**Download file**

package com.dropbox.android.sample;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.content.DialogInterface.OnClickListener;

import android.graphics.drawable.Drawable;

import android.os.AsyncTask;

import android.widget.ImageView;

import android.widget.Toast;

import com.dropbox.client2.DropboxAPI;

import com.dropbox.client2.ProgressListener;

import com.dropbox.client2.DropboxAPI.Entry;

import com.dropbox.client2.DropboxAPI.ThumbFormat;

import com.dropbox.client2.DropboxAPI.ThumbSize;

import com.dropbox.client2.exception.DropboxException;

import com.dropbox.client2.exception.DropboxIOException;

import com.dropbox.client2.exception.DropboxParseException;

import com.dropbox.client2.exception.DropboxPartialFileException;

import com.dropbox.client2.exception.DropboxServerException;

import com.dropbox.client2.exception.DropboxUnlinkedException;

public class DownloadFile extends AsyncTask<Void, Long, Boolean> {

private Context mContext;

private final ProgressDialog mDialog;

private DropboxAPI<?> mApi;

private String mPath;

private FileOutputStream mFos;

private boolean mCanceled;

private Long mFileLen;

private String mErrorMsg;

private String FILE\_NAME;

public DownloadFile(Context context, DropboxAPI<?> api, String dropboxPath, String filename) {

mContext = context.getApplicationContext();

mApi = api;

mPath = dropboxPath;

FILE\_NAME = filename;

mDialog = new ProgressDialog(context);

mDialog.setMessage("Downloading "+FILE\_NAME+" . . .");

mDialog.setButton("Cancel", new OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

mCanceled = true;

mErrorMsg = "Canceled";

if (mFos != null) {

try {

mFos.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

});

mDialog.show();

}

@Override protected Boolean doInBackground(Void... params) {

try {

if (mCanceled) {

return false;

}

Entry entryfile = mApi.metadata(mPath, 1000, null, true, null);

mFileLen = entryfile.bytes;

try {

File folder = new File("/mnt/sdcard/RestoredFiles\_Dropbox/");

folder.mkdirs();

File savedfile = new File(folder, FILE\_NAME);

mFos = new FileOutputStream(savedfile);

mApi.getFile(mPath, null, mFos, new ProgressListener() {

@Override public long progressInterval() { // Update the progress bar every half-second or so

return 500;

}

@Override public void onProgress(long bytes, long total) {

publishProgress(bytes);

}

});

if (!mCanceled) {

return true;

}

}

catch (FileNotFoundException e) {

e.printStackTrace();

mErrorMsg = "Couldn't create a local file to store the file";

return false;

}

if (mCanceled) {

return false;

}

return true;

} catch (DropboxUnlinkedException e) {

// The AuthSession wasn't properly authenticated or user unlinked.

} catch (DropboxPartialFileException e) {

// We canceled the operation

mErrorMsg = "Download canceled";

} catch (DropboxServerException e) {

// Server-side exception. These are examples of what could happen,

// but we don't do anything special with them here.

if (e.error == DropboxServerException.\_304\_NOT\_MODIFIED) {

// won't happen since we don't pass in revision with metadata

} else if (e.error == DropboxServerException.\_401\_UNAUTHORIZED) {

// Unauthorized, so we should unlink them. You may want to

// automatically log the user out in this case.

} else if (e.error == DropboxServerException.\_403\_FORBIDDEN) {

// Not allowed to access this

} else if (e.error == DropboxServerException.\_404\_NOT\_FOUND) {

// path not found (or if it was the thumbnail, can't be

// thumbnailed)

} else if (e.error == DropboxServerException.\_406\_NOT\_ACCEPTABLE) {

// too many entries to return

} else if (e.error == DropboxServerException.\_415\_UNSUPPORTED\_MEDIA) {

// can't be thumbnailed

} else if (e.error == DropboxServerException.\_507\_INSUFFICIENT\_STORAGE) {

// user is over quota

} else {

// Something else

}

// This gets the Dropbox error, translated into the user's language

mErrorMsg = e.body.userError;

if (mErrorMsg == null) {

mErrorMsg = e.body.error;

}

} catch (DropboxIOException e) {

// Happens all the time, probably want to retry automatically.

mErrorMsg = "Network error. Try again.";

} catch (DropboxParseException e) {

// Probably due to Dropbox server restarting, should retry

mErrorMsg = "Dropbox error. Try again.";

} catch (DropboxException e) {

// Unknown error

mErrorMsg = "Unknown error. Try again.";

}

return false;

}

@Override protected void onProgressUpdate(Long... progress) {

int percent = (int)(100.0\*(double)progress[0]/mFileLen + 0.5);

mDialog.setProgress(percent);

}

@Override protected void onPostExecute(Boolean result) {

mDialog.dismiss();

if (result) {

showToast("File download successfully");

} else {

showToast(mErrorMsg);

}

}

private void showToast(String msg) {

Toast error = Toast.makeText(mContext, msg, Toast.LENGTH\_LONG);

error.show();

}

}