2016

Session 5 - Exercises

V1.0

JUAN RONDON

Table of Contents

Android Note taking application

Database Implementation (ORM with Realm)

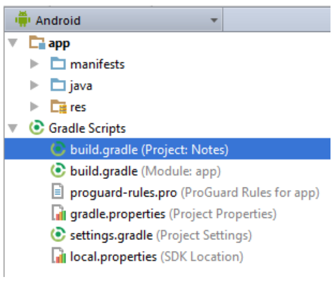
In this lab, you will be adding the code for deleting a note and the use of a persistent database using REALM ORM so every time the application is closed your existing notes won't be destroyed.

1. Install Realm into your android app.

   Installation

   Realm is installed as a Gradle plugin.
   Installing Realm as a Gradle plugin is a two-step process.

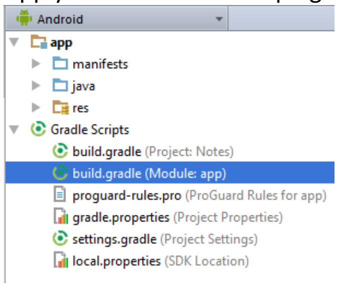
   a. Add the following class path dependency to the **project level build.gradle file**.


Android
app
manifests
java
res
Gradle Scripts
build.gradle (Project: Notes)
build.gradle (Module: app)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.2'
        classpath "io.realm:realm-gradle-plugin:1.2.0"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

   b. Apply the realm-android plugin to the **top of application level build.gradle file**.


Android
app
manifests
java
res
Gradle Scripts
build.gradle (Project: Notes)
build.gradle (Module: app)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

```
apply plugin: 'com.android.application'
apply plugin: 'realm-android'

android {
    compileSdkVersion 23
    buildToolsVersion "24.0.1"

    defaultConfig {
        applicationId "androidcourse.notes"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
```

Once these two changes are made, simply sync your gradle dependencies.

2. Open **Note.java** file and make it a subclass of **RealmObject** class so a database table will be automatically created for our notes.
    - Next you will be adding some annotations **(@PrimaryKey, @Required**) to some of the properties of the class.
    - Next you will be removing the static property that was used to create the id for the note.
    - Create an empty Note constructor (Required by Realm)
    - Finally create a set method for the id.

Completed Note class:

```
package androidcourse.notes.Models;

import java.text.SimpleDateFormat;
import java.util.Date;

import io.realm.RealmObject;
import io.realm.annotations.PrimaryKey;
import io.realm.annotations.Required;

/**
 * Created by Juan on 19/08/2016.
 */
public class Note extends RealmObject {

    @PrimaryKey
    private int id;
    @Required
    private String title;
    @Required
    private String content;
    @Required
    private Date lastModified;
    private String password;

    public Note(String title, String content) {
        this.title = title;
        this.content = content;
        lastModified = new Date();
    }

    public Note(String title, String content, String password) {
        this.title = title;
        this.content = content;
        this.password = password;
        lastModified = new Date();
    }

    public Note() {
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getContent() {
        return content;
    }

    public Date getLastModified() {
        return lastModified;
    }

    public String getPassword() {
        return password;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setContent(String content) {
        this.content = content;
    }
    public void setLastModified(Date lastModified) {
        this.lastModified = lastModified;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String dateFormatted() {
        SimpleDateFormat sdf = new SimpleDateFormat("MMM dd - HH:mm");
        return "Last edited on: " + sdf.format(lastModified);
    }
}
```

Note that password property is **not** set as required otherwise Realm won't allow us to create a note without password.

3. You will need to modify **NotesAdapter** class in order to work with a **List<Note>** objects instead of **ArrayList<Note>**. When querying Realm for all the rows, it returns a list of **RealmResults;** this collection is not compatible with **ArrayList**.

```
public class NotesAdapter extends ArrayAdapter<Note> {

    private Context mContext;
    private List<Note> mNotelist;
    private int mLayoutResourceId;

    private static class ViewHolder {
        TextView title;
        TextView date;
        ImageView img;
        ImageView pwd;
    }

    public NotesAdapter(Context context, int layoutResourceId, List<Note> notelist) {
        super(context, layoutResourceId, notelist);
        mContext = context;
        mNotelist = notelist;
        mLayoutResourceId = layoutResourceId;
    }
```

4. Once the adapter is fixed, open **NotesList** java file. You will need to modify some of the code from this file.
    - Start by removing both **request codes** at the top. For this application, when you use a database, you no longer require to start an activity for result, as soon as you create, edit or remove a note from the database the changes will be reflected in **NotesList**.
    - Remove the **ArrayList** of Notes. You will replace the Array List with a method that will return a list of all the notes from the database. (***getNotesList()***).
    - Remove the method **updateNotesList**
    - Remove the method **onActivityResult**

    Note: At this point some errors will appear, just ignore them, we will be fixing them soon.

