



Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour ×

Download a file with Android, and showing the progress in a ProgressDialog

I am trying to write a simple application that gets updated. For this I need a simple function that can download a file and show the current progress in a ProgressDialog. I know how to do the ProgressDialog, but I'm not sure how to display the current progress and how to download the file in the first place.

java android download android-asynctask

edited Jun 12 '10 at 13:05



Cristian
93.4k ●29 ●231 ●203

asked Jun 12 '10 at 10:22



Tom Leese
4,654 ●6 ●25 ●45

1 Simple function would be : dixitpatel.com/?p=128 – star18bit Aug 8 '13 at 23:27

I hope below link may be help you... [androidhive.info/2012/04/...](http://androidhive.info/2012/04/) – Ganesh Katikar Jan 23 at 11:14

add a comment

7 Answers

There are many ways to download files. Following I will post most common ways; it is up to you to decide which method is better for your app.

1. Use `AsyncTask` and show the download progress in a dialog

This method will allow you to execute some background processes and update the UI at the same time (in this case, we'll update a progress bar).

This is an example code:

```
// declare the dialog as a member field of your activity
ProgressDialog mProgressDialog;

// instantiate it within the onCreate method
mProgressDialog = new ProgressDialog(YourActivity.this);
mProgressDialog.setMessage("A message");
mProgressDialog.setIndeterminate(true);
mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
mProgressDialog.setCancelable(true);

// execute this when the downloader must be fired
final DownloadTask downloadTask = new DownloadTask(YourActivity.this);
downloadTask.execute("the url to the file you want to download");

mProgressDialog.setOnCancelListener(new DialogInterface.OnCancelListener() {
    @Override
    public void onCancel(DialogInterface dialog) {
        downloadTask.cancel(true);
    }
});
```

The `AsyncTask` will look like this:

```

connection = (URLConnection) url.openConnection();
connection.connect();

// expect HTTP 200 OK, so we don't mistakenly save error report
// instead of the file
if (connection.getResponseCode() != HttpURLConnection.HTTP_OK) {
    return "Server returned HTTP " + connection.getResponseCode()
        + " " + connection.getResponseMessage();
}

// this will be useful to display download percentage
// might be -1: server did not report the length
int fileLength = connection.getContentLength();

// download the file
input = connection.getInputStream();
output = new FileOutputStream("/sdcard/file_name.extension");

byte data[] = new byte[4096];
long total = 0;
int count;
while ((count = input.read(data)) != -1) {
    // allow canceling with back button
    if (isCancelled()) {
        input.close();
        return null;
    }
    total += count;
    // publishing the progress...
    if (fileLength > 0) // only if total length is known
        publishProgress((int) (total * 100 / fileLength));
    output.write(data, 0, count);
}
} catch (Exception e) {
    return e.toString();
} finally {

```

The method above (`doInBackground`) runs always on a background thread. You shouldn't do any UI tasks there. On the other hand, the `onProgressUpdate` and `onPostExecute` run on the UI thread, so there you can change the progress bar:

```

@Override
protected void onPreExecute() {
    super.onPreExecute();
    // take CPU lock to prevent CPU from going off if the user
    // presses the power button during download
    PowerManager pm = (PowerManager) context.getSystemService(Context.POWER_SERVICE);
    mWakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
        getClass().getName());
    mWakeLock.acquire();
    mProgressDialog.show();
}

@Override
protected void onProgressUpdate(Integer... progress) {
    super.onProgressUpdate(progress);
    // if we get here, length is known, now set indeterminate to false
    mProgressDialog.setIndeterminate(false);
    mProgressDialog.setMax(100);
    mProgressDialog.setProgress(progress[0]);
}

@Override
protected void onPostExecute(String result) {
    mWakeLock.release();
    mProgressDialog.dismiss();
    if (result != null)
        Toast.makeText(context, "Download error: " + result, Toast.LENGTH_LONG).show();
    else
        Toast.makeText(context, "File downloaded", Toast.LENGTH_SHORT).show();
}

```

For this to run, you need the `WAKE_LOCK` permission.

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

2. Download from Service

The big question here is: *how do I update my activity from a service?*. In the next example we are going to use two classes you may not be aware of: `ResultReceiver` and `IntentService`. `ResultReceiver` is the one that will allow us to update our thread from a service; `IntentService` is a subclass of `Service` which spawns a thread to do background work from there (you should know that a `Service` runs actually in the same thread of your app; when you extends `Service`, you must manually spawn new threads to run CPU blocking operations).

