

Android

IN 1 HOUR

A lighting-fast introduction to Android development



Disclaimer

- This hands-on / getting started is by no means:
 - Complete (only scrapes the surface)
 - Showing best practices (would need more time)
 - Technically deep
 - Only superficial notions are provided
 - Guide to self-learning and self-documentation

Summary

- Short history
- Architecture
- Tool-chain
- Main Components
 - Types
 - LifeCycle
- Hands-on
 - Life cycle + Notification Manager
 - REST client

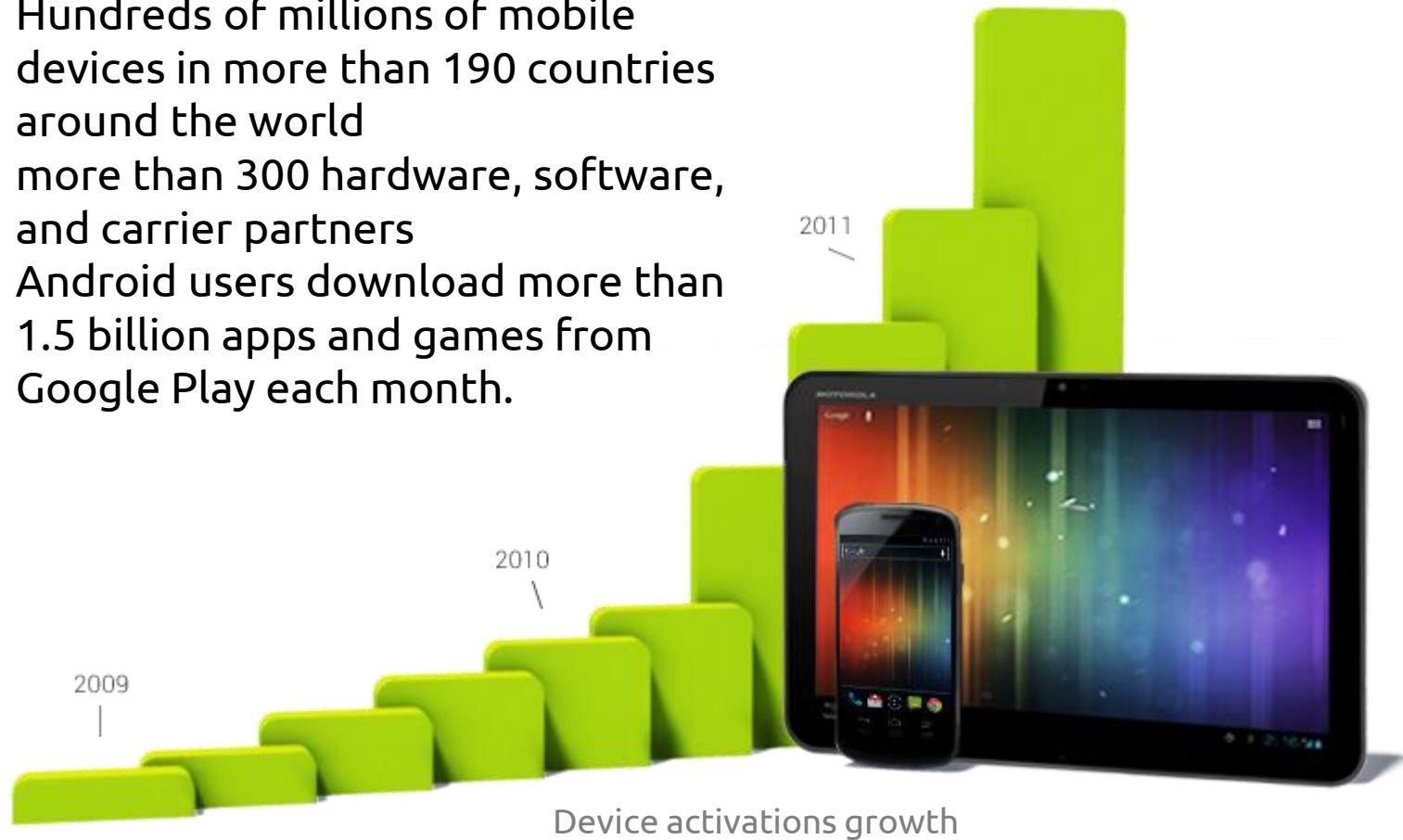
History

- Originally created by Andy Rubin
- Acquired by Google Inc. in 2005
- Several stable releases since then

