Android

Deep dive into developing MobileApp using Android

Team Emertxe

Android Fundamentals

What is Android

What is Android

- Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google.
- Android is designed primarily for touchscreen mobile devices such as smartphones and tablet computers.
- Android's source code is released by Google under open source licenses.
- Android is the most popular mobile OS. As of 2013, Android devices sell more than Windows, iOS, and Mac OS devices combined.

History And Versions

Android Version And Features

Version	Release Date	Features
1.6 (Donut)	15 Sep 2009	•Improved Android Market experience •Gallery now enables users to select multiple photos for deletion • Support for CDMA and GSM • Support for more screen sizes and resolutions • Quick search box with updated Voice Search, integration with native applications •Updated technology support text-to-speech engine

Android Version And Features

Version	Release Date	Features
2.0/2.1 (Eclair)	26 Oct 2009	•New capabilities in Accounts, contacts & Sync Management •Quick Contact •Optimized hardware speed •New Browser UI and HTML 5 support •Improved Google Maps 3.1.2 •MicrosoftExchage support •Built in flash support for Camera •MotionEvent class enhanced to track multi-touch events •Improved virtual keyboard •Bluetooth 2.1 •Live Wallpapers/Folder

Android Version And Features

Version	Release Date	Features
2.2 (Froyo)	20 May 2010	•OS speed, memory, and performance optimizations •Dalivk JIT Complier •Improved application launcher with shortcuts to Phone and Browser applications •USB tethering and Wi-Fi hotspot functionality •Automatic App update •Quick switching between multiple keyboard languages and their dictionaries •Support for numeric and alphanumeric passwords •Support for file upload fields in the Browser application •Adobe Flash10.1 support

Android Version And Features

Version	Release Date	Features
2.3 (Gingerbread)	6 Dec 2010	•Updated user interface design •New audio effects •Support for Near Field Communication •Redesigned multi-touch software keyboard •Enhanced support for native code development •Audio, graphical, and input enhancements for game developers •Concurrent garbage collection for increased performance •Native support for more sensors (such as gyroscopes and barometers) •A download manager for long running downloads •Improved power management and application control

Android Version And Features

Version	Release Date	Features
3.1 and 3.3 (Honeycomb)	22 Feb 2011	• It was the first Android OS version which was specifically designed for tablets. •It adds toolbars at top and bottom and incorporates tabbed browsing and other desktop features. •Honeycomb uses Microsoft's Media Transfer Protocol (see MTP) for file transfer rather than connecting as a USB mass storage device.

Android Version And Features

Version	Release Date	Features
4.0(Ice Cream Sandwich) 	19 Oct 2011	• New typeface called Roboto • New Face unlock feature • Android Beam – A secure NFC powered content sharing platform • Re-arrangeable folders, Favorites Tray, Screenshots • Swipe to dismiss notifications, tasks and browser tabs • W-Fi Direct • Manage apps running in background • Revamped Gmail interface.

Android Version And Features

Version	Release Date	Features
4.1, 4.2 and 4.3(Jelly Bean	27 June 2012	• User interface has been made faster and smoother under Project Butter. • Notification center has been improved with expandable and actionable notifications • Offline voice recognition and typing facility. • Better Google Voice Search. • Support for external Braille input in enhanced Accessibility options. • Android Beam with enhanced option to transfer photos and videos. • Google Now • App encryption and Smart App updates

Android Version And Features

Version	Release Date	Features
4.4 (KitKat)	November 2013	• Faster Multi-tasking • A smarter caller id • Print where ever, when ever. • Pick a file, any file

