Application Building Components

Mobile Applications 60014105
(3 credits)

THEORY LECTURE

ISRA UNIVERSITY

Objectives

Application Building Blocks

What's in an Application?

Debugging in Android

Application Manifest

System level protection for Android app

Android Permissions

Activity Lifecycle

first Android app

In the previous lecture……

Application Building Blocks:

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

There are following four main components that can be used within an Android application:

Activities – visual user interface focused on a single thing a user can do

Services – no visual interface – they run in the background

Broadcast Receivers – receive and react to broadcast announcements

Content Providers – allow data exchange between applications

Application Building Blocks:

Sr.No	Components & Description

1	**Activities**
	They dictate the UI and handle the user interaction to the smart phone screen.
	`public class MainActivity extends Activity {`
	`}`

2	**Services**
	They handle background processing associated with an application.
	`public class MyService extends Service {`
	`}`

3	**Broadcast Receivers**
	They handle communication between Android OS and applications.
	`public class MyBroadcastReceiver extends BroadcastReceiver {`
	`public void onReceive() {`
	`}`

4	**Content Providers**
	They handle data and database management issues.
	`public class MyContentProvider extends ContentProvider {`
	`public void onReceive() {`
	`}`

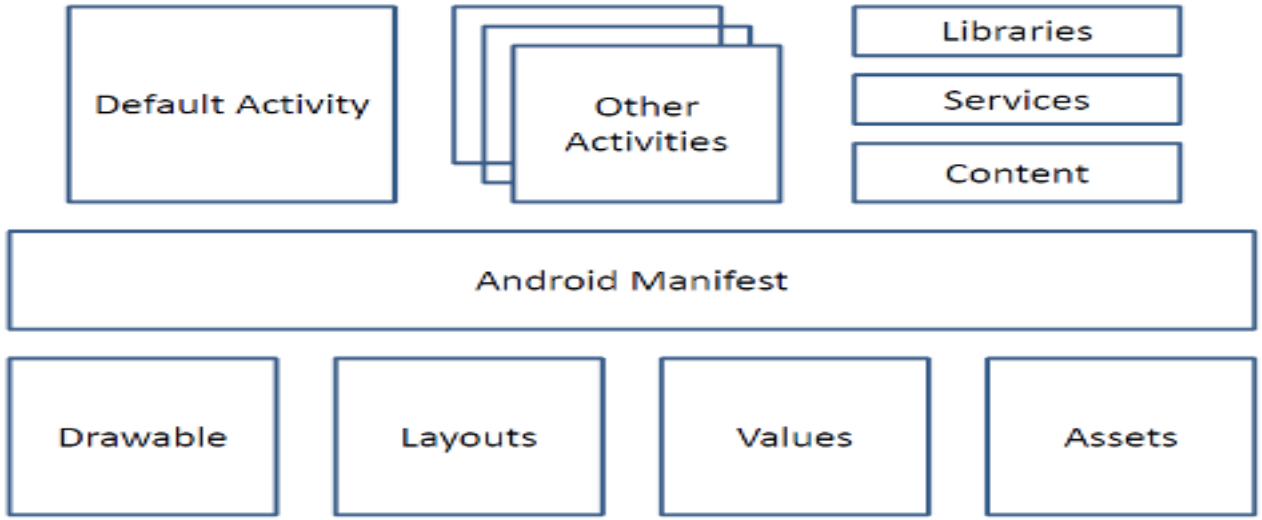
Additional Components

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them.

These components are:

S.No	Components & Description
1	**Fragments** Represents a portion of user interface in an Activity.
2	**Views** UI elements that are drawn on-screen including buttons, lists forms etc.
3	**Layouts** View hierarchies that control screen format and appearance of the views.
4	**Intents** Messages wiring components together.
5	**Resources** External elements, such as strings, constants and drawable pictures.
6	**Manifest** Configuration file for the application.

What's in an Application?



Debugging in Android

- Traditional *System.out.println* is not available in the Android system
- We can debug through the app user interface

– Errors crash and close app

- Instead use Logging mechanisms

– ***Log.v(String tag, String message);***

– *tag -> app name*

– *Message -> debug message.*

- Requires importing android.util.Log
- Log messages appear in the LogCat component of the Android Studio interface

Application Manifest

Every Android app must include an AndroidManifest.xml file describing functionality.