5. Modify **EditNoteIntent** so it will look like the following code:

```
private void editNoteIntent(Note note) {
    Intent editNoteIntent = new Intent(NotesList.this, EditNote.class);
    editNoteIntent.putExtra("note to edit", note.getId());
    startActivity(editNoteIntent);
}
```

6. You don't need to send the entire note to **EditNote** activity, this time you'll send only the id and then you can load that specific note by id from the database.

7. Realm needs to be **configured** only once in the application we do it in the launcher activity (**NotesList**) Type the following code inside **onCreate** method before **notesList listener**.

```
RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(this).build();
Realm.setDefaultConfiguration(realmConfiguration);
_context = Realm.getDefaultInstance();
```

8. Create **getNotesList()** method:

```
private List<Note> getNotesList() {
    return _context.where(Note.class).findAll().sort("title");
}
```

After creating the method, replace all the ocurrences of **notes** with **getNotesList()**

9. At this point you are almost done with **NotesList** file; modify **notesListView** listener so it will look like this code.

```
notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        //get the selected note from the list
        Note note = adapter.getItem(position);
        //if note is password protected
        if (note.getPassword() != null) {
            displayPinPrompt(note);
        } else {
            editNoteIntent(note);
        }
    }
});
```

10. Refresh the adapter every time **NotesList** is loaded. For this you can override onStart method.

```
@Override
protected void onStart() {
    super.onStart();
    adapter.notifyDataSetChanged();
}
```

11. Modify the listener for addNote so the intent will be just start activity and not start activity for result.

```
//listener for add NoteImage
ImageView addNoteImg = (ImageView) findViewById(R.id.addNoteImg);
addNoteImg.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //create an intent
        Intent addNoteIntent = new Intent(getBaseContext(), AddNote.class);
        startActivity(addNoteIntent);
    }
});
```

12. Finally override onDestroy method:

```
@Override
protected void onDestroy() {
    super.onDestroy();
    _context.close();
}
```

Close realm database connection

13. Completed **NotesList** java file (import statements omitted)

```
package androidcourse.notes;

public class NotesList extends AppCompatActivity {

    private NotesAdapter adapter;
    private Realm _context;

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

        final ListView notesListView = (ListView) findViewById(R.id.listView);
        adapter = new NotesAdapter(this, R.layout.note_row, getNotesList());
        notesListView.setAdapter(adapter);

        RealmConfiguration realmConfiguration = new RealmConfiguration.Builder(this).build();
        Realm.setDefaultConfiguration(realmConfiguration);
        _context = Realm.getDefaultInstance();


        notesListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                //get the selected note from the list
                Note note = adapter.getItem(position);
                //if note is password protected
                if (note.getPassword() != null) {
                    displayPinPrompt(note);
                } else {
                    editNoteIntent(note);
                }
            }
        });

        //listener for add NoteImage
        ImageView addNoteImg = (ImageView) findViewById(R.id.addNoteImg);
        addNoteImg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //create an intent
                Intent addNoteIntent = new Intent(getBaseContext(), AddNote.class);
                startActivity(addNoteIntent);
            }
        });
    }

    private List<Note> getNotesList() {
        return _context.where(Note.class).findAll().sort("title");
    }


    private void editNoteIntent(Note note) {
        Intent editNoteIntent = new Intent(NotesList.this, EditNote.class);
        editNoteIntent.putExtra("note to edit", note.getId());
        startActivity(editNoteIntent);
    }


    private void displayPinPrompt(final Note note) {
        View layout = getLayoutInflater().inflate(R.layout.pin_prompt_layout, null);
        final EditText password1 = (EditText) layout.findViewById(R.id.pwd1);
        final TextView error = (TextView) layout.findViewById(R.id.TextView_PwdProblem);

        password1.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
            }

            public void afterTextChanged(Editable s) {
                String strPass1 = password1.getText().toString();
                //validate if password is correct
                if (!strPass1.equals(note.getPassword())) {
                    error.setText("Invalid Password");
                    error.setTextColor(Color.RED);
                } else {
                    error.setText("Valid Password");
                    error.setTextColor(Color.GREEN);
                }
            }
        });

        AlertDialog.Builder builder = new AlertDialog.Builder(NotesList.this);
        builder.setView(layout);
        builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int whichButton) {
                dialog.cancel();
            }
        });
        builder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                String strPassword1 = password1.getText().toString();
                if (strPassword1.equals(note.getPassword())) {
                    editNoteIntent(note);
                }
            }
        });
        AlertDialog passwordDialog = builder.create();
        passwordDialog.show();
    }
    @Override
    protected void onStart() {
        super.onStart();
        adapter.notifyDataSetChanged();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        _context.close();
    }
}
```

