Andhra Pradesh State Skill Development Corporation

ANDROID APPLICATION DEVELOPMENT

SQLITE DATABASE

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.

Here, we are going to see the example of sqlite to store and fetch the data. Data is displayed in the TextView.

SQLiteOpenHelper class provides the functionality to use the SQLite database.

SQLiteOpenHelper class

The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management. For performing any database operation, you have to provide the implementation of onCreate() and onUpgrade() methods of SQLiteOpenHelper class.

Constructors of SQLiteOpenHelper class

There are two 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.

Methods of SQLiteOpenHelper class

There are many methods in SQLiteOpenHelper class. Some of them are as follows:

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.

Example of android SQLite database

Let's see the simple example of android sqlite database.

SqliteDbExample

- In the above diagram we have taken two values i.e Name and Mobileno, Our requirement is we should have to store the Name and mobileno in the sqlite database.
- let's create a new Android Studio project with the name of **SqliteDbExample**
- Take a **Empty Activity**
- Design the screen as per the above diagram

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your name"
        android:id="@+id/nameEditText"
        />
```

```
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter your Mobileno"
        android:id="@+id/mobileEditText"
        />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center"
        android:layout_marginTop="10dp"
        >
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="save"
            android:id="@+id/saveButton"
            />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Retrive"
            android:id="@+id/retriveButton"
            />
    </LinearLayout>
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Result"
        android:textSize="25sp"
        android:textStyle="bold"
        android:textColor="#000000"
        android:id="@+id/resultTextview"
        />
    </ScrollView>
</LinearLayout>
```

MainActivity.Java

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    EditText et_name,et_mobileno;
    TextView tv_result;
    Button saveButton,retriveButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        et_name = findViewById(R.id.nameEditText);
        et_mobileno = findViewById(R.id.mobileEditText);
        tv_result = findViewById(R.id.resultTextview);
        saveButton = findViewById(R.id.saveButton);
        retriveButton = findViewById(R.id.retriveButton);
        saveButton.setOnClickListener(this);
        retriveButton.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        switch (view.getId()){
            case R.id.saveButton:
                saveData();
                break;
            case R.id.retriveButton:
                retriveData();
                break;
        }

    }

    private void saveData() {
        String name = et_name.getText().toString();
        String mobileno = et_mobileno.getText().toString();
    }

    private void retriveData() {
    }

}
```

- It's time to create a **DbHelper** class extends with **SqliteOpenHelper** class
- After creating the class implement the required methods i.e **onCrete()** and **onUpgrade()**
- And also implement the Constructor mathing with super class,

DbHelper.Java

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

```
import androidx.annotation.Nullable;

public class DbHelper extends SQLiteOpenHelper {

    public DbHelper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {

    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

    }
}
```

- Let's create some static constant String varibles and int varibles in the DbHelper class.

```
    public static final String DB_NAME="SqlDb";
    public static final String TABLE_NAME="users";
    public static final String COL_0="id";
    public static final String COL_1="name";
    public static final String COL_2="mobileno";
    public static final int VERSION=1;
    public static final String TABLE_QUERY = "create table if not exists "+TABLE_NAME+"("+COL_0+" integer primary key autoincrement,"+COL_1+" varchar(200),"+COL_2+" varchar(10));";
```

ContentValues Class ContentValue class lets you put information inside an object in the form of Key-Value pairs for columns and their value. The object can then be passed to the insert() method of an instance of the SQLiteDatabase class to insert or update your WritableDatabase.

