App Development
in Android Studio

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Lab 3: Components

October 21, 2017

Jón Steinn Elíasson

jonsteinn@gmail.com

Contents

List of Tables

List of Figures

List of Listings

1 Activities

We have already used an activity without going into too much detail what it is. An activity is a single screen unit (usually full screen) with an user interface. So far we have only worked with a single activity but an Android app can have multiple activities. It breaks the app up into section with different purpose and UI. For example, a menu in an email app could be an activity while composing an email might be another, opened from the menu.

One activity serves as a luncher activity. It is our starting point when running the app (opposed to a C++ main function) and from there on we can start navigating to other activities if any. Every app must have a luncher activity. All activities must be declared in our app's manifest[1] and the luncher activity is also determined there.

Each activity uses a layout file that defines their UI at leat partially (some of it may be done dynamically in Java). In the `onCreate` function we have been setting the activity's layout with the `setContentView` method. Activities can share layouts although it serves a limited purpose unless most of the UI is dynamic. We will look at better ways to share UI in Fragments.

1.1 Starting a new Activity

Lets start by creating a new activity with File > New > Activity > Empty Activity and name it `SecondActivity` and leave the other options as they are. Before proceeding you should inspect your manifest. Add a button to the luncher activity which calls the method `clicked()` upon being clicked and add some text to the second activity so it differentiates from the other one. Finally add the following method to the luncher activity and run the progam.

```
1 public void clicked(View view) {
2     startActivity(new Intent(MainActivity.this, SecondActivity.class));
3 }
```

The `startActivity` method takes `Intent` as a paramter. We will look at that class, its parameters and methods in more detail later but for now, you can think of it as a bridge between two activities.

1.2 Lifecycle

Activities are managed by a stack called the activity stack (or the back stack) and whatever activity is in our foreground is at the top of our stack. During the lifecycle of an activity, it can have various states and the event that brings us to set states have their own callback methods. This lifecycle can be seen in figure 1.

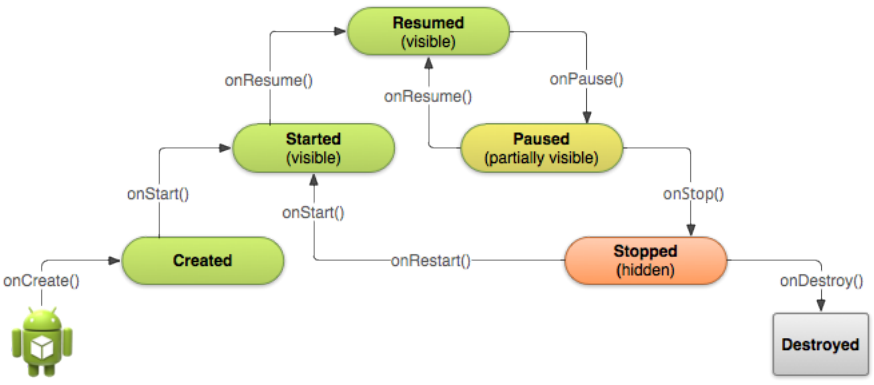

Figure 1: The lifecycle of an Activity

The states are

- **Starting**. In the process of setting up.

[1] Android Studio does this automatically when an activity is created

- **Running**. In foreground.
- **Paused**. Not in foreground.
- **Stopped**. Inactive but remains in memory.
- **Destroyed**. Shut down and removed from memory.

There are multiple callback methods called when moving between states. We have already seen `onCreate` which is the only one that activities are required to override but there are several others.

- **`onCreate()`**.
  This callback method is for the event of creating an activity and is called before the activity starts for the first time. It is typically used for initialization as we have been using it already.
- **`onStart()`**.
  Is called every time the activity starts, after entering the starting state. Here the activity becomes visible and prepares for entering the foreground and becoming interactive. This is where the app initializes the code that maintains the UI of the activity.
- **`onResume()`**.
  When entering the running state the activity is coming to the foreground and this callback is invoked. Here we should initialize components that are released by `onPause`, such as animations and camera.
- **`onPause()`**.
  This method is invoked when you are leaving your activity and it seizes to be in the foreground but is still visible. From there on it can either be resumed later or stopped. Here we must release system resources such as GPS and camera as well as animations but we should not perform long and heavy tasks of cleaning up here.
- **`onStop()`**.
  After your activity siezes to be visible, this callback is invoked. The `onPause` method is always called before this one and any large task of releasing resources should happen here. The activity is still in memory at this stage.
- **`onRestart()`**.
  If the activity goes from being stopped to starting it will invoke the `onRestart` method.
- **`onDestroy()`**.
  When you are removing your activity from memory this method is invoked but at what time exactly is unpredictable. To destroy our current activity we can call the `finish` method.