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.4%
4.1.x	Jelly Bean	16	18.4%
4.2.x		17	19.8%
4.3		18	6.3%
4.4	KitKat	19	39.7%
5.0	Lollipop	21	1.6%

Figures

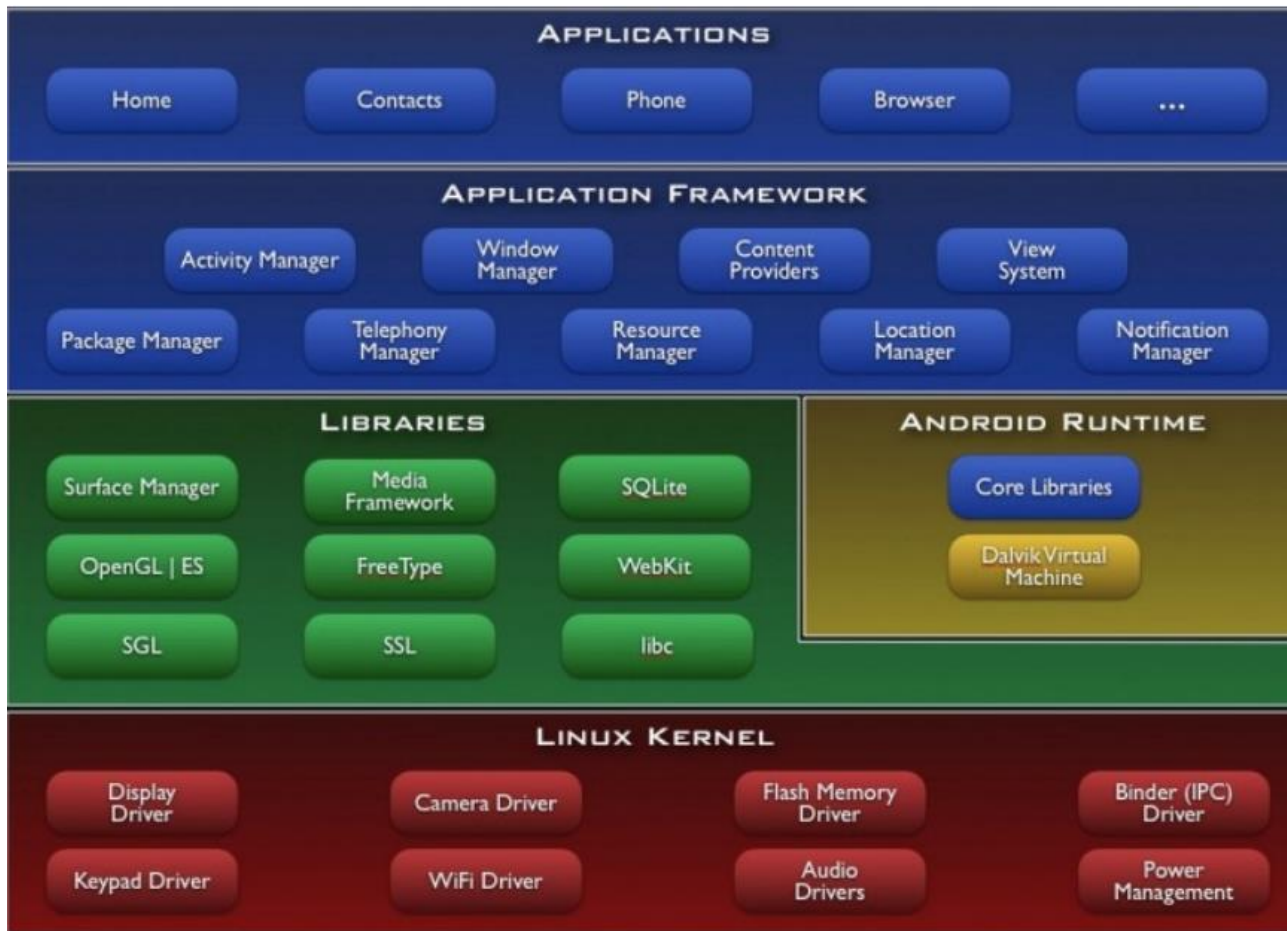
- Hundreds of millions of mobile devices in more than 190 countries around the world
- more than 300 hardware, software, and carrier partners
- Android users download more than 1.5 billion apps and games from Google Play each month.



