package com.dropbox.android.sample;

import java.io.File;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.graphics.Color;

import android.net.Uri;

import android.os.Bundle;

import android.util.Log;

import android.view.ContextMenu;

import android.view.ContextMenu.ContextMenuInfo;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.AdapterContextMenuInfo;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ImageButton;

import android.widget.ListView;

import android.widget.SimpleAdapter;

import android.widget.TextView;

import android.widget.Toast;

import com.dropbox.client2.DropboxAPI;

import com.dropbox.client2.DropboxAPI.Entry;

import com.dropbox.client2.android.AndroidAuthSession;

import com.dropbox.client2.exception.DropboxException;

import com.dropbox.client2.session.AccessTokenPair;

import com.dropbox.client2.session.AppKeyPair;

import com.dropbox.client2.session.Session.AccessType;

import com.dropbox.client2.session.TokenPair;

public class dropME extends Activity {

private static final String TAG = "dropME";

final static private String APP\_KEY = "02v33p1k5gtvgtm";

final static private String APP\_SECRET = "sh1tfsoskzciacu";

final static private AccessType ACCESS\_TYPE = AccessType.APP\_FOLDER;

final static private String ACCOUNT\_PREFS\_NAME = "prefs";

final static private String ACCESS\_KEY\_NAME = "ACCESS\_KEY";

final static private String ACCESS\_SECRET\_NAME = "ACCESS\_SECRET";

DropboxAPI<AndroidAuthSession> mApi;

// ImageView entryicon;

// TextView entrytext;

ImageButton imagebtn\_up, imagebtn\_down;

ListView listView;

TextView textViewPath;

Boolean remoteFileSystem;

String rootdirpath\_remote = null;

String currentpath\_remote = null;

SimpleAdapter adapter\_remote;

ArrayList<String> history\_remote;

String[] arr3\_remote;

int explorecount\_remote;

boolean nowItsAFile\_remote, nowItsAFolder\_remote;

String contextFile\_remote;

String contextFilename\_remote;

String rootdirpath\_local = null;

String currentpath\_local = null;

SimpleAdapter adapter\_local;

ArrayList<String> history\_local;

String[] arr3\_local;

int explorecount\_local;

boolean nowItsAFile\_local, nowItsAFolder\_local;

File contextFile\_local;

Entry cacheentry = null;

String cachepath = null;

// pd = ProgressDialog.show(this, "Working..", "Searching..", true, true);

// Thread thread = new Thread(this);

// thread.start();

// private Handler handler = new Handler() {

// @Override public void handleMessage(Message msg) {

// pd.dismiss();

//

//

// };

@Override

public boolean onCreateOptionsMenu(Menu menu) {

menu.add(0, 1, 0, "Refresh");

menu.add(0, 2, 0, "Exit");

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

if(item.getItemId()==1)

{

showToast("Refreshing...");

}

else if(item.getItemId()==2)

{

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Are you sure you want to exit?")

.setCancelable(false)

.setPositiveButton("Yes", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dropME.this.finish();

}

})

.setNegativeButton("No", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

dialog.cancel();

}

});

builder.show();

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.test);

rootdirpath\_remote = "/";

currentpath\_remote = rootdirpath\_remote;

rootdirpath\_local = "/mnt/sdcard";

currentpath\_local = rootdirpath\_local;

imagebtn\_up = (ImageButton) findViewById(R.id.buttonUp);

imagebtn\_down = (ImageButton) findViewById(R.id.buttonDown);

listView = (ListView) findViewById(R.id.llist);

textViewPath = (TextView) findViewById(R.id.path);

// entryicon = (ImageView) findViewById(R.id.entryicon);

// entrytext = (TextView) findViewById(R.id.entryname);

registerForContextMenu(listView);

history\_remote = new ArrayList<String>();

explorecount\_remote = 0;

arr3\_remote = new String[] {};

history\_local = new ArrayList<String>();

explorecount\_local = 0;

arr3\_local = new String[] {};

remoteFileSystem = true;

listView.setAdapter(adapter\_remote); // ?