Cursor Class Cursor on the other hand, is a kind of a pointer(not the C/C++ one) which you get after calling the query() method on an instance of a ReadableDatabase of the SQLiteDatabase class. The cursor points to individual rows retuned after querying the database and you can use several methods like getInt(), getString() etc to extract information from specific columns of the current row. You can also traverse rows using methods like moveToNext(), moveToFirst() etc on the cursor object

- let's add two methods in the DBHelper class for,to insert the data into the table and retrive the data from the table. i.e **insertMyData()** and **retriveMydata()**

```
public long insertMyData(ContentValues contentValues){
        SQLiteDatabase db = getWritableDatabase();
        long i = db.insert(TABLE_NAME,null,contentValues);
        return i;
    }
```

```
public Cursor retriveMydata(){
    SQLiteDatabase db = getReadableDatabase();
    Cursor c = db.rawQuery("SELECT *FROM "+TABLE_NAME,null);
    return c;
}
```

- Now its time to call that DbHelper class into the MainActivity.Java ```java
  DbHelper helper = new DbHelper(this);

- when we click the saveButton it will invokes **saveData()** method, So now we should have to assign users data to the ContentValues class object. After that we can send send our ContentValues calss object to the **insertMyData()** which is available in the DbHelper class.java

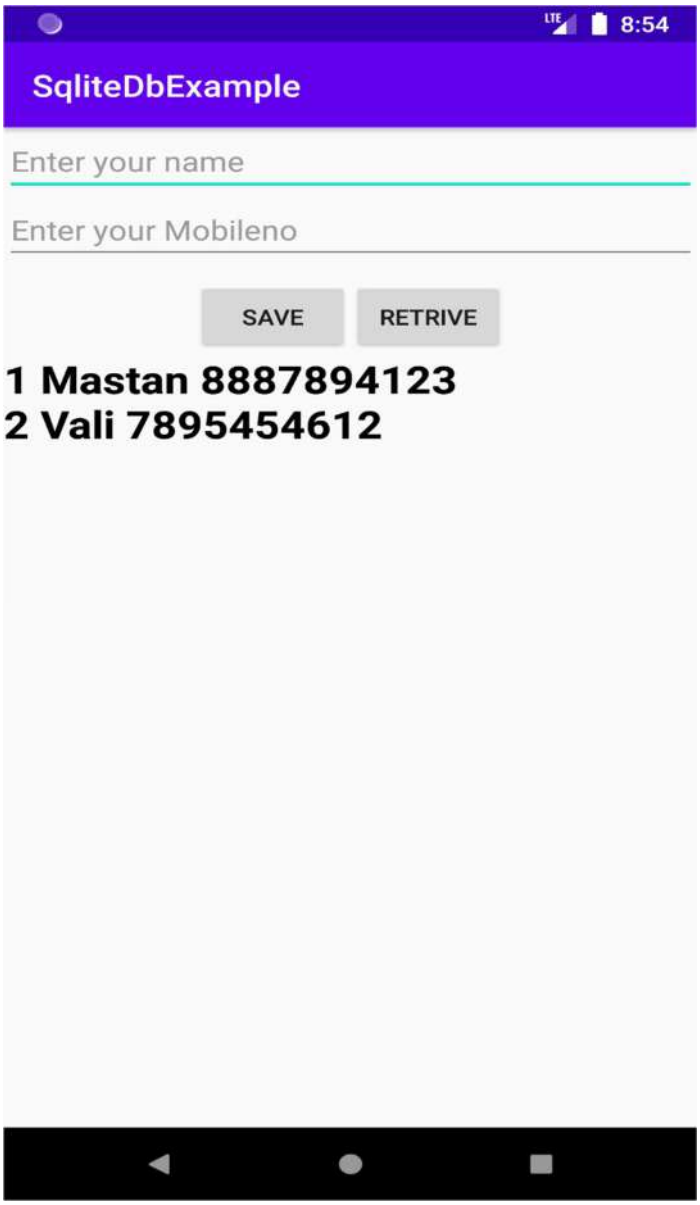
```
private void saveData() {
    String name = et_name.getText().toString();
    String mobileno = et_mobileno.getText().toString();
    ContentValues values=new ContentValues();
    values.put(DbHelper.COL_1,name);
    values.put(DbHelper.COL_2,mobileno);
    long i = helper.insertMyData(values);
    if(i>0){
        Toast.makeText(this,
                "Data inserted Successfully", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText(this,
                "Insertion Failed", Toast.LENGTH_SHORT).show();
    }

}
```

