton.getInstance(getApplicati
            }
        });
        request = new ImageRequest(Helper.imageDownloadP
            @Override
            public void onResponse(Bitmap bitmap) {
                imagePlaceHolder.setImageBitmap(bitmap);
            }
        }, 0, 0, null,
                new Response.ErrorListener() {
                    public void onErrorResponse(VolleyEr
                        Toast.makeText(VolleyDownloadAct
                    }
                });
    }
}
```

Copy and paste the following code inside this Activity class.

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import com.nostra13.universalimageloader.core.DisplayIma
import com.nostra13.universalimageloader.core.ImageLoade
import com.nostra13.universalimageloader.core.ImageLoade
import com.nostra13.universalimageloader.core.download.B
public class UniversalDownloadActivity extends AppCompat
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    private ImageLoader imageLoader;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_universal_downl
        imagePlaceHolder = (ImageView)findViewById(R.id.
        downloadButton = (Button)findViewById(R.id.downl
        ImageLoaderConfiguration config = new ImageLoade
                .imageDownloader(new BaseImageDownloader
                .defaultDisplayImageOptions(DisplayImage
                .build();
        ImageLoader.getInstance().init(config);
        imageLoader = ImageLoader.getInstance();
        downloadButton.setOnClickListener(new View.OnCli
            @Override
            public void onClick(View v) {
                imageLoader.displayImage(Helper.imageDow
            }
        });
    }
}
```

Create a new empty activity file and name it PicassoDownloadActivity.java. Set up the layout file by copying the previous layout code to it.

Open the Activity class, copy and paste the following code inside. You can see how Picasso is achieving the same image download with a single line code.

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import com.squareup.picasso.Picasso;
public class PicassoDownloadActivity extends AppCompatAc
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_picasso_downloa
        imagePlaceHolder = (ImageView)findViewById(R.id.
        downloadButton = (Button)findViewById(R.id.downl
        downloadButton.setOnClickListener(new View.OnCli
            @Override
            public void onClick(View v) {
                Picasso.with(PicassoDownloadActivity.thi
            }
        });
    }
}
```

Using OkHttp for Remote Image Download

Finally, we will create another empty activity file and name it

```
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import java.io.IOException;
import java.io.InputStream;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
public class OkhttpDownloadActivity extends AppCompatAct
    private ImageView imagePlaceHolder;
    private Button downloadButton;
    private final OkHttpClient client = new OkHttpClient
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_okhttp_download
        imagePlaceHolder = (ImageView)findViewById(R.id.
        downloadButton = (Button)findViewById(R.id.downl
        downloadButton.setOnClickListener(new View.OnCli
            @Override
            public void onClick(View v) {
                try {
                    run();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
    public void run() throws Exception {
        Request request = new Request.Builder()
                .url(Helper.imageDownloadPath)
                .build();
       client.newCall(request).enqueue(new Callback() {
           @Override
```

```
                       imagePlaceHolder.setImageBitmap(b
                   }
               });
           }
       });
    }
}
```

You have seen how you can use any of these libraries and default download manager in android.

This brings us to the end of this tutorial, If you find anything confusing kindly contact me with your questions or use the comment box below.

Now, when you run your application you will see the interface that looks similar to the sample that was shown earlier on.

You can download the code for this tutorial below. If you are having hard time downloading the tutorials, kindly contact me.

Image Download USING ANDROID DOWNLOADMANAGER, ANDROID ASYNCTASK, VOLLEY, UNIVERSAL IMAGE LOADER, PICASSO AND OKHTTP 8.25 MB
Download

Remember to subscribe with your email address so that you will be among the first to receive my new post once it is published.

Please if you love this tutorial, kindly download my new android app – ***Daily Pocket Calculator*** – in Google Play Store and let me know what you think about it.

OTHER INTERESTING POSTS:

in Android Studio

TabWidget or with FragmentTabHost and Fragment

Android how to add Search Widget and SearchView Implementation in Android UI

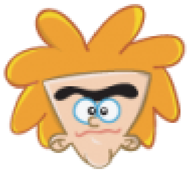
Populate Android Spinner Data from Remote MYSQL Database

Android CameraView Library Demo Example

Android Room with LiveData Example Tutorial

Tags: Android AsyncTask, Android DownloadManager, Picasso and OkHttp, Universal Image Loader, Volley

ABOUT THE AUTHOR



ADD A COMMENT

You must be logged in to post a comment.