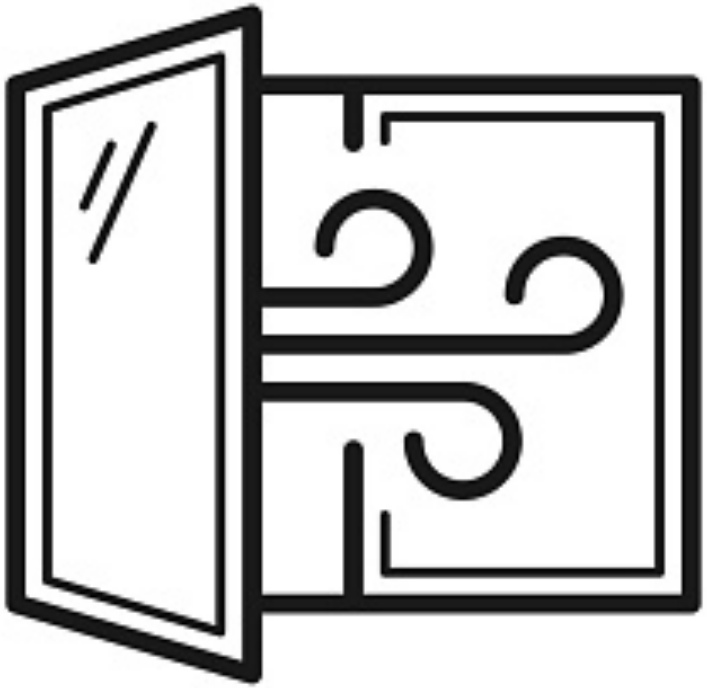
Android App Programming

Lecture 5: Databases
November 16, 2021

Anastasia Bezerianos
Université Paris-Saclay

course based in part on that of Thomas Nowak

in class



Reminder:
next week
TD noté

TD noté 1

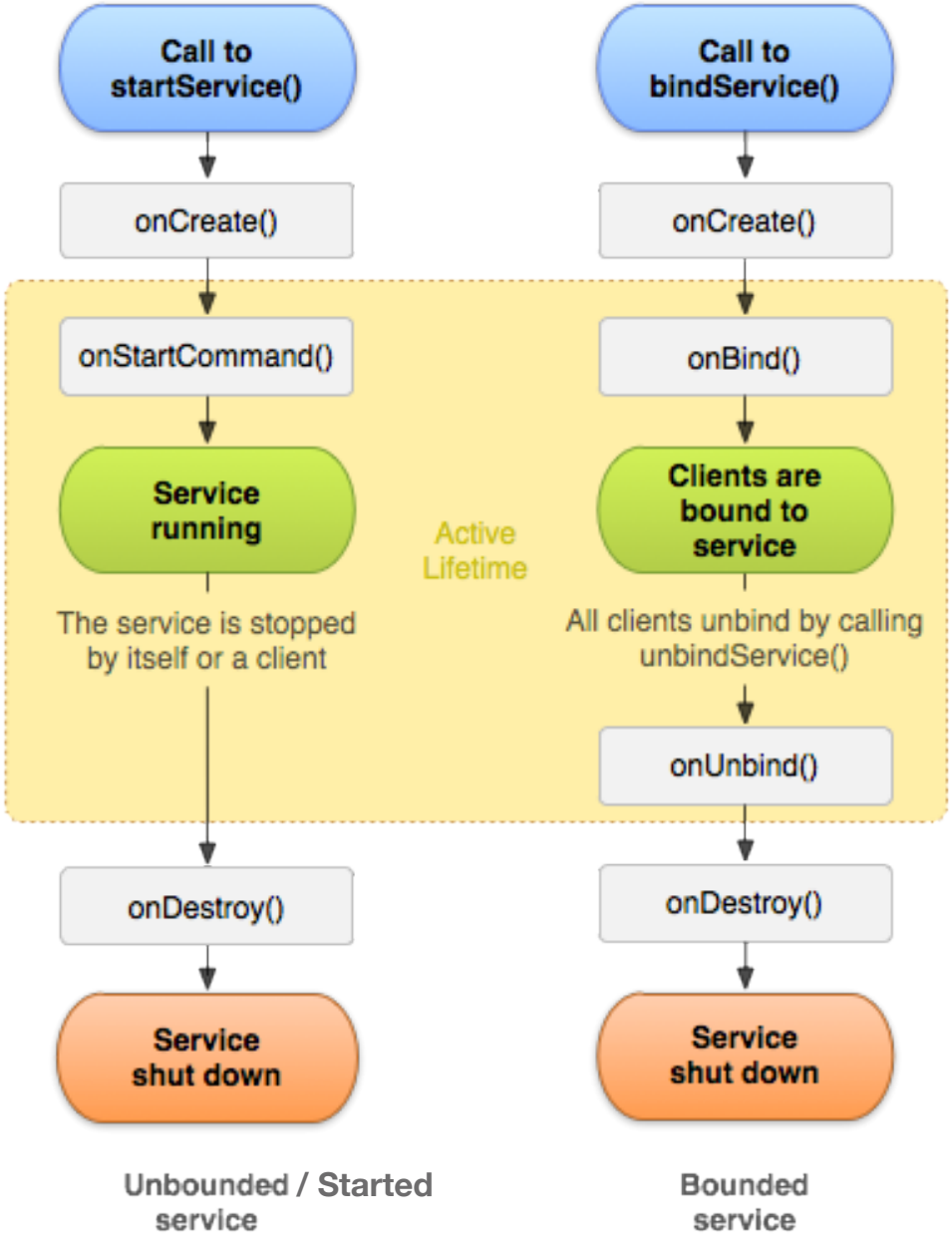
- TD noté on 23/11
- Time needed 2h
- You will work alone (cannot discuss with others)
- Can use any material from class (transparencies, lab solutions, videos), the book, or the internet
- BUT the final solution should be yours