The manifest specifies:

- App's Activities, Services, etc.
- Permissions requested by app
- Minimum API required
- Hardware features required, e.g., camera with autofocus
- External libraries to which app is linked, e.g., Google Maps library

Application Manifest(cont.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.os5" android:versionCode="1" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="9" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ContactPickerActivity"
            android:label="@string/app_name">
            <!--
                <intent-filter> <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            -->
            <intent-filter>
                <action android:name="android.intent.action.PICK" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:path="contacts" android:scheme="content" />
            </intent-filter>
        </activity>
        <activity android:name=".ContentPickerTestExampleActivity" android:label="Content Picker Test">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
</manifest>
```

System level protection for Android app

- Each app runs as a unique user identity such that Android can limit the potential damage of programming flaws.
- Each app runs in own VM sandbox using unique UID
- Each app requests a simple permission label assignment model to restrict access to resources and other applications if necessary
- Ex. of permission: Internet, camera, GPS
- Permission specifies an access policy to protect its resources.

Android Permissions

- All permission of Android's policy are set at install time and can't change until the application is reinstalled.
- Android's permission only restricts access to components and doesn't currently provide information flow guarantees.
- A permission is listed in app's manifest definition XML file.
- If a public component doesn't explicitly declare any access permission, Android permits any application to access it.
- Component A's ability to access components B and C is determined by comparing the access permission labels on B and C to the collection of permission labels assigned to application A.

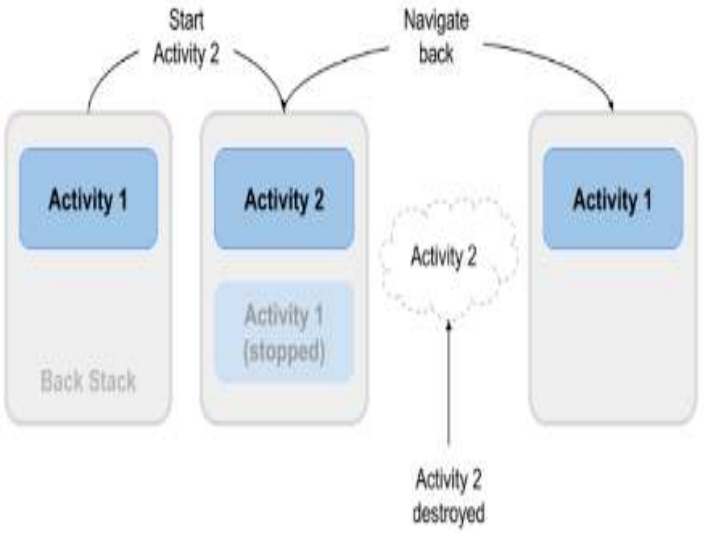
Activity Lifecycle

The activity lifecycle is the set of states an activity can be in during its entire lifetime.

As the user interacts with your app and other apps on the device, activities move into different states.

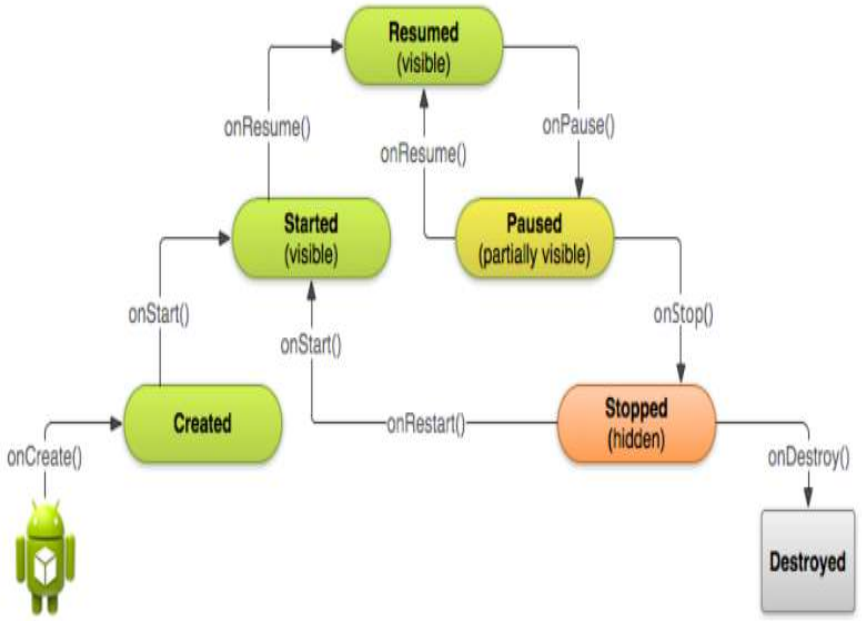
For example:

- When you start an app, the app's main activity ("Activity 1" in the figure belc is started, comes to the foreground, and receives the user focus.
- When you start a second activity ("Activity 2" in the figure below), a new act is created and started, and the main activity is stopped.
- When you're done with the Activity 2 and navigate back, Activity 1 resumes. Activity 2 stops and is no longer needed.
- If the user doesn't resume Activity 2, the system eventually destroys it.



Activity Lifecycle (cont.)

- Activity transitions into and out of the different lifecycle states.
- The Android system calls several lifecycle callback methods at each stage.
- The lifecycle states (and callbacks) are per Activity, not per app.



Activity Lifecycle (cont.)

Sr.No	Callback & Description
1	**onCreate()** This is the first callback and called when the activity is first created.
2	**onStart()** This callback is called when the activity becomes visible to the user.
3	**onResume()** This is called when the user starts interacting with the application.
4	**onPause()** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
5	**onStop()** This callback is called when the activity is no longer visible.
6	**onDestroy()** This callback is called before the activity is destroyed by the system.
7	**onRestart()** This callback is called when the activity restarts after stopping it.

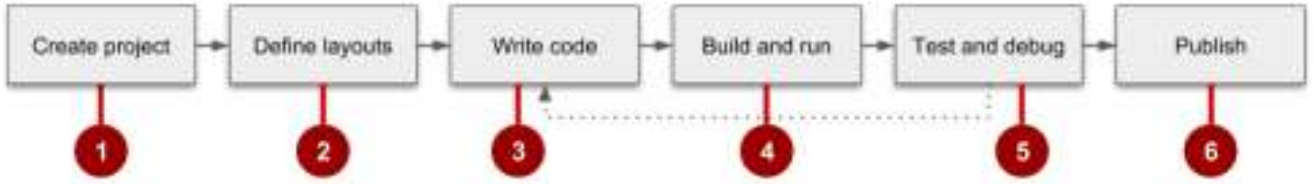
Activity Lifecycle (cont.)

Note: The `Activity` instance state is particular to a specific instance of an `Activity`, running in a single task. If the user force-quits the app or reboots the device, or if the Android system shuts down the app process to preserve memory, the `Activity` instance state is lost. To keep state changes across app instances and device reboots, you need to write that data to shared preferences. You learn more about shared preferences in another chapter.

Your first Android app

The development process:

1. Create the project in Android Studio and choose an appropriate template.
2. Define a layout for each screen that has UI elements. You can place UI elements on the screen using the layout editor, or you can write code directly in the Extensible Markup Language (XML).
3. Write code using the Java programming language. Create source code for all of the app's components.
4. Build and run the app on real and virtual devices. Use the default build configuration or create custom builds for different versions of your app.
5. Test and debug the app's logic and UI.
6. Publish the app by assembling the final APK (package file) and distributing it through channels such as Google Play.



The Main Activity File

- The main activity code is a Java file **MainActivity.java**.
- It gets converted to a Dalvik executable and runs your application.
- *Example:* ***Hello World!*** **Application:**
- Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the *res/layout* folder.
- The ***onCreate()*** method is one of many methods that are figured when an activity is loaded.