Architecture

- Based on the Linux Kernel
- Open source platform, and Oracle (formerly Sun Microsystems) Java 6 Standard Edition, one of the world's most popular programming languages.
 - Android 5 (Lollipop) uses a 64-bit Linux Kernel, and Java 7.

Architecture



Java – based development

Dalvik VM

- Register based virtual machine
 - As opposed to normal stack based VMs
- Optimized to ensure that a device can run multiple instances efficiently.
 - More info at <http://www.dalvikvm.com/> and
 - <http://sites.google.com/site/io/dalvik-vm-internals>
- Uses own bytecode not Java bytecode
- No JIT
- Android programs are compiled into Dalvik executable files (.dex) which are then zipped into Android packages (.apk).
 - Dex-files are used to ensure minimal memory footprint.
 - Uncompressed dex-file is usually still smaller than the same Java code as compressed jar-file.

Application Framework

- Framework services
 - Activity manager
 - Views
 - Notification manager
 - Content providers
 - Resource manager
- Lots of APIs:
 - telephony, media, sensors, location, WebKit-based browser, Google Maps, P2P & Google Talk, home screen widgets, OpenGL, FreeType, SQLite to name a few

Toolchain – core tools

- Core components
 - Java SDK (6 or higher)
 - Android SDK
 - <https://developer.android.com/sdk/index.html>
 - Android Integrated Development Environment
 - Android Studio (official IDE, used in this hands-on)
 - Eclipse ADT
 - Android SDK emulator
 - Native images are faster
 - Can in general be quite slow
 - Third-party emulators
 - Based on hardware virtualization
 - Typically faster
 - E.g., GenyMotion

Toolchain – optional tools

- Additional tools
 - Graphic assets editor
 - E.g., GIMP
 - UI Designing and Wireframing
 - E.g., Pencil

Toolchain - Guidelines

- Google defines simple guidelines to design effective apps
- Based on 3 main principles
 - Enchant me
 - ...your app should strive to combine beauty, simplicity and purpose to create a magical experience that is effortless and powerful.
 - Simplify my life
 - ...simple tasks never require complex procedures, and complex tasks are tailored to the human hand and mind. People of all ages and cultures feel firmly in control, and are never overwhelmed by too many choices or irrelevant flash.
 - Make me amazing
 - ...android apps empower people to try new things and to use apps in inventive new ways.

Toolchain - Design

- Effective and well-known designs and patterns for applications
 - Best practices
 - Graphic elements
 - Colors
 - Typical navigation patterns
 - Etc.
- See the google developer site

Application life-cycle

- In an ideal case
 - all Android applications started by the user remain in memory,
 - Faster restart
- But
 - the available memory on an Android device is limited
- Therefore
 - the Android system is allowed to terminate running processes or recycling Android components.