Last lecture recap

Services

- for background tasks
- ex: downloads (that should continue after the app quit)
- ex: music player
- ex: polling an SMTP for new emails

Service Lifecycle



Databases

Saving State

- several possibilities:
  - passing bundles (ugh)
  - write into a file (uuugh)
  - preferences
    ```
    SharedPreferences prefs =
    getPreferences(MODE_PRIVATE)
    ```
  - databases
  - more on storage at https://developer.android.com/training/data-storage

Databases

- SQL databases:
  - relational data model
  - tables (primary / unique key)
  - proven, robust, optimized
- NoSQL databases:
  - different data model, e.g., objects, graphs
  - less data translation necessary

Databases

- Physical location of data varies:
  - locally (on the phone)
  - remote (on one or more servers on the internet)
    - e.g., Firebase (https://developer.android.com/studio/write/firebase)

SQL

- Structured Query Language
- standard language used by most relational database management systems (DB in forms of tables)
- common SQL commands / queries:
- (does not expect result) `create table <DB table name> (<id> integer primary key autoincrement, <columnA> text, <columnB> text, …)`
- (expects result) `select * from <DB table name>`
- (expects result) `select * from <DB table name> where <columnA>="a_value"`

SQLite

- DBMS integrated into Android: SQLite
- packaged into the app
  - runs in the same process
  - no connection setup necessary
- use the class `SQLiteDatabase` included in Android

SQLite in Android

- create and manage your DB with a class that extends `SQLiteOpenHelper` that helps you create & manage the DB and run SQL statements / queries
- use `db.rawQuery` for SQL statements that expect results (e.g., queries)
- for SQL statements without result, call `db.execSQL`
- there are specialized methods for some operations, e.g., `db.insert, db.close, db.update, …`

SQLiteOpenHelper

- You can access your DB with `getReadableDatabase()`, `getWritableDatabase()`
- use objects of the class `ContentValues` to create a DB record and insert it
- you can loop over result of a `rawQuery` with a `Cursor` object (it is like an iterator over all returned DB records).
- Useful methods of Cursor are `moveToFirst()`, `moveToNext()`, `getColumnIndex(…)`, the check `isAfterLast() == false`, …

Coding demo

Lab 5: Ex 1

DBHelper

17/11: fixed bug/typo. The old slides had the SQL statement malformed - see this as a String to pass in SQL, the names of the table & columns need to be passed as variable names. Also verify the spaces. It should read:
`create table countries (id integer primary key autoincrement, name text, currency text)`

You may need to delete the old DB that was created using the wrong table / column names from your device. One way is to WipeData on your emulator; and delete the application on a physical device.

```
package fr.ups.simpledb;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

import androidx.annotation.Nullable;
import java.util.ArrayList;

public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "MyDBcountries.db";
    public static final String COUNTRIES_TABLE_NAME = "countries";
    public static final String COUNTRIES_COLUMN_ID = "id";
    public static final String COUNTRIES_COLUMN_NAME = "name";
    public static final String COUNTRIES_COLUMN_CURRENCY = "currency";

    public DBHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(
            "create table " + QUOTES_TABLE_NAME +
                    " (" + QUOTES_COLUMN_ID + " integer primary key autoincrement, "
                        + QUOTES_COLUMN_NAME +" text)"
        );
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS country");
        onCreate(sqLiteDatabase);
    }

    public boolean insertCountry (String name, String currency) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("name", name);
        contentValues.put("currency", currency);
        db.insert("countries", null, contentValues);
        return true;
    }

    public ArrayList<String> getAllCountries() {
        ArrayList<String> array_list = new ArrayList<String>();

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res =  db.rawQuery( "select * from "+ COUNTRIES_TABLE_NAME, null );
        res.moveToFirst();

        while(res.isAfterLast() == false){
            @SuppressLint("Range") String id = res.getString(res.getColumnIndex(COUNTRIES_COLUMN_ID));
            @SuppressLint("Range") String name = res.getString(res.getColumnIndex(COUNTRIES_COLUMN_NAME));
            @SuppressLint("Range") String currency = res.getString(res.getColumnIndex(COUNTRIES_COLUMN_CURRENCY));

            array_list.add(id + " " + name + " " + currency);
            res.moveToNext();
        }
        return array_list;
    }
}
```

