Mobile App Development

Lec 7: Saving data (Database)

Ekarat Rattagan, PhD

Outline

- Saving Key-Value Sets
- Saving Files
- Saving Data in SQL Databases

SQL Databases

- Ideal for repeating or structured data, such as contact information.

- SQLite
  - A relational DBMS contained in a C programming library.
  - Embedded into the end program.
  - Android.database.sqlite

DBMS

- Schema: a formal declaration of how the DB is organized, such as table, relation, etc.

Table name: Contacts

Field	Type	Key
id	INT	PRIMARY
name	TEXT	
phone_number	TEXT	

Example 1 (Contact DB).

- Writing Contact.java Class

```java
public class Contact
{
    public int _id;
    public String _name;
    public String _phone_number;

    public Contact(){}

    public Contact(String name, String _phone_number)
    {
        this._name = name;
        this._phone_number = _phone_number;
    }

    public Contact(int id, String name, String _phone_number)
    {
        this._id = id;
        this._name = name;
        this._phone_number = _phone_number;
    }
}
```

Example 1.

- Use SQLiteOpenHelper class to handle the DB operations

```java
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.provider.ContactsContract;
public class DatabaseHandler extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "contactsManager";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_CONTACTS = "contacts";
    // Contacts Table Columns names
    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_PH_NO = "phone_number";
    private static final String KEY_LINE_ID = "line_id";
    SQLiteDatabase db;

    public DatabaseHandler(Context context){
        super(context,DATABASE_NAME, null, DATABASE_VERSION);
        db = getWritableDatabase();
    }
```

Example 1.

- Use SQLiteOpenHelper class to handle the DB operations

```java
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS + "("
            + KEY_ID + " INTEGER PRIMARY KEY,"
            + KEY_NAME + " TEXT NOT NULL UNIQUE,"
            + KEY_PH_NO + " TEXT NOT NULL UNIQUE" + ")";
    db.execSQL(CREATE_CONTACTS_TABLE);
}


private static final String DATABASE_ALTER_CONTACT_1 = "ALTER TABLE "
    + TABLE_CONTACTS + " ADD COLUMN " + KEY_LINE_ID + " string;";

// Upgrading database
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    if (oldVersion < 2) {
        db.execSQL(DATABASE_ALTER_CONTACT_1);
    }
}
}
```

CRUD operations

- Use SQLiteOpenHelper class to handle the DB operations

```java
// Adding new contact
public void addContact(Contact contact) {}

// Getting single contact
public Contact getContact(int id) {}

// Getting All Contacts
public List<Contact> getAllContacts() {}

// Updating single contact
public int updateContact(Contact contact) {}

// Deleting single contact
public void deleteContact(Contact contact) {}
```

CRUD (Insert)

- Long insert (String table, String nullColumnHack, ContentValues values)

```java
// Adding new contact
public void addContact(Contact contact)
{

    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact._name);
    values.put(KEY_PH_NO,contact._phone_number);

    long insert = db.insert(TABLE_CONTACTS, null, values);

    db.close();

}
```

CRUD (Select)

- Select (query)

```java
// Getting single contact
public Contact getContact(int id) {

    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = db.query(
        TABLE_CONTACTS
            , new String[]{KEY_ID, KEY_NAME, KEY_PH_NO}
            , KEY_ID + "=?"
            , new String[]{String.valueOf(id)}
            , null
            , null
            , null
            , null);

    if (cursor != null)
        cursor.moveToFirst();

    Contact contact = new Contact(Integer.parseInt(cursor.getString(0)),
            cursor.getString(1), cursor.getString(2));
    // return contact
    return contact;
}
```

```java
Cursor query (

    String table,
    String [] columns,
    String selection,
    String [] selectionArgs,
    String groupBy,
    String having,
    String orderBy,
    String limit

)
```

CRUD (Select)