Application life-cycle 2

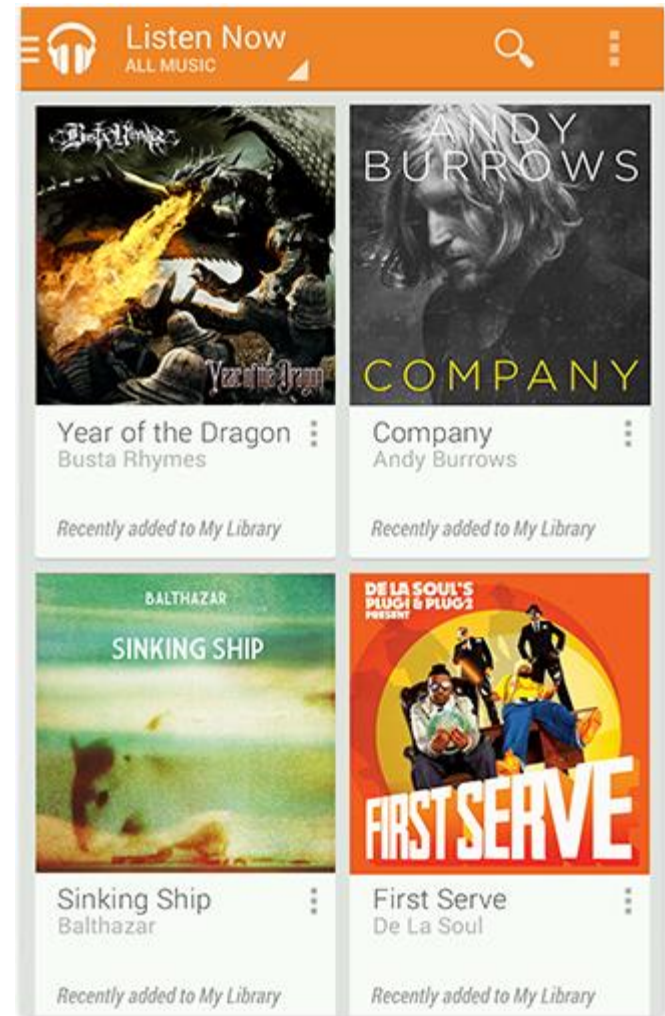
Process status	Description	Priority
Foreground	An application in which the user is interacting with an activity, or which has an service which is bound to such an activity. Also if a service is executing one of its lifecycle methods or a broadcast receiver which runs its onReceive() method.	1
Visible	User is not interacting with the activity, but the activity is still (partially) visible or the application has a service which is used by a inactive but visible activity.	2
Service	Application with a running service which does not qualify for 1 or 2.	3
Background	Application with only stopped activities and without a service or executing receiver. Android keeps them in a least recent used (LRU) list and if requires terminates the one which was least used.	4
Empty	Application without any active components	5

Main Components

- **Activities**
 - Handle the visible part of the application
- **Intents**
 - Activate services and components
- **Services**
 - Background services
- **Content providers**
 - Share data between activities and applications
- **Broadcast receivers**
 - Receive and react to broadcasted events
- **Resources**
 - Support different localizations and form factors easily