Download service can look like this:

```
public class DownloadService extends IntentService {
    public static final int UPDATE_PROGRESS = 8344;
    public DownloadService() {
        super("DownloadService");
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        String urlToDownload = intent.getStringExtra("url");
        ResultReceiver receiver = (ResultReceiver) intent.getParcelableExtra("receiver");
        try {
            URL url = new URL(urlToDownload);
            URLConnection connection = url.openConnection();
            connection.connect();
            // this will be useful so that you can show a typical 0-100% progress bar
            int fileLength = connection.getContentLength();

            // download the file
            InputStream input = new BufferedInputStream(connection.getInputStream());
            OutputStream output = new FileOutputStream("/sdcard/BarcodeScanner-debug.");

            byte data[] = new byte[1024];
            long total = 0;
            int count;
            while ((count = input.read(data)) != -1) {
                total += count;
                // publishing the progress...
                Bundle resultData = new Bundle();
                resultData.putInt("progress", (int) (total * 100 / fileLength));
                receiver.send(UPDATE_PROGRESS, resultData);
                output.write(data, 0, count);
            }

            output.flush();
            output.close();
            input.close();
        }
    }
}
```

Add the service to your manifest:

```
<service android:name=".DownloadService"/>
```

And the activity will look like this:

```
// initialize the progress dialog like in the first example

// this is how you fire the downloader
mProgressDialog.show();
Intent intent = new Intent(this, DownloadService.class);
intent.putExtra("url", "url of the file to download");
intent.putExtra("receiver", new DownloadReceiver(new Handler()));
startService(intent);
```

Here is where `ResultReceiver` comes to play:

```
private class DownloadReceiver extends ResultReceiver{
    public DownloadReceiver(Handler handler) {
        super(handler);
    }

    @Override
    protected void onReceiveResult(int resultCode, Bundle resultData) {
        super.onReceiveResult(resultCode, resultData);
        if (resultCode == DownloadService.UPDATE_PROGRESS) {
            int progress = resultData.getInt("progress");
            mProgressDialog.setProgress(progress);
            if (progress == 100) {
                mProgressDialog.dismiss();
            }
        }
    }
}
```

2.1 Use Groundy library

Groundy is a library a friend and I wrote time ago. It basically helps you run pieces of code in a background service, and it is based on the `ResultReceiver` concept shown above. This is how the **whole** code would look like...

The activity where you are showing the dialog...

```

public class MainActivity extends Activity {

    private ProgressDialog mProgressDialog;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        findViewById(R.id.btn_download).setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                String url = ((EditText) findViewById(R.id.edit_url)).getText().toString();
                Bundle extras = new Bundler().add(DownloadTask.PARAM_URL, url).build();
                Groundy.create(DownloadExample.this, DownloadTask.class)
                    .receiver(mReceiver)
                    .params(extras)
                    .queue();

                mProgressDialog = new ProgressDialog(MainActivity.this);
                mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
                mProgressDialog.setCancelable(false);
                mProgressDialog.show();
            }
        });
    }

    private ResultReceiver mReceiver = new ResultReceiver(new Handler()) {
        @Override
        protected void onReceiveResult(int resultCode, Bundle resultData) {
            super.onReceiveResult(resultCode, resultData);
            switch (resultCode) {
                case Groundy.STATUS_PROGRESS:
                    mProgressDialog.setProgress(resultData.getInt(Groundy.KEY_PROGRESS));
                    break;
                case Groundy.STATUS_FINISHED:
                    mProgressDialog.dismiss();
                    break;
            }
        }
    };
}

```

A `GroundyTask` implementation used by **Groundy** to download the file and show the progress:

```

public class DownloadTask extends GroundyTask {
    public static final String PARAM_URL = "com.groundy.sample.param.url";

    @Override
    protected boolean doInBackground() {
        try {
            String url = getParameters().getString(PARAM_URL);
            File dest = new File(getContext().getFilesDir(), new File(url).getName());
            DownloadUtils.downloadFile(getContext(), url, dest, DownloadUtils.getDefaultDownloadDir());
            return true;
        } catch (Exception e) {
            return false;
        }
    }
}

```

And just add this to the manifest:

```
<service android:name="com.codeslap.groundy.GroundyService"/>
```

It couldn't be easier I think. Just grab the latest jar [from Github](#) and you are ready to go. Keep in mind that **Groundy**'s main purpose is to make calls to external REST apis in a background service and post results to the UI with easily. If you are doing something like that in your app, it could be really useful.