```
package com.example.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

`AppCompatActivity` class lets you use up-to-date Android app features

`setContentView()` method is called with the path to a layout file, the system creates all the initial views from the specified layout and adds them to your `Activity`

The Manifest File

- you must declare all its components in a *manifest.xml* which resides at the root of the application project directory.
- This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS.
- For example, a default manifest file will look like as following file:

The Manifest File(cont.)

- <application>...</application> tags enclosed the components related to the application.
- Attribute *android:icon* will point to the application icon available under *res/drawable-hdpi*.
- The application uses the image named ic_launcher.png located in the drawable folders.
- <activity> tag is used to specify an activity.
- *android:name* attribute specifies the fully qualified class name of the *Activity* subclass.
- *android:label* attributes specifies a string to use as the label for the activity.
- The **action** for the intent filter is named *android.intent.action.MAIN* to indicate that this activity serves as the entry point for the application.
- The **category** for the intent-filter is named *android.intent.category.LAUNCHER* to indicate that the application can be launched from the device's launcher icon.
- The *@string* refers to the *strings.xml* file.
- *@string/app_name* refers to the *app_name.*

The Strings File

- The **strings.xml** file is located in the *res/values* folder.
- It contains all the text that your application uses.
- For example, the names of buttons, labels, default text. For example, a default strings file will look like as following file:

```
<resources>
   <string name="app_name">HelloWorld</string>
   <string name="hello_world">Hello world!</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_main">MainActivity</string>
</resources>
```

The Layout File

- The **activity_main.xml** is a layout file available in *res/layout* directory.
- It is referenced by your application when building its interface.
- It is modified frequently to change the layout of your application.
- For your "Hello World!" application, this file will have following content related to default layout:

```
<RelativeLayout 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" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:padding="@dimen/padding_medium"
      android:text="@string/hello_world"
      tools:context=".MainActivity" />

</RelativeLayout>
```

Resources

Elliotte Rusty Harold and Scott Means, *XML Programming*, O'Reilly & Associates, Inc., 2002.

W3Schools Online Web Tutorials, http://www.w3schools.com.

https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-1-build-your-first-app/1-1-c-your-first-android-app/1-1-c-your-first-android-app.html

https://www.tutorialspoint.com/android/android_acitivities.htm

https://www.tutlane.com/tutorial/android/android-ui-layouts-linear-relative-frame-table-listview-gridview-webview