Activity

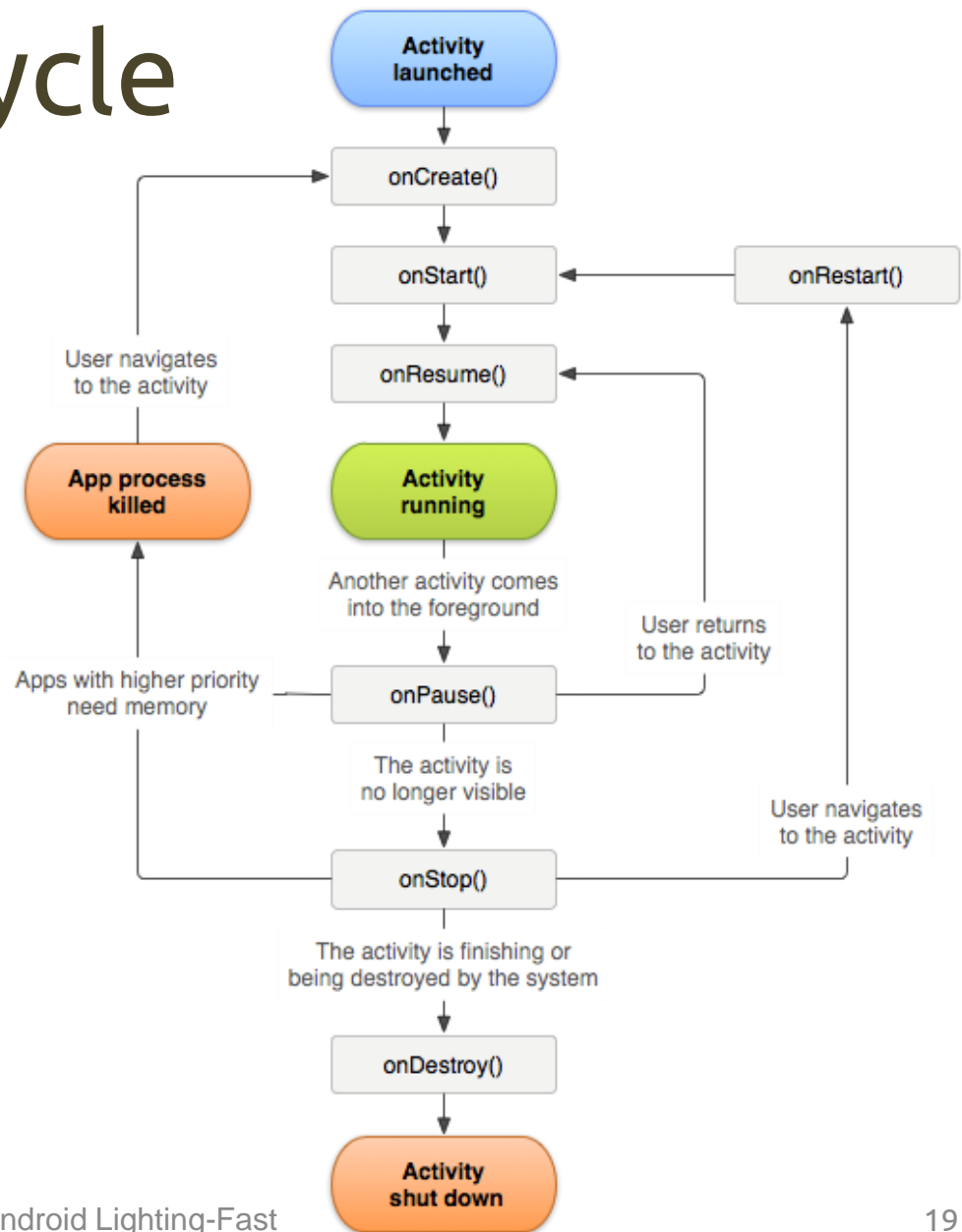
- Foreground activity is a visible screen in the application
- Handles user interaction and feedback
- Android applications have no exit. Platform handles the lifecycle.



Activity 2

- Application:
 - multiple activities that are loosely bound to each other.
- One activity in an application is specified as the "main" activity
 - presented to the user when launching the application for the first time.
- Each activity can start another activity
 - to perform different actions.
- Each time a new activity starts,
 - the previous activity is stopped,
 - the system preserves the activity in a stack (the "back stack").
- When a new activity starts,
 - it is pushed onto the back stack and takes user focus

Activity Lifecycle



Activity lifecycle 2

- 3 main states:
 - Resumed
 - The activity is in the foreground of the screen and has user focus. (This state is also sometimes referred to as "running".)
 - Paused
 - Another activity is in the foreground and has focus, but this one is still visible.
 - A paused activity is completely alive (the Activity object is retained in memory, it maintains all state and member information, and remains attached to the window manager)
 - Stopped
 - The activity is completely obscured by another activity (the activity is now in the "background"). A stopped activity is also still alive (the Activity object is retained in memory, it maintains all state and member information, but is not attached to the window manager).