textViewPath.setText(currentpath\_remote);

imagebtn\_down.setBackgroundColor(Color.LTGRAY);

imagebtn\_down.setEnabled(false);

imagebtn\_up.setBackgroundColor(Color.TRANSPARENT);

imagebtn\_up.setEnabled(true);

AndroidAuthSession session = buildSession();

mApi = new DropboxAPI<AndroidAuthSession>(session);

// if (mLoggedIn) {

// // logOut();

// }

// else {

// }

SharedPreferences shared = getSharedPreferences("config", 0);

if (shared.getBoolean("hasRunBefore", false)==true)

{

explore\_Remote(currentpath\_remote);

}

else

{

mApi.getSession().startAuthentication(dropME.this);

explore\_Remote(currentpath\_remote);

SharedPreferences.Editor editor = shared.edit();

editor.putBoolean("hasRunBefore", true);

editor.commit();

}

}

public void onclickUp(View v) {

remoteFileSystem = false;

textViewPath.setText(currentpath\_local); // ?

explore\_Local(currentpath\_local);

listView.setAdapter(adapter\_local);

imagebtn\_up.setBackgroundColor(Color.LTGRAY);

imagebtn\_up.setEnabled(false);

imagebtn\_down.setBackgroundColor(Color.TRANSPARENT);

imagebtn\_down.setEnabled(true);

}

public void onclickDown(View v) {

remoteFileSystem = true;

textViewPath.setText(currentpath\_remote); // ?

explore\_Remote(currentpath\_remote);

listView.setAdapter(adapter\_remote);

imagebtn\_down.setBackgroundColor(Color.LTGRAY);

imagebtn\_down.setEnabled(false);

imagebtn\_up.setBackgroundColor(Color.TRANSPARENT);

imagebtn\_up.setEnabled(true);

}

public void explore\_Remote(String path\_remote) {

final Entry entry;

try {

if (cacheentry != null && cachepath != null) {

currentpath\_remote = cachepath;

history\_remote.add(explorecount\_remote, currentpath\_remote);

entry = cacheentry;

} else {

currentpath\_remote = path\_remote;

history\_remote.add(explorecount\_remote, path\_remote);

entry = mApi.metadata(currentpath\_remote, 1000, null, true,

null);

}

cacheentry = null;

cachepath = null;

if (!entry.isDir || entry.contents == null) {

showToast("File or empty directory");

} else {

List<Entry> listofchildren = entry.contents;

ArrayList<Map<String, String>> arrlist = new ArrayList<Map<String, String>>();

Map<String, String> m = null;

ArrayList<String> dynamiclist = new ArrayList<String>();

for (Entry ent : listofchildren) {

dynamiclist.add(ent.fileName());

m = new HashMap<String, String>();

if (ent.isDir) {

m.put("key", String.valueOf(R.drawable.ic\_folder));

m.put("title", ent.fileName());

} else if (!ent.isDeleted) {

m.put("key", String.valueOf(R.drawable.ic\_file));

m.put("title", ent.fileName());

}

arrlist.add(m);

}

String[] arr = new String[] {};

arr = dynamiclist.toArray(arr);

// adapter\_remote = new

// ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,

// arr);

adapter\_remote = new SimpleAdapter(this, arrlist,

R.layout.rowdata, new String[] { "key", "title" },

new int[] { R.id.entryicon, R.id.entryname });

listView.setAdapter(adapter\_remote);

final String[] arr2 = arr;

arr3\_remote = arr;

listView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

String nextpath = currentpath\_remote + "/"

+ arr2[position];

try {

Entry subentry = mApi.metadata(nextpath, 1000,

null, true, null);

if (subentry.isDir) {

cachepath = nextpath;

cacheentry = subentry;

explorecount\_remote++;

explore\_Remote(currentpath\_remote);

textViewPath.setText(currentpath\_remote);

} else {

String selectedfile = subentry.fileName();

textViewPath.setText(subentry.path);

}

} catch (DropboxException e) {

e.printStackTrace();

}

}

});

}

} catch (DropboxException e) {

e.printStackTrace();