My DB names (of the DB, the table and columns)

constructor that defines the context (activity that calls this class), the DB name, access factory (null), and version of DB

run an SQL statement without result that creates the table with three columns, one is the primary key that is auto-created

this updates the DB to a new version

get access to the DB with `getWritableDatabase()`.
Use an object ContentValue to create pairs of (column, value) and insert in the DB.

an SQL query that returns results and saves them in a Cursor object. Starts at the beginning of the results (`moveToFirst`) and while it has not reached the end (`isAfterLast()==false`), it moves to the next one (`moveToNext`).
It accesses values with `getColumnIndex()`

MainActivity

```java
package fr.ups.simpledb;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    DBHelper mydb;
    TextView txt_db;
    EditText txt_name;
    EditText txt_currency;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mydb = new DBHelper(this);

        txt_name = (EditText) findViewById(R.id.editCountry);
        txt_currency = (EditText) findViewById(R.id.editCurrency);

        txt_db = (TextView) findViewById(R.id.DB_entries);
        txt_db.setMovementMethod(new ScrollingMovementMethod());

        print_countries();
    }

    void print_countries(){
        ArrayList<String> all_countries = mydb.getAllCountries();

        txt_db.setText(" ");
        for (String c : all_countries){
            txt_db.append(c+"\n");
        }
    }

    public void add_DB_entry(View view) {
        mydb.insertCountry(txt_name.getText().toString(), txt_currency.getText().toString());

        print_countries();
    }
}
```

define an object of my DBHelper class

creating the DB helper class object

aside, making this TextView scrollable. Also add in the xml definition of the TextView
android:gravity="bottom"
android:scrollbars="vertical"

methods defined in my DGHelper to print and to insert countries

Room

- (aside)
- alternative way to access SQLite
- the Room library is an abstraction layer over SQLite
- provides compile-time query verification
- we will focus on SQLite, but if you want to learn more on Room https://developer.android.com/training/data-storage/room

Lab 5

2. Re-visit Chuck Norris and store quotes in DB

17/11: Please look at Lab 3 Ex1, and the description of the solution about how to add:
- the appropriate libraries in the dependencies section in you app's build.grade (Retrofit2, Picasso)
- allow internet permission in your AndroidManifest.xml

Coding hints

Lab 5: Ex 2

Sharing DBs

- For Ex 2, you may want to access the DB from multiple activities
- use a Singleton to Instantiate the SQLiteOpenHelper
- declare a Private Static instance of the helper internally in the class - this will be the only instance / object of this class
- Make its constructor private so no other class can access it and create the instance / object
- Provide a way for classes outside (eg other activities) to access this instance

DBHelper

```
public class DBHelper extends SQLiteOpenHelper {

    private static DBHelper sInstance;

    public static final String DATABASE_NAME = "myDB.db";
    …

    public static synchronized DBHelper getInstance(Context context) {

        // Use the application context, which will ensure that you
        // don't accidentally leak an Activity's context.
        if (sInstance == null) {
            sInstance = new DBHelper(context.getApplicationContext());
        }
        return sInstance;
    }

    /**
     * Constructor should be private to prevent direct instantiation.
     * make call to static method "getInstance()" instead.
     */
    private DBHelper(@Nullable Context context) {
        super(context, DATABASE_NAME, null, 1);
    }

…
…
…
…
}
```

singleton reference to a unique object of this class

way for other classes to access the unique instance

if the instance exists return it, else create it

private constructor to make sure no other class can access it …

other activity that wants to access the DBHelper

```
package fr.ups.test_db;

import androidx.appcompat.app.AppCompatActivity;


public class CreateDBActivity extends AppCompatActivity {

    DBHelper mydb;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_db);

        mydb = DBHelper.getInstance(this);
    }

…
…

}
```

way to access the DBHelper
(instead of **new** DBHelper(**this**);

Questions