Parameters	
table	String: The table name to compile the query against.
columns	String: A list of which columns to return. Passing null will return all columns, which is discouraged to prevent reading data from storage that isn't going to be used.
selection	String: A filter declaring which rows to return, formatted as an SQL WHERE clause (excluding the WHERE itself). Passing null will return all rows for the given table.
selectionArgs	String: You may include ?s in selection, which will be replaced by the values from selectionArgs, in order that they appear in the selection. The values will be bound as Strings.
groupBy	String: A filter declaring how to group rows, formatted as an SQL GROUP BY clause (excluding the GROUP BY itself). Passing null will cause the rows to not be grouped.
having	String: A filter declare which row groups to include in the cursor, if row grouping is being used, formatted as an SQL HAVING clause (excluding the HAVING itself). Passing null will cause all row groups to be included, and is required when row grouping is not being used.
orderBy	String: How to order the rows, formatted as an SQL ORDER BY clause (excluding the ORDER BY itself). Passing null will use the default sort order, which may be unordered.
limit	String: Limits the number of rows returned by the query, formatted as LIMIT clause. Passing null denotes no LIMIT clause.

Returns	
Cursor	A Cursor object, which is positioned before the first entry. Note that Cursors are not synchronized, see the documentation for more details.

CRUD (Select)

```java
public List<Contact> getAllContacts()
{
    List<Contact> contactList = new ArrayList<Contact>();
    String selectQuery = "SELECT  * FROM " + TABLE_CONTACTS;

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        do {
            Contact contact = new Contact();
            contact._id = Integer.parseInt(cursor.getString(0));
            contact._name = cursor.getString(1);
            contact._phone_number = cursor.getString(2);
             contactList.add(contact);
        } while (cursor.moveToNext());
    }

    // return contact list
    return contactList;
}
```

CRUD (Update)

```java
// Updating single contact
public int updateContact(Contact contact)
{
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_NAME, contact._name);
    values.put(KEY_PH_NO, contact._phone_number);

    // updating row
    return db.update(
      TABLE_CONTACTS,
      values,
      KEY_ID + " = ?",
        new String[] { String.valueOf(contact._id) }
      );
}
```

CRUD (Delete)

```java
// Deleting single contact
public void deleteContact(Contact contact)
{
    SQLiteDatabase db = this.getWritableDatabase();

    db.delete(
        TABLE_CONTACTS,
        KEY_ID + " = ?",
            new String[] { String.valueOf(contact._id) }
            );

    db.close();
}
```

Example: MainActivity.java

```java
@Override
protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    DatabaseHandler db = new DatabaseHandler(this);

    db.addContact(new Contact("Ekarat","0899999999"));

    List<Contact> contacts = db.getAllContacts();

}
```
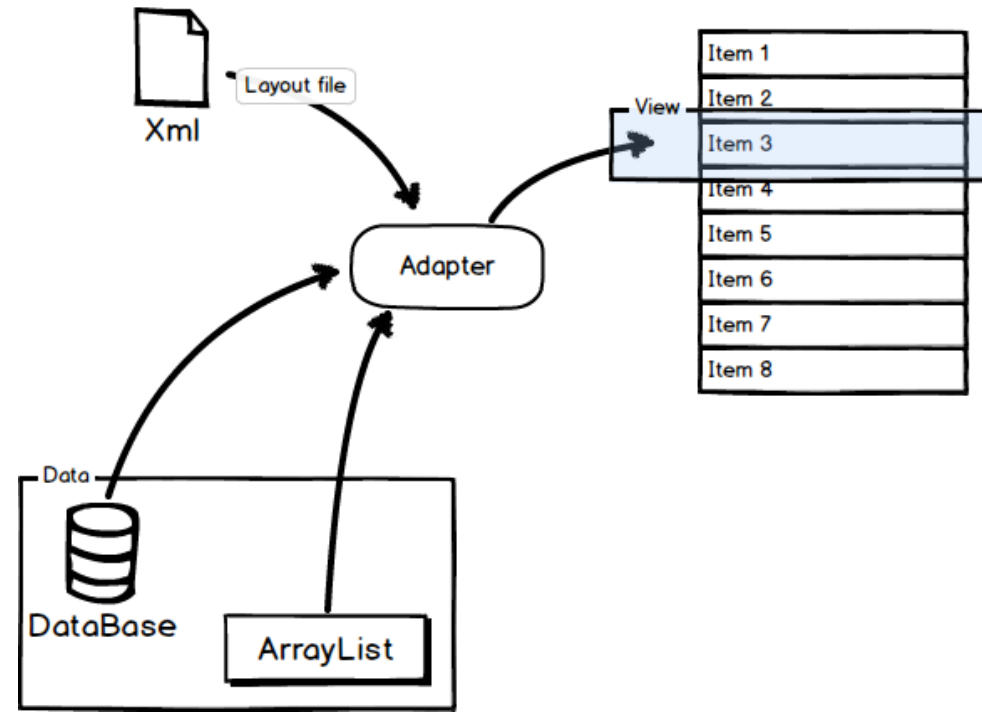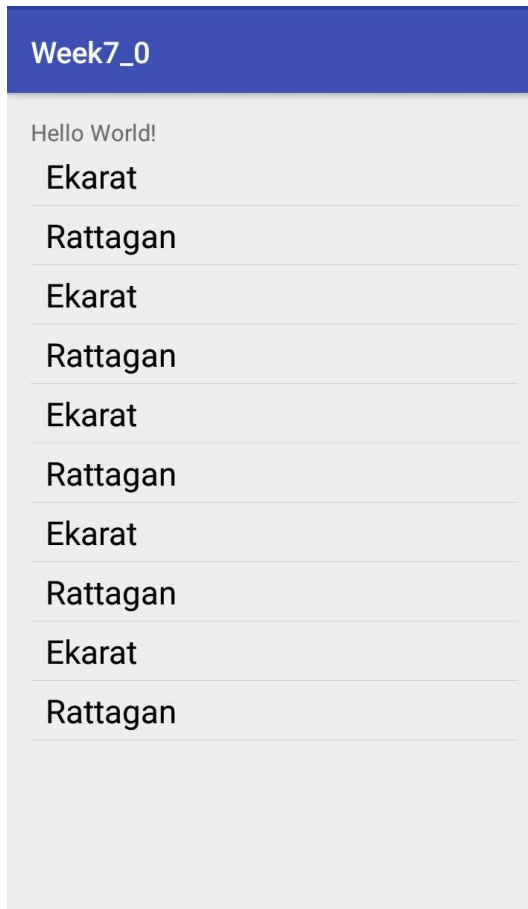
15