- Now we should have to retrive the data from the table

```
private void retriveData() {
    Cursor c = helper.retriveMydata();
    StringBuilder builder = new StringBuilder();
    while (c.moveToNext()){
        builder.append(c.getInt(0)+" "+c.getString(1)+" "+c.getString(2)+"\n");
    }
    tv_result.setText(builder.toString());
}
```

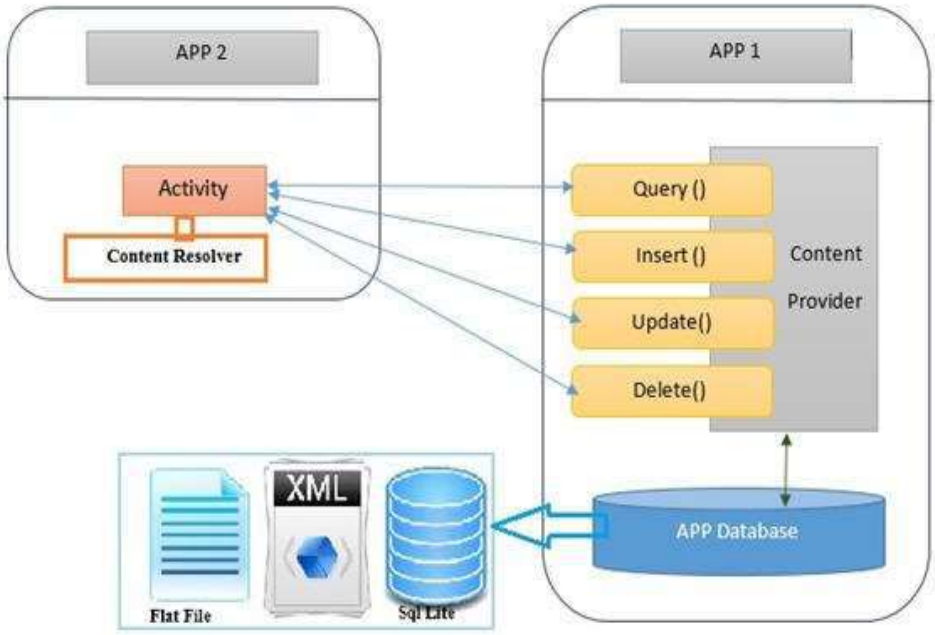
OutPut



ContentProvider

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. A content provider can use different ways to store its data and the data can be stored in a database, in files, or even over a network. sometimes it is required to share data across applications. This is where content providers become very useful.

Content providers let you centralize content in one place and have many different applications access it as needed. A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods. In most cases this data is stored in an SQlite database.



A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

```
import android.content.ContentProvider;

public class MyApplication extends ContentProvider {
}
```

Content URIs

To query a content provider, you specify the query string in the form of a URI which has following format −

<prefix>://<authority>/<data_type>/<id>

Here is the detail of various parts of the URI −

Sr.No	Part & Description
1	**prefix** This is always set to content://
2	**authority** This specifies the name of the content provider, for example contacts, browser etc. For third-party content providers, this could be the fully qualified name,such as com.example.statusprovider
3	**data_type** This indicates the type of data that this particular provider provides. For example, if you are getting all the contacts from the Contacts content provider, then the data path would be people and URI would look like thiscontent://contacts/people
4	**id** This specifies the specific record requested. For example, if you are looking for contact number 5 in the Contacts content provider then URI would look like this content://contacts/people/5.