14. Open **addNote** java file. You will be adding the code required to save the notes in the database.

- Add an instance of Realm context.

```
public class AddNote extends AppCompatActivity {

    private String mPassword;
    private Realm _context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_note);
        _context = Realm.getDefaultInstance();
        final CheckBox password = (CheckBox) findViewById(R.id.pwdCheckBox);
```

- Create a method that will be used to save a note.

```
private void saveNote(Note note) {
    //set the id for the note
    note.setId(getNextNoteId());
    // persist your data
    _context.beginTransaction();
    _context.copyToRealm(note);
    _context.commitTransaction();
}

//used to generate the next note id
private int getNextNoteId() {
    int id = 1;
    if (_context.where(Note.class).findAll().size() > 0)
        id = _context.where(Note.class).max("id").intValue() + 1;
    return id;
}
```

15. Modify **onOptionsItemSelected** menu method in order to remove the intent and just call saveNote method.

```
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.SaveNote) {
        String title = ((EditText) findViewById(R.id.title_add)).getText().toString();
        String contents = ((EditText) findViewById(R.id.title_add)).getText().toString();
        Note note;
        //check if the note has password set.
        if (mPassword == null) {
            note = new Note(title, contents);
        } else {
            note = new Note(title, contents, mPassword);
        }
        saveNote(note);
        finish();
    }
    return true;
}
```

16. Override **onDestroy** method to this activity so the realm connection is closed.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    _context.close();
}
```

17. Completed code for **AddNote** java file (Imports removed):

```
package androidcourse.notes;
public class AddNote extends AppCompatActivity {

    private String mPassword;
    private Realm _context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_note);
        _context = Realm.getDefaultInstance();
        final CheckBox password = (CheckBox) findViewById(R.id.pwdCheckBox);
        assert password != null;
        password.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                View layout = getLayoutInflater().inflate(R.layout.pin_layout, null);
                final EditText password1 = (EditText) layout.findViewById(R.id.pwd1);
                final EditText password2 = (EditText) layout.findViewById(R.id.pwd2);
                final TextView error = (TextView) layout.findViewById(R.id.TextView_PwdProblem);
                //Listener for the password checkbox
                password2.addTextChangedListener(new TextWatcher() {
                    @Override
                    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                    }

                    @Override
                    public void onTextChanged(CharSequence s, int start, int before, int count) {
                    }

                    public void afterTextChanged(Editable s) {
                        String strPass1 = password1.getText().toString();
                        String strPass2 = password2.getText().toString();
                        //validate if both passwords are the same
                        if (strPass1.equals(strPass2) && strPass2.length() == 4) {
                            error.setText("Passwords Match");
                            error.setTextColor(Color.GREEN);
                        } else if (strPass2.length() != 4) {
                            error.setText("Password must contain 4 digits");
                            error.setTextColor(Color.RED);
                        } else {
                            error.setText("Passwords do not Match");
                            error.setTextColor(Color.RED);
                        }
                    }
                });
                AlertDialog.Builder builder = new AlertDialog.Builder(AddNote.this);
                builder.setView(layout);
                builder.setNegativeButton(android.R.string.cancel, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {
                        //if password prompt is cancelled disable checkbox
                        password.setChecked(false);
                        mPassword = null;
                    }
                });
                builder.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        String strPassword1 = password1.getText().toString();
                        String strPassword2 = password2.getText().toString();
                        if (strPassword1.equals(strPassword2)) {
                            mPassword = strPassword1;
                        }
                    }
                });
                AlertDialog passwordDialog = builder.create();
                passwordDialog.show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.add_note_menu, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.SaveNote) {
            String title = ((EditText) findViewById(R.id.title_add)).getText().toString();
            String contents = ((EditText) findViewById(R.id.title_add)).getText().toString();
            Note note;
            //check if the note has password set.
```