Using `Log.d` we can use printf debugging with certain tags and filter them with the Android monitor as shown in figure 2.

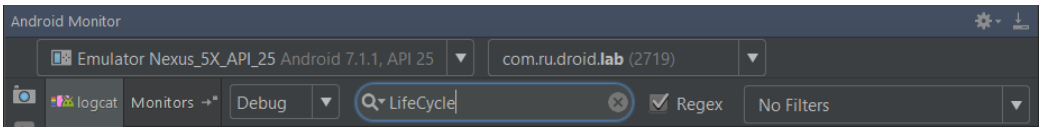

Figure 2: Filter logs

Using `Log.d` we can monitor the lifecycle of the two activities found in listings 1 and 2. You should try various scenarios of navigating between the apps as well as rotating the screen and using the Android back button.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="vertical"
```

```
    android:gravity="center"
    tools:context="com.ru.droid.lab.MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Activity 1"/> <!-- move to res -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="gotoWithoutFinish"
        android:text="goto 2"/> <!-- move to res -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="gotoAndFinish"
        android:text="goto 2 and finish"/> <!-- move to res -->
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context="com.ru.droid.lab.SecondActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Activity 2"/> <!-- move to res -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="gotoWithoutFinish"
        android:text="goto 1"/> <!-- move to res -->
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="gotoAndFinish"
        android:text="goto 1 and finish"/> <!-- move to res -->
</LinearLayout>
```

Listing 1: Two layouts for two activities

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("LifeCycle", "Activity 1: onCreate");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("LifeCycle", "Activity 1: onStart");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("LifeCycle", "Activity 1: onResume");
    }
    @Override
    protected void onPause() {
        super.onPause();
```

```
21          Log.d("LifeCycle", "Activity 1: onPause");
22      }
23      @Override
24      protected void onStop() {
25          super.onStop();
26          Log.d("LifeCycle", "Activity 1: onStop");
27      }
28      @Override
29      protected void onDestroy() {
30          super.onDestroy();
31          Log.d("LifeCycle", "Activity 1: onDestroy");
32      }
33      public void gotoWithoutFinish(View view) {
34          startActivity(new Intent(MainActivity.this, SecondActivity.class));
35      }
36      public void gotoAndFinish(View view) {
37          startActivity(new Intent(MainActivity.this, SecondActivity.class));
38          finish();
39      }
40  }
41
42  public class SecondActivity extends AppCompatActivity {
43      @Override
44      protected void onCreate(Bundle savedInstanceState) {
45          super.onCreate(savedInstanceState);
46          setContentView(R.layout.activity_second);
47          Log.d("LifeCycle", "Activity 2: onCreate");
48      }
49      @Override
50      protected void onStart() {
51          super.onStart();
52          Log.d("LifeCycle", "Activity 2: onStart");
53      }
54      @Override
55      protected void onResume() {
56          super.onResume();
57          Log.d("LifeCycle", "Activity 2: onResume");
58      }
59      @Override
60      protected void onPause() {
61          super.onPause();
62          Log.d("LifeCycle", "Activity 2: onPause");
63      }
64      @Override
65      protected void onStop() {
66          super.onStop();
67          Log.d("LifeCycle", "Activity 2: onStop");
68      }
69      @Override
70      protected void onDestroy() {
71          super.onDestroy();
72          Log.d("LifeCycle", "Activity 2: onDestroy");
73      }
74      public void gotoWithoutFinish(View view) {
75          startActivity(new Intent(SecondActivity.this, MainActivity.class));
76      }
77      public void gotoAndFinish(View view) {
78          startActivity(new Intent(SecondActivity.this, MainActivity.class));
79          finish();
80      }
81  }
```

Listing 2: Overriding on activity lifecycle callbacks for two activities