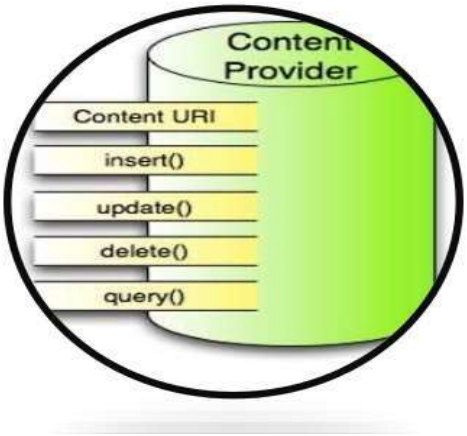
Create Content Provider

This involves number of simple steps to create your own content provider.

- First of all you need to create a Content Provider class that extends the ContentProviderbaseclass.
- Second, you need to define your content provider URI address which will be used to access the content.
- Next you will need to create your own database to keep the content. Usually, Android uses SQLite database and framework needs to override onCreate() method which will use SQLite Open Helper method to create or open the provider's database. When your application is launched, the onCreate() handler of each of its Content Providers is called on the main application thread.
- Next you will have to implement Content Provider queries to perform different database specific operations.
- Finally register your Content Provider in your activity file using tag.

Here is the list of methods which you need to override in Content Provider class to have your Content Provider working −

Android Content Provider



- onCreate() This method is called when the provider is started.
- query() This method receives a request from a client. The result is returned as a Cursor object.
- insert()This method inserts a new record into the content provider.
- delete() This method deletes an existing record from the content provider.
- update() This method updates an existing record from the content provider.
- getType() This method returns the MIME type of the data at the given URI.

In SqliteDbExample Projcet just create a one ContentProvider classs with the name of **MyApplication.java**. After adding the contentprovider class just inialize the DbHelper class with in the oncreate method. After that just call the retriveMydata() method with in the query() method.

```
package com.example.sqlitedbexample;

import android.content.ContentProvider;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

public class MyApplication extends ContentProvider {
    DbHelper helper;
    @Override
    public boolean onCreate() {
```

```
        helper = new DbHelper(getContext());
        return true;
    }

    @Nullable
    @Override
    public Cursor query(@NonNull Uri uri, @Nullable String[] strings, @Nullable String s, @Nul
lable String[] strings1, @Nullable String s1) {
        Cursor c = helper.retriveMydata();
        return c;
    }

    @Nullable
    @Override
    public String getType(@NonNull Uri uri) {
        return null;
    }

    @Nullable
    @Override
    public Uri insert(@NonNull Uri uri, @Nullable ContentValues contentValues) {
        return null;
    }

    @Override
    public int delete(@NonNull Uri uri, @Nullable String s, @Nullable String[] strings) {
        return 0;
    }

    @Override
    public int update(@NonNull Uri uri, @Nullable ContentValues contentValues, @Nullable Stri
ng s, @Nullable String[] strings) {
        return 0;
    }
}
```

Now you can observe the AndroiManifest.xml file with in that tag was added.

```
<provider
    android:authorities="com.example.sqlitedbexample.MYDATA"
    android:name=".MyApplication"
    android:enabled="true"
    android:exported="true"
    ></provider>
```

In the above code we have authorities by using that we can access the SqliteDbExample database to the Another Application.

- Lets create a one new project called ContentResolverTest.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Get Data"
        android:onClick="getData"
        />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/data"
    android:textSize="25sp"
    />


</LinearLayout>
```

MainActivity.Java

```
package com.example.contentresolvertest;

import androidx.appcompat.app.AppCompatActivity;

import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView tv;
    Uri uri;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = findViewById(R.id.data);
        uri = Uri.parse("content://com.example.sqlitedbexample.MYDATA");
    }

    public void getData(View view) {
      Cursor c = getContentResolver().query(uri,null,null,null,null);
```

```
        StringBuilder builder = new StringBuilder();
         while (c.moveToNext()){
            builder.append(c.getInt(0)+" "+c.getString(1)+" "+c.getString(2)+"\n");
         }
         tv.setText(builder.toString());
    }
}
```

OutPut