SQLITE +Volley Library

Dieudonne U

SQLITE DATABASE

SQLite is an **open-source relational database** i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.

It is embedded in android bydefault. So, there is no need to perform any database setup or administration task.

SQLiteOpenHelper class provides the functionality to use the SQLite database.

Constructors of SQLiteOpenHelper class

Constructor	**Description**
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	creates an object for creating, opening and managing the database.
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	creates an object for creating, opening and managing the database. It specifies the error handler.

Some of Methods of SQLiteOpenHelper class

Method	Description
public abstract void onCreate(SQLiteDatabase db)	called only once when database is created for the first time.
public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be upgraded.
public synchronized void close ()	closes the database object.
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be downgraded.

SQLiteDatabase class

- It contains methods to be performed on sqlite database such as create, update, delete, select etc.
- Methods of SQLiteDatabase class
- There are many methods in SQLiteDatabase class. Some of them are as follows:

Method	Description
void execSQL(String sql)	executes the sql query not select query.
long insert(String table, String nullColumnHack, ContentValues values)	inserts a record on the database. The table specifies the table name, nullColumnHack doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	updates a row.
Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	returns a cursor over the resultset.

Cursors

Cursors are what contain the result set of a query made against a database in Android. The Cursor class has an API that allows an app to read (in a type-safe manner) the columns that were returned from the query as well as iterate over the rows of the result set

Reading Cursor Data

- Once a cursor has been returned from a database query, an app needs to iterate over the result set and read the column data from the cursor.
- Internally, the cursor stores the rows of data returned by the query along with a position that points to the current row of data in the result set.
-
- When a cursor is returned from a query() method, its position points to the spot before the first row of data.
- This means that before any rows of data can be read from the cursor, the position must be moved to point to a valid row of data

Cursor Methods

- The Cursor class provides the following methods to manipulate its internal position:
- boolean Cursor.move(int offset): Moves the position by the given offset
- boolean Cursor.moveToFirst(): Moves the position to the first row
- boolean Cursor.moveToLast(): Moves the position to the last row

Methods

- boolean Cursor.moveToNext(): Moves the cursor to the next row relative to the current position
- boolean Cursor.moveToPosition(int position): Moves the cursor to the specified position
- Cursor.moveToPrevious(): Moves the cursor to the previous row relative to the current position
- Each move() method returns a boolean to indicate whether the operation was successful or not. This flag is useful for iterating over the rows in a cursor.

Coding

Create the Sqlite database App to save contacts.

Display Contacts in RecycleView

Volley

- Volley is an HTTP library developed by Google to ease networking tasks in Android Applications. Volley supersedes Java’s **java.net.HttpURLConnection** class and Apache’s **org.apache.http.client** in handling network requests. Volley can handle almost each and everything you will need to do over the network, it handles HTTP request and also manages the async tasks that you need to use while working with canonical networking classes.

Features of Android Volley Library :-

- Volley manages network request automatically and without you using any **AsyncTask**. It offers multiple concurrent network connections, resulting in faster and efficient network operations.
- Volley caches all network requests, utilizing the previous results in the case of change in activity configuration. However due to the same reason, it is only good for small Volley and not suitable for large download or streaming operations.
- It has a powerful cancellation API. You can fully control the request cancellation , or you can set blocks or scopes of requests to cancel.
- Volley allows you to control and order the network requests making it easy to populate UI elements in an orderly manner , useful in case of social networking apps like facebook , twitter. It also provides support for request prioritization.
- Volley has built in debugging and tracing tools which allow user to get to the root of the error.

Adding Volley Library in a Project

- To add Volley to your project add the following dependency in your App's build.gradle file

```
implementation  'com.android.volley:volley:1.1.1’
```

- Add Internet Permission

Add the following permission to your AndroidManifest.xml file

AndroidManifest.xml

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

build.gradle

```
android {
   useLibrary 'org.apache.http.legacy'
}

dependencies {
 implementation 'com.android.volley:volley:1.0.0'
 implementation 'org.apache.httpcomponents:httpcore:4.4.1'
Implementation  'org.apache.httpcomponents:httpclient:4.5'
}
```

Volley Key Classes

The following are the Key classes of Volley:

– RequestQueue: A Queue containing the Network/HTTP Requests that needs to be made.

– Request: A Base Class which contains Network related information like HTTP Methods.

– StringRequest: HTTP Request where the response is parsed a String.

– JsonObjectRequest: HTTP Request where the response is JSONObject.

Getting Started with Android Volley

At first make a RequestQueue, which holds the HTTP Requests.

Ideally, the RequestQueue should be made once and then referred to it across the Application.

The Application is an ideal place to make it.

```
RequestQueue queue = Volley.newRequestQueue(this);
```

Making GET Requests

Making GET Requests is simple. The example below uses JsonObjectRequest. It prepares a JsonObjectRequest and passes and then adds it to RequestQueue.

The JsonObject accepts 4 parameters

(Http method, Url, Json values, Response Listener – Invoked on success, Error Listener – Invoked on failure).

Example

```
// prepare the Request
JsonObjectRequest getRequest =
new JsonObjectRequest (
    Request.Method.GET,//method to be used
    url,                //url to get data
    null,               // Json values
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            Log.d("Response", response.toString());
        }
    },
new Response.ErrorListener() {//if there is an error
        @Override
        public void onErrorResponse(VolleyError error) {
            // error
            Log.d("Error.Response", response);
      }
   }
);
// add it to the RequestQueue
queue.add(getRequest);
```

Making POST Requests

- For a POST request, to add form parameters/values, the getParams() method needs to be overridden and a Map needs to be returned.

Example

```
StringRequest postRequest =
new StringRequest(
    Request.Method.POST,//method to be used
    url, //url to post data
    new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            // response
            Log.d("Response", response);
        }
    },
new Response.ErrorListener() {//if there is an error
     @Override
     public void onErrorResponse(VolleyError error) {
        // error
        Log.d("Error.Response", response);
    }
  }
)
```

```
{
    @Override
    protected Map<String, String> getParams()
    {
            Map<String, String>  params = new HashMap<String,
String>();
            params.put(“param1", “value1");
            params.put(“param2", “value2");

            return params;
    }
};
queue.add(postRequest);
```

Assignment (LAB 3) deadline:6-may-2019

- Modify this Lab and add the following functionalities

1. Searching from the list
2. Updating contact
3. Deleting the contact
4. Calling one member of the contact
5. Search one contact
6. Modify the GUI