SQLite

Himadri Parikh

Contents

- SQLiteOpenHelper class
- SQLiteDatabase class
- Creating a new project
- Java class to handle operations
- Activities for update, delete, Register and Login
- Defining actions to the respective activities

SQLiteOpenHelper class

- A helper class to manage database creation and version management.
- You create a subclass implementing onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) and optionally onOpen(SQLiteDatabase),
  - and this class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary.
- Transactions are used to make sure the database is always in a sensible state.
- This class makes it easy for ContentProvider implementations to defer opening and upgrading the database until first use,
  - to avoid blocking application startup with long-running database upgrades.

SQLiteDatabase class

- Exposes methods to manage a SQLite database.
- SQLiteDatabase has methods to create, delete, execute SQL commands,
    - and perform other common database management tasks.
- Database names must be unique within an application, not across all applications.

Creating a new Project

- Create a new project as SQLiteDemo.
- This project shall be able to register a user, allow login access, allow update and delete the particular user.

Java class to handle operations

Let us add two java classes to handle operations

- Name the first java class ‘TableData’ and code it as below:

```
public class TableData {
    public  static  abstract class TableInfo implements BaseColumns{
        public static final String DATABASE_NAME = "user_info";
        public static final String TABLE_NAME = "reg_info";
        public static final String USERNAME = "user_name";
        public static final String PASSWORD = "user_pass";
    }
}
```

DatabseOperations.java

- Name the second java class ‘DatabseOperations’ and code it as below:

```
public class DatabaseOperations extends SQLiteOpenHelper{ }
```

- Extend it with SQLiteOpenHelper, it would prompt you to implement methods, and add the constructor to it.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {    }

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

- Add database_version as 1:

```
public static final int db_version = 1;
```

- Add a create table query as below:

```
public String CREATE_QUERY = "Create table "+ TableData.TableInfo
.TABLE_NAME +"("+ TableData.TableInfo.USERNAME+" TEXT, "+
TableData .TableInfo.PASSWORD+" TEXT);";
```

- Edit the constructor as shown below:

```
public DatabaseOperations(Context context) {
super(context, TableData.TableInfo.DATABASE_NAME, null, db_version);
    Log.i("Database_operations", "Database Created");
}
```

- Edit the onCreate() method as shown below:

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(CREATE_QUERY);
        Log.i("Database_operations", "Table created");
}
```

- Method to get information from database (Select query)

```
public Cursor getInformation(DatabaseOperations dop) {
        SQLiteDatabase SQ = dop.getReadableDatabase();
        String[] columns =
        {TableData.TableInfo.USERNAME,  TableData.TableInfo.PASSWORD};
        Cursor CR = SQ.query(TableData.TableInfo.TABLE_NAME, columns,
                null, null,null,null, null );
        return CR;
    }
```

Add method to execute insert query

```
public void putInformation(DatabaseOperations dop, String name, String pass){
        SQLiteDatabase SQ = dop.getWritableDatabase();
        ContentValues cv = new ContentValues();

        cv.put(TableData.TableInfo.USERNAME, name);
        cv.put(TableData.TableInfo.PASSWORD, pass);

        long k = SQ.insert(TableData.TableInfo.TABLE_NAME, null, cv);
        Log.i("Database_operations", "1 row inserted");

    }
```

Method to retreive password, uname is given

```
public Cursor getUserPass(DatabaseOperations dop, String user){
        SQLiteDatabase SQ = dop.getReadableDatabase();
        String selection = TableData.TableInfo.USERNAME+ " LIKE ? ";
        String columns[] = {TableData.TableInfo.PASSWORD};
        String arg[] = {user};

        Cursor CR  = SQ.query(TableData.TableInfo.TABLE_NAME, columns,
                selection, arg, null, null, null);

        return CR;
    }
```

Method to Delete the given user

```
public void deleteUser(DatabaseOperations dop, String user, String pass){
        SQLiteDatabase SQ = dop.getWritableDatabase();
        String arg[] = {user, pass};

        String selection = TableData.TableInfo.USERNAME+
                " LIKE ? AND "+ TableData.TableInfo.PASSWORD+" LIKE ?";

        SQ.delete(TableData.TableInfo.TABLE_NAME, selection, arg);
    }
```

Method to update username

```
public void updateUser(DatabaseOperations dop, String user, String pass,
                        String new_user){
       SQLiteDatabase SQ = dop.getWritableDatabase();
       String selection = TableData.TableInfo.USERNAME+
              " LIKE ? AND "+ TableData.TableInfo.PASSWORD+" LIKE ?";
       String arg[] = {user, pass};

       ContentValues cv = new ContentValues();
       cv.put(TableData.TableInfo.USERNAME, new_user);

       SQ.update(TableData.TableInfo.TABLE_NAME, cv, selection, arg);

   }
```

Activities for update, delete, Register and Login

Changes to be made in MainActivity

- Make objects for buttons:

```
Button btnLogin, btnReg, btnUpdate, btnDelete; //outside onCreate()
int status = 0;
```

- Bind view with local objects:

```
btnLogin = findViewById(R.id.btnLoginAct);
btnReg = findViewById(R.id.btnRegisterAct);
```

- All the coding in MainActivity is to be done in onCreate() method.

Code to be executed on Login button click in MainActivity:

```
btnLogin.setOnClickListener(new View.OnClickListener() {
            Toast.makeText(this, "Please wait for a while...", Toast.LENGTH_SHORT)
.show();
        user_name = edtUser.getText().toString();
        user_pass = edtPass.getText().toString();

        DatabaseOperations dop = new DatabaseOperations(LoginAcitivity.this);
        Cursor CR = dop.getInformation(dop);

        CR.moveToFirst();
        String NAME = "";
do{
            if(user_name.equals(CR.getString(0))
&&user_pass.equals(CR.getString(1))){
                login_status = true;
                NAME = CR.getString(0);
            }
        }while (CR.moveToNext());
        startActivity(new Intent(MainActivity.this, NextActivity.class));
    }
```

Code to be executed on Register button click:

```
btnReg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent(MainActivity.this, RegisterActivity.class));
            }
        });
```

Defining actions to the respective activities

Changes to be made in NextActivity

- Local variables:

```
btnUpdate = findViewById(R.id.btnUpdateAct);
btnDelete = findViewById(R.id.btnDeleteAct);
```

Code to be executed on Delete button click:

```
btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(NextActivity.this, DeleteAcitivity.class);
                startActivity(i);
            }
        });
```

Code to be executed on Update button click:

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(NextActivity.this, UpdateAcitivity.class);
                startActivity(i);
            }
        });
```

In UpdateActivity

- Make the declarations: //out of onCreate()

```
String user_name, user_pass, new_user_pass;
EditText edtNewUser;
Button btnUpdate;
DatabaseOperations dop;
```

- Bind views:

```
edtNewUser = findViewById(R.id.edtUnameUpdate);
btnUpdate = findViewById(R.id.btnUpdate);
```

Make the onClick listener of Update button:

```
btnUpdate.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new_user_pass = edtNewUser.getText().toString();
            dop = new DatabaseOperations(UpdateActivity.this);
            dop.updateUser(dop, user_name, user_pass, new_user_pass);

            Toast.makeText(UpdateActivity.this,
                    "1 row successfully updated...", Toast.LENGTH_SHORT).show();

            finish();
        }
    });
```

In DeleteActivity

- Make declarations:

```
Bundle bn;
String user_name, user_pass;
Button btnDelete;
EditText edtPass;
DatabaseOperations dop;
```

- Bind Views:

```
btnDelete = findViewById(R.id.btnDelete);
edtPass = findViewById(R.id.edtPassDelete);
```

onClick listener of delete button:

```
btnDelete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            user_pass = edtPass.getText().toString();
            dop = new DatabaseOperations(DeleteActivity.this);
            Cursor CR = dop.getUserPass(dop, user_name);
            boolean login_status = false;
            CR.moveToFirst();
            do{
                if(user_pass.equals(CR.getString(0))){
                    login_status = true;
                }
            }while (CR.moveToNext());
if (login_status) {
                dop.deleteUser(dop, user_name, user_pass);
                Toast.makeText(DeleteActivity.this,user_name+" removed
successfully..",Toast.LENGTH_SHORT).show();
                finish();
            }else {
                Toast.makeText(DeleteActivity.this,
                        "Invalid user...Try again!", Toast.LENGTH_SHORT).show();
            }
        }
    });
}
}
```

In Registration Activity

- Make th edeclarations: //out of onCreate()

```
EditText edtUname, edtPass, edtConPass;
Button btnReg;
String user_name, user_pass, con_pass;
```

- Bind views:

```
edtUname = findViewById(R.id.edtUnameReg);
edtPass = findViewById(R.id.edtPassReg);
edtConPass = findViewById(R.id.edtCPassReg);
btnReg = findViewById(R.id.btnReg);
```

- onClick event of Registration Button:

```
btnReg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        user_name = edtUname.getText().toString();
        user_pass = edtPass.getText().toString();
        con_pass = edtConPass.getText().toString();

        if(!user_pass.equals(con_pass)){
            Toast.makeText(RegisterActivity.this,
                    "Password doesn't match...", Toast.LENGTH_SHORT).show();
            edtPass.setText("");
            edtConPass.setText("");
        }
else {
                    DatabaseOperations dop = new
DatabaseOperations(RegisterActivity.this);
                    dop.putInformation(dop, user_name, user_pass);

                    Toast.makeText(RegisterActivity.this, "Registered Successfully...",
                            Toast.LENGTH_SHORT).show();
                    finish();
                }
            }
        });
    }
}
```

MUMBAI CHAPTER

Himadri Vrajesh Parikh

himadrichaudhary1314@gmail.com

https://www.linkedin.com/in/himadri-parikh-506219109/