```
                if (mPassword == null) {
                    note = new Note(title, contents);
                } else {
                    note = new Note(title, contents, mPassword);
                }
                saveNote(note);
                finish();
            }
            return true;
        }

        private void saveNote(Note note) {
            //set the id for the note
            note.setId(getNextNoteId());
            // persist your data
            _context.beginTransaction();
            _context.copyToRealm(note);
            _context.commitTransaction();
        }

        //used to generate the next note id
        private int getNextNoteId() {
            int id = 1;
            if (_context.where(Note.class).findAll().size() > 0)
                id = _context.where(Note.class).max("id").intValue() + 1;
            return id;
        }

        @Override
        protected void onDestroy() {
            super.onDestroy();
            _context.close();
        }
    }
```

18. Open **EditNote** java file. You will be adding the code required to update the notes in the database and delete notes from the database.

19. Add an instance to Realm (_context) like you did in AddNote activity.

20. The next step would be to create a method that will be used to search for a note by its id.

```
private Note findNote(int id) {
    return _context.where(Note.class).equalTo("id", id).findFirst();
}
```

21. Get the note from the database by using the **id** passed from **NotesList** activity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_edit_note);
    _context = Realm.getDefaultInstance();
    //obtaining the note from main activity
    Intent intent = getIntent();
    if (intent.hasExtra("note to edit")) {
        int id = intent.getIntExtra("note to edit", 0);
        noteToEdit = findNote(id);
    }
    //updating the UI with the note info
    EditText title = (EditText) findViewById(R.id.title_edit);
    title.setText(noteToEdit.getTitle());
    EditText content = (EditText) findViewById(R.id.note_info_edit);
    content.setText(noteToEdit.getContent());
}
```

As seen from the above code, instead of receiving a Note object, the intent carries the id of the note that you want to edit.
Your app uses **findNote** method to retrieve the note from the database and then it populates the title and content widgets.

22. Create the CRUD functionality for Update and Delete a note.

```
private void updateNote(String title, String content) {
    _context.beginTransaction();
    noteToEdit.setTitle(title);
    noteToEdit.setContent(content);
    noteToEdit.setLastModified(new Date());
    _context.commitTransaction();
}

private void deleteNote() {
    _context.beginTransaction();
    noteToEdit.deleteFromRealm();
    _context.commitTransaction();
}
```

23. Locate **onOptionsItemSelected** method and modify the existing code in order to use both methods that you created in the previous step.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.SaveNote) {
        String title = ((EditText) findViewById(R.id.title_edit)).getText().toString();
        String content = ((EditText) findViewById(R.id.note_info_edit)).getText().toString();
        //updating the note
        updateNote(title, content);
    } else if (item.getItemId() == R.id.DeleteNote) {
        deleteNote();
    }
    //Return to notesList activity
    finish();
    return true;
}
```

24. The last step is to **override onDestroy** method in order to close the realm connection.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    realm.close();
}
```

25. Completed code for EditNote (Imports removed):

```
package androidcourse.notes;

public class EditNote extends AppCompatActivity {
    private Note noteToEdit;
    private Realm _context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_note);
        _context = Realm.getDefaultInstance();
        //obtaining the note from main activity
        Intent intent = getIntent();
        if (intent.hasExtra("note to edit")) {
            int id = intent.getIntExtra("note to edit", 0);
            noteToEdit = findNote(id);
        }
        //updating the UI with the note info
        EditText title = (EditText) findViewById(R.id.title_edit);
        title.setText(noteToEdit.getTitle());
        EditText content = (EditText) findViewById(R.id.note_info_edit);
        content.setText(noteToEdit.getContent());
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.edit_note_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.SaveNote) {
            String title = ((EditText) findViewById(R.id.title_edit)).getText().toString();
            String content = ((EditText) findViewById(R.id.note_info_edit)).getText().toString();
            //updating the note
            updateNote(title, content);
        } else if (item.getItemId() == R.id.DeleteNote) {
            deleteNote();
        }
        //Return to notesList activity
        finish();
        return true;
    }


    private Note findNote(int id) {
        return _context.where(Note.class).equalTo("id", id).findFirst();
    }

    private void updateNote(String title, String content) {
        _context.beginTransaction();
        noteToEdit.setTitle(title);
        noteToEdit.setContent(content);
        noteToEdit.setLastModified(new Date());
        _context.commitTransaction();
    }

    private void deleteNote() {
        _context.beginTransaction();
        noteToEdit.deleteFromRealm();
        _context.commitTransaction();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        _context.close();
    }
}
```

26. Test your application in the emulator/device and make sure all the functions are working properly.