Exercise 1

1.1 Implement the Contact app and use debug to check the result.

1.2 Update the name and phone of the 2nd row.

1.3 Delete the first row.

1.4 Add a new column "salary".

ListView (Obsolete) Replaced by RecyclerView (Tutorial link)

Show the results of DB in ListView

http://neohsu.github.io/images/SVG/0-talk-about-listview-and-adapter-on-android.png

17

ListView

1. Create a ListView in activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<ActivityLayout xmlns:android=http://schemas.android.com/apk/res/android
    …… >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:id="@+id/textView" />

    <ListView
        android:layout_marginTop="20dp"
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true" />

</ActivityLayout>
```

ListView

2. Create a CustomAdapter class (BaseAdapter)

```java
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

public class CustomAdapter extends BaseAdapter {

    Context mContext;
    String[] strName;

    public CustomAdapter(Context context, String[] strName){
        mContext = context;
        this.strName = strName;

    }
```

ListView

2. Create a CustomAdapter class

```java
@Override
public int getCount() {
    return strName.length;
}

@Override
public Object getItem(int i) {
    return null;
}

@Override
public long getItemId(int i) {
    return 0;
}
```

ListView

2. Create a CustomAdapter class

```java
@Override
public View getView(int position, View view, ViewGroup parent)
{
    LayoutInflater mInflater = (LayoutInflater)mContext.getSystemService
    (Context.LAYOUT_INFLATER_SERVICE);

    if(view == null)
        view = mInflater.inflate(R.layout.listview_row, parent, false);

    TextView textView = (TextView)view.findViewById(R.id.textView1);
    textView.setText(strName[position]);

    return view;
}
}
```

--

ListView

3. Inside MainActivity.java

```java
db.addContact(new Contact("Ekarat","0899999999"));
db.addContact(new Contact("Rattagan","011111111"));

List<Contact> contacts = db.getAllContacts();

String[] datas = new String[contacts.size()];
for(int i=0; i<datas.length; i++)
{
    datas[i]= contacts.get(i)._name;
}

CustomAdapter adapter = new CustomAdapter(getApplicationContext(),datas);
ListView listView = (ListView)findViewById(R.id.listView1);
listView.setAdapter(adapter);
```

Resources

http://www.sqlitetutorial.net/
https://developer.android.com/training/data-storage/room/index.html