Handling lifecycle

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

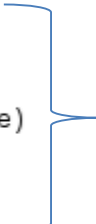
Hands-on

- Develop an Android app for showing the activity lifecycle
 - Simple empty activity
 - Exploit the Notification manager service to show current status through notifications


Hands-on 2

- Notification manager use


```
private void notify(String methodName) {  
    String name = this.getClass().getName();  
    String[] strings = name.split("\\.");  
    Notification noti = new Notification.Builder(this)  
        .setContentTitle(methodName + " " + strings[strings.length - 1]).setAutoCancel(true)  
        .setSmallIcon(R.drawable.ic_launcher)  
        .setContentText(name).build();  
    NotificationManager notificationManager =  
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);  
    notificationManager.notify((int) System.currentTimeMillis(), noti);  
}
```



Build the
notification



Get the
Manager



Send the
notification

Hands-on 3

- Generate a new notification for each call
 - onCreate(Bundle savedInstanceState)
 - onPause()
 - onResume()
 - onStop()
 - onDestroy()
 - onRestoreInstanceState(Bundle savedInstanceState)
 - onSaveInstanceState(Bundle outState)

Intents

- An Intent is a messaging object you can use to request an action from another app component.
- 3 main uses:
 - To start an activity
 - by passing an Intent to `startActivity()`
 - To start a service
 - To deliver a broadcast
 - A broadcast is a message that any app can receive
 - by passing an Intent to `sendBroadcast()`, `sendOrderedBroadcast()`, or `sendStickyBroadcast()`.

Hands-on

- Develop a simple app for showing Intents adoption
 - 1 Home Activity
 - 1 View handled by the Home Activity and showing one button
 - Upon button click, use an Intent to start a new Activity

Hands-on 2

- Add a button to the main activity layout

```
<Button  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/home_start_activity"  
    android:id="@+id/button"  
    android:clickable="true"  
    android:onClick="newActivityClicked"/>
```