}

}

public void explore\_Local(String path\_local) {

history\_local.add(explorecount\_local, path\_local);

File f = new File(path\_local);

File[] files = f.listFiles();

if (files == null) {

showToast("File or empty directory");

} else {

ArrayList<Map<String, String>> arrlist = new ArrayList<Map<String, String>>();

Map<String, String> m = null;

ArrayList<String> dynamiclist = new ArrayList<String>();

for (File file : files) {

dynamiclist.add(file.getName());

m = new HashMap<String, String>();

if (file.isDirectory()) {

m.put("key", String.valueOf(R.drawable.ic\_folder));

m.put("title", file.getName());

} else if (file.isFile()) {

m.put("key", String.valueOf(R.drawable.ic\_file));

m.put("title", file.getName());

}

arrlist.add(m);

}

String[] arr = new String[] {};

arr = dynamiclist.toArray(arr);

// adapter = new

// ArrayAdapter<String>(this,android.R.layout.simple\_list\_item\_1,

// arr);

adapter\_local = new SimpleAdapter(this, arrlist, R.layout.rowdata,

new String[] { "key", "title" }, new int[] {

R.id.entryicon, R.id.entryname });

listView.setAdapter(adapter\_local);

final String[] arr2 = arr;

arr3\_local = arr;

listView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

if (new File(currentpath\_local + File.separator

+ arr2[position]).listFiles() != null) {

currentpath\_local = currentpath\_local + File.separator

+ arr2[position];

explorecount\_local++;

explore\_Local(currentpath\_local);

textViewPath.setText(currentpath\_local);

} else {

File file = new File(currentpath\_local + File.separator

+ arr2[position]);

// String selectedfile = file.getName();

textViewPath.setText(file.getAbsolutePath());

}

}

});

}

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

if (v.getId() == R.id.llist) {

if (remoteFileSystem) {

AdapterContextMenuInfo info = (AdapterContextMenuInfo) menuInfo;

String name = arr3\_remote[info.position];

contextFilename\_remote = name;

contextFile\_remote = currentpath\_remote + "/" + name;

menu.setHeaderTitle(name);

menu.add(0, 1, 0, "Download");

menu.add(0, 2, 0, "Delete");

} else {

AdapterContextMenuInfo info = (AdapterContextMenuInfo) menuInfo;

String name = arr3\_local[info.position];

contextFile\_local = new File(currentpath\_local + File.separator

+ name);

menu.setHeaderTitle(name);

menu.add(0, 1, 0, "Upload");

menu.add(0, 2, 0, "open");

menu.add(0, 3, 0, "Delete");

}

}

}

@Override

public boolean onContextItemSelected(MenuItem item) {

if (remoteFileSystem) {

if (item.getItemId() == 1) { // download remote file

DownloadFile download = new DownloadFile(this, mApi,

contextFile\_remote, contextFilename\_remote);

download.execute();

}

else if (item.getItemId() == 2) { // delete remote file

try {

showToast("Deleting....");

mApi.delete(contextFile\_remote);

explore\_Remote(currentpath\_remote);

showToast("File Deleted!");

} catch (DropboxException e) {

e.printStackTrace();

}

}

}

else {

if (item.getItemId() == 1) { // upload local file

UploadFile upload = new UploadFile(this, mApi,

"/BackupFiles\_Dropbox/", contextFile\_local);

upload.execute();

}

else if(item.getItemId()==2)

{

Intent intent = new Intent(android.content.Intent.ACTION\_VIEW);

Uri uri = Uri.fromFile(contextFile\_local);

intent.setDataAndType(uri, "\*/\*");

// if(contextFile\_local.getName().endsWith(".jpg")||contextFile\_local.getName().endsWith(".jpeg")) {

// intent.setDataAndType(uri, "\*/\*");

// }

// else if(contextFile\_local.getName().endsWith(".mp3")||contextFile\_local.getName().endsWith(".ogg")) {

// intent.setDataAndType(uri,"media/\*");