3. Use `DownloadManager` class (`GingerBread` and newer only)

This method is awesome, you do not have to worry about downloading the file manually, handle threads, streams, etc. `GingerBread` brought a new feature: `DownloadManager` which allows you to download files easily and delegate the hard work to the system.

First, let's see a utility method:

```

/**
 * @param context used to check the device version and DownloadManager information
 * @return true if the download manager is available
 */
public static boolean isDownloadManagerAvailable(Context context) {
    try {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.GINGERBREAD) {
            return false;
        }
        Intent intent = new Intent(Intent.ACTION_MAIN);
        intent.addCategory(Intent.CATEGORY_LAUNCHER);
        intent.setClassName("com.android.providers.downloads.ui", "com.android.provider
        List<ResolveInfo> list = context.getPackageManager().queryIntentActivities(inte
        PackageManager.MATCH_DEFAULT_ONLY);
        return list.size() > 0;
    } catch (Exception e) {
        return false;
    }
}

```

Method's name explains it all. Once you are sure `DownloadManager` is available, you can do something like this:

```

String url = "url you want to download";
DownloadManager.Request request = new DownloadManager.Request(Uri.parse(url));
request.setDescription("Some description");
request.setTitle("Some title");
// in order for this if to run, you must use the android 3.2 to compile your app
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
    request.allowScanningByMediaScanner();
    request.setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBLE_NOTIFY
}
request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS, "name-of-the

// get download service and enqueue file
DownloadManager manager = (DownloadManager) getSystemService(Context.DOWNLOAD_SERVICE);
manager.enqueue(request);

```

Download progress will be showing in the notification bar.

Final thoughts


First and second methods are just the tip of the iceberg. There are lots of things you have to keep in mind if you want your app to be robust. Here is a brief list:

- You must check whether user has an internet connection available
- Make sure you have the right permissions (`INTERNET` and `WRITE_EXTERNAL_STORAGE`); also `ACCESS_NETWORK_STATE` if you want to check internet availability.
- Make sure the directory were you are going to download files exist and has write permissions.
- If download is too big you may want to implement a way to resume the download if previous attempts failed.
- Users will be grateful if you allow them to interrupt the download.

Unless you want to have full control over the download process, I highly recommend using `DownloadManager` which already handles most of the items listed above.

edited Aug 9 at 17:40

community wiki
 18 revs, 6 users 74%
 Cristian

-
- 7 I edited my question to show so. – [Cristian](#) Feb 14 '11 at 13:33
-
- 4 DownloadManager is part of the OS, which means it will always be available in GB+ and cannot be uninstalled. – [Cristian](#) Aug 22 '12 at 16:16 
-
- 8 There is a problem in Cristian's answer. Because code at "**1. Use AsyncTask and show the download progress in a dialog**" does `connection.connect();` then `InputStream input = new BufferedInputStream(url.openStream());`; code makes 2 connections to the server. I have managed to change this behavior by updating the code as follows `InputStream input = new BufferedInputStream(connection.getInputStream());`; – [nLL](#) Nov 16 '12 at 11:42
-
- 6 i wish android documentation was this concise. – [Louis Morda](#) Nov 28 '12 at 23:08
-
- 4 Suggested to `close()` the streams (`input` and `output`) in `finally` instead of `try`, otherwise if any exception is thrown before `close()`, you have unclosed streams hanging around. – [Pang](#) Jan 26 '13 at 10:02
-

show **38** more comments

Don't forget to add permissions to your manifest file if you're gonna be downloading stuff from the internet!

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloandroid"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="10" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"></uses-permission>

    <application android:icon="@drawable/icon" android:label="@string/app_name" android:
    </application>
```

edited Apr 18 '12 at 15:30



Cristian

93.4k ● 29 ● 231 ● 203

answered Aug 29 '11 at 14:34



Mnightmare

593 ● 5 ● 4

[add a comment](#)

Yes the code above will work .But if you are updating your progressbar in `onProgressUpdate` of `AsyncTask` and you press back button or finish your activity `AsyncTask` loses its track with your UI .And when you go back to your activity, even if download is running in background you will see no update on progressbar. So on `OnResume()` try to run a thread like `runOnUiThread` with a timer task that updates ur progressbar with values updating from the `AsyncTask` running background.