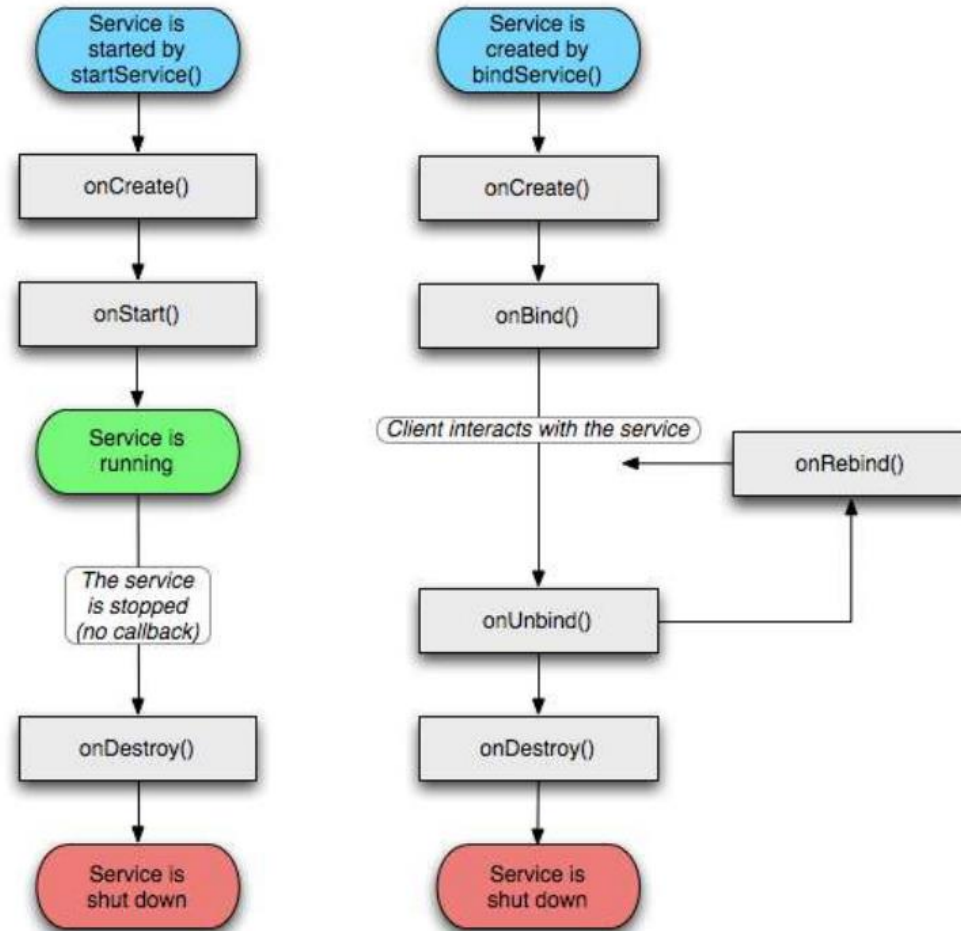
- Add an handler for the button

```
public void newActivityClicked(View view)  
{  
    Intent startActivityIntent = new Intent(this, SecondActivity.class);  
    this.startActivity(startActivityIntent);  
}
```

Service

- A Service is an application component that
 - can perform long-running operations in the background
 - does not provide a user interface.
- Any application component can use the service (even from a separate application)
 - by starting it with an Intent.
- Runs still on the main Activity's thread
 - Need to separate to own thread if needed
 - Communication using e.g. Handlers if in separate threads
- Lifecycle differs from Activity
 - Entire lifecycle vs active lifecycle
 - onCreate, onStart, onDestroy or
 - onCreate, onBind, unbind, (rebind), onDestroy
 - According to starting mechanism.

Service lifecycle



Long-running tasks

- Long tasks should not “jeopardize” the app main thread
 - Run long tasks in separate threads
 - (be aware that services are not on separate threads by default)
- The most effective way to create a worker thread for longer operations is
 - Using the AsyncTask class.
 - Simply extend AsyncTask and
 - Implement the doInBackground() method to perform the work
 - Post progress changes by calling publishProgress(), which invokes the onProgressUpdate() callback method

Example

```
public class AsyncJsonGetter extends AsyncTask<String, Void, JSONObject>
{
    private JsonGetterCompletionListener theListener;

    public AsyncJsonGetter(JsonGetterCompletionListener listener) { this.theListener = listener; }

    @Override
    protected void onPostExecute(JSONObject obj)
    {
        super.onPostExecute(obj);

        //notify the listener
        this.theListener.onTracksLoadingComplete(obj);
    }

    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected JSONObject doInBackground(String... params)
    {
        JSONObject tracks = null;

        try
        {
            //prepare the URL representing the API endpoint to call
            URL flaskAPIURL = new URL(params[0]);

            //get a connection object for the given url
            HttpURLConnection connection = (HttpURLConnection) flaskAPIURL.openConnection();

            //set the connection method
            connection.setRequestMethod("GET");

            //connect
            connection.connect();
        }
    }
}
```

Hands-on

- Write a client app for the Flask REST Music Server developed in the course
- See the course GitHub for the result.

More to come...

- Many open topics and issues
- Where to go?
 - Android developer documentation
 - Android developer forums
 - Books
 - Play and learn

Questions?

01PRD AMBIENT INTELLIGENCE: TECHNOLOGY AND DESIGN

Dario Bonino
bonino@ismb.it