// }

// else if(contextFile\_local.getName().endsWith(".mp4")||contextFile\_local.getName().endsWith(".avi")||contextFile\_local.getName().endsWith(".flv")) {

// intent.setDataAndType(uri,"media/\*");

// }

// intent.setDataAndType(uri,"text/\*");

// }

//

startActivity(intent);

// showToast(contextFile\_local.getAbsolutePath());

}

else if (item.getItemId() == 3) {

boolean deleted = contextFile\_local.delete();

if (deleted)

{ showToast("deleted");

explore\_Local(currentpath\_local);

showToast("folder contains sub datas");

}

}

}

return true;

}

@Override

public void onBackPressed() {

// showToast("REM: "+remoteFileSystem+"\n"+

// "LOC: "+currentpath\_local+"\n"+

// "REM: "+currentpath\_remote);

if (remoteFileSystem) {

String[] arr = new String[] {};

arr = history\_remote.toArray(arr);

explorecount\_remote--;

if (explorecount\_remote >= 0) {

currentpath\_remote = arr[explorecount\_remote];

explore\_Remote(currentpath\_remote);

textViewPath.setText(currentpath\_remote);

} else {

super.onBackPressed();

}

} else {

String[] arr = new String[] {};

arr = history\_local.toArray(arr);

explorecount\_local--;

if (explorecount\_local >= 0) {

currentpath\_local = arr[explorecount\_local];

explore\_Local(currentpath\_local);

textViewPath.setText(currentpath\_local);

} else {

super.onBackPressed();

}

}

}

@Override

protected void onResume() {

super.onResume();

AndroidAuthSession session = mApi.getSession();

// The next part must be inserted in the onResume() method of the

// activity from which session.startAuthentication() was called, so

// that Dropbox authentication completes properly.

if (session.authenticationSuccessful()) {

try {

// Mandatory call to complete the auth

session.finishAuthentication();

// Store it locally in our app for later use

TokenPair tokens = session.getAccessTokenPair();

storeKeys(tokens.key, tokens.secret);

// setLoggedIn(true);

} catch (IllegalStateException e) {

showToast("Couldn't authenticate with Dropbox:"

+ e.getLocalizedMessage());

Log.i(TAG, "Error authenticating", e);

}

}

}

private void showToast(String msg) {

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

error.show();

}

private AndroidAuthSession buildSession() {

AppKeyPair appKeyPair = new AppKeyPair(APP\_KEY, APP\_SECRET);

AndroidAuthSession session;

String[] stored = getKeys();

if (stored != null) {

AccessTokenPair accessToken = new AccessTokenPair(stored[0],

stored[1]);

session = new AndroidAuthSession(appKeyPair, ACCESS\_TYPE,

accessToken);

} else {

session = new AndroidAuthSession(appKeyPair, ACCESS\_TYPE);

}

return session;

}

/\*\*

\* Shows keeping the access keys returned from Trusted Authenticator in a

\* local store, rather than storing user name & password, and

\* re-authenticating each time (which is not to be done, ever).

\*/

private void storeKeys(String key, String secret) {

// Save the access key for later

SharedPreferences prefs = getSharedPreferences(ACCOUNT\_PREFS\_NAME, 0);

Editor edit = prefs.edit();

edit.putString(ACCESS\_KEY\_NAME, key);

edit.putString(ACCESS\_SECRET\_NAME, secret);

edit.commit();

}

/\*\*

\* Shows keeping the access keys returned from Trusted Authenticator in a

\* local store, rather than storing user name & password, and

\* re-authenticating each time (which is not to be done, ever).

\*

\* @return Array of [access\_key, access\_secret], or null if none stored

\*/

private String[] getKeys() {

SharedPreferences prefs = getSharedPreferences(ACCOUNT\_PREFS\_NAME, 0);

String key = prefs.getString(ACCESS\_KEY\_NAME, null);

String secret = prefs.getString(ACCESS\_SECRET\_NAME, null);

if (key != null && secret != null) {

String[] ret = new String[2];

ret[0] = key;

ret[1] = secret;

return ret;

} else {

return null;

}

}

}