```
private void updateProgressBar(){
    Runnable runnable = new updateProgress();
    background = new Thread(runnable);
    background.start();
}
public class updateProgress implements Runnable {
    public void run() {
        while(Thread.currentThread()==background)
            //while (!Thread.currentThread().isInterrupted()) {
            try {
                Thread.sleep(1000);
                Message msg = new Message();
                progress = getProgressPercentage();
                handler.sendMessage(msg);
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
            } catch (Exception e) {
            }
        }
    }
}
private Handler handler = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        progress.setProgress(msg.what);
    }
};
```

Don't forget to Destroy the thread when ur activity is not visible.

```
private void destroyRunningThreads()
{
    if(background!=null)
    {
        background.interrupt();
        background=null;
    }
}
```

edited Mar 14 '13 at 5:57



Sachin Chavan

2,285 ● 2 ● 24 ● 51

answered Jun 12 '12 at 14:21



sheetal


1,036 ● 7 ● 18

This is exactly my problem. Can you tell me how to do a timer task to update progressbar? How to get values updating from the `AsyncTask` running behind – [user1417127](#) Jul 10 '12 at 6:43

- okk take a global or static variable to where to update your values from `asynctask`...or u can insert it in `DataBase` for safe side..so that closing application wont hamper..And when u restart the activity where u want to update ur UI run the UI thread..see example below – [sheetal](#) Jul 10 '12 at 13:40

The UI should have the fresh reference..Like the newly initialized `ProgressBar` in my case – [sheetal](#) Jul 10 '12

at 13:53

@sheetal, But It works fine without your code! Why?! My device is Xperia P with Android 4.0.4. I've defined a static boolean variable that the onPreExecute sets it to true and the onPostExecute sets it to false. It shows that we are downloading or not, so we can check if the variable is equal to true, show the previous progressbar dialog. – Behzad Feb 20 '13 at 21:35 

@sheetal your code is little obscurant, can you tell me some advice? – iSun Mar 6 '13 at 15:27

show 2 more comments

I'd recommend you to use my Project [Netroid](#), It base on [Volley](#) which Google IO 2013 presentation, I extend it, add some features such as multi-events callback, file download management, I think that's what you looking for.

answered May 7 at 15:56



Does it support multiple file download? I'm working on a project that allow user to download scheduled. At a specific time (12 AM for instance), the service will download all the link that user selected before. I mean the service I need has to be able to queue download links, and then download all of them (one by one or parallel, depend on user). Thank you – piavgh Jun 26 at 7:43


@piavgh yes, all you wanted was satisfy, you can checkout my sample apk, it included a file download management demo. – VinceStyling Jun 26 at 8:49

add a comment

Do not forget to replace "/sdcard..." by **new File("/mnt/sdcard/...")** otherwise you will get a **FileNotFoundException**

answered Feb 10 at 0:17



In part 2 Download from service Progress dialog can not display progress increment. it direct return 100,so if 100 it direct check setprogress 100 and progress over, how to increment progress ??it display only 0 progress but actually download running – Nirav Mehta Apr 10 at 6:09 

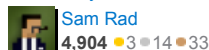
it display only 0% out of 100 only other work properly – Nirav Mehta Apr 10 at 6:15

2 Do not do it! There is `Environment.getExternalStorageDirectory().getAbsolutePath()` for getting a path to sdcard. Also don't forget to check if the external storage is mounted - `Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)` (Mannaz) – naXa Apr 15 at 2:02

add a comment

My personal advice is to use **Progress Dialog** and build up before execution , or initiate at `OnPreExecute()` , publish progress often if you use horizontal style of progress bar of the progress dialog. The remaining part is to optimize the algorithm of `doInBackground` .

edited May 30 at 9:17



add a comment

answered Jul 2 '13 at 17:11



I found [this](#) blog post very helpful, Its using loopJ to download file, it has only one Simple function, will be helpful to some new android guys.

answered Aug 1 at 14:55



add a comment

protected by [ho1](#) Dec 7 '11 at 11:13

Thank you for your interest in this question. Because it has attracted low-quality answers, posting an answer now requires 10 [reputation](#) on this site.

Would you like to answer one of these [unanswered questions](#) instead?

Not the answer you're looking for? Browse other questions tagged [java](#) [android](#)

[download](#) [android-asyncTask](#) or [ask your own question](#).