Installing Softwares

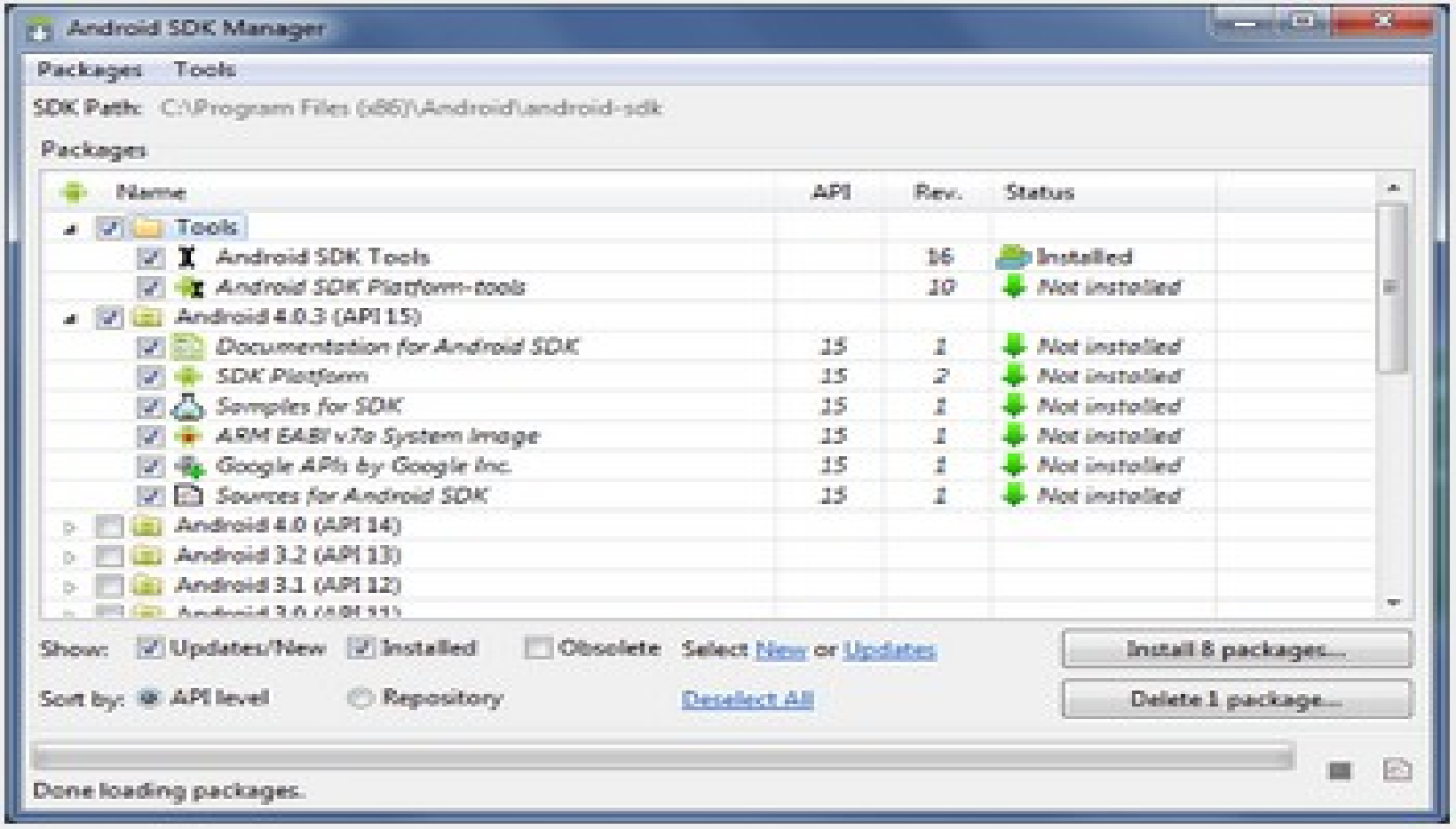
Installing Android Software

- You can download Eclipse ADT bundle for development.
- Further you should add more package to make the SDK a complete one.

Adding SDK Packages

- In Eclipse or Android Studio, click SDK Manager in the toolbar.
- If you're not using Eclipse or Android Studio:
- Windows: Double-click the SDK Manager.exe file at the root of the Android SDK directory.

Adding SDK Packages



EMERTXE

Hello Android Example

Hello Android Example

Summary steps to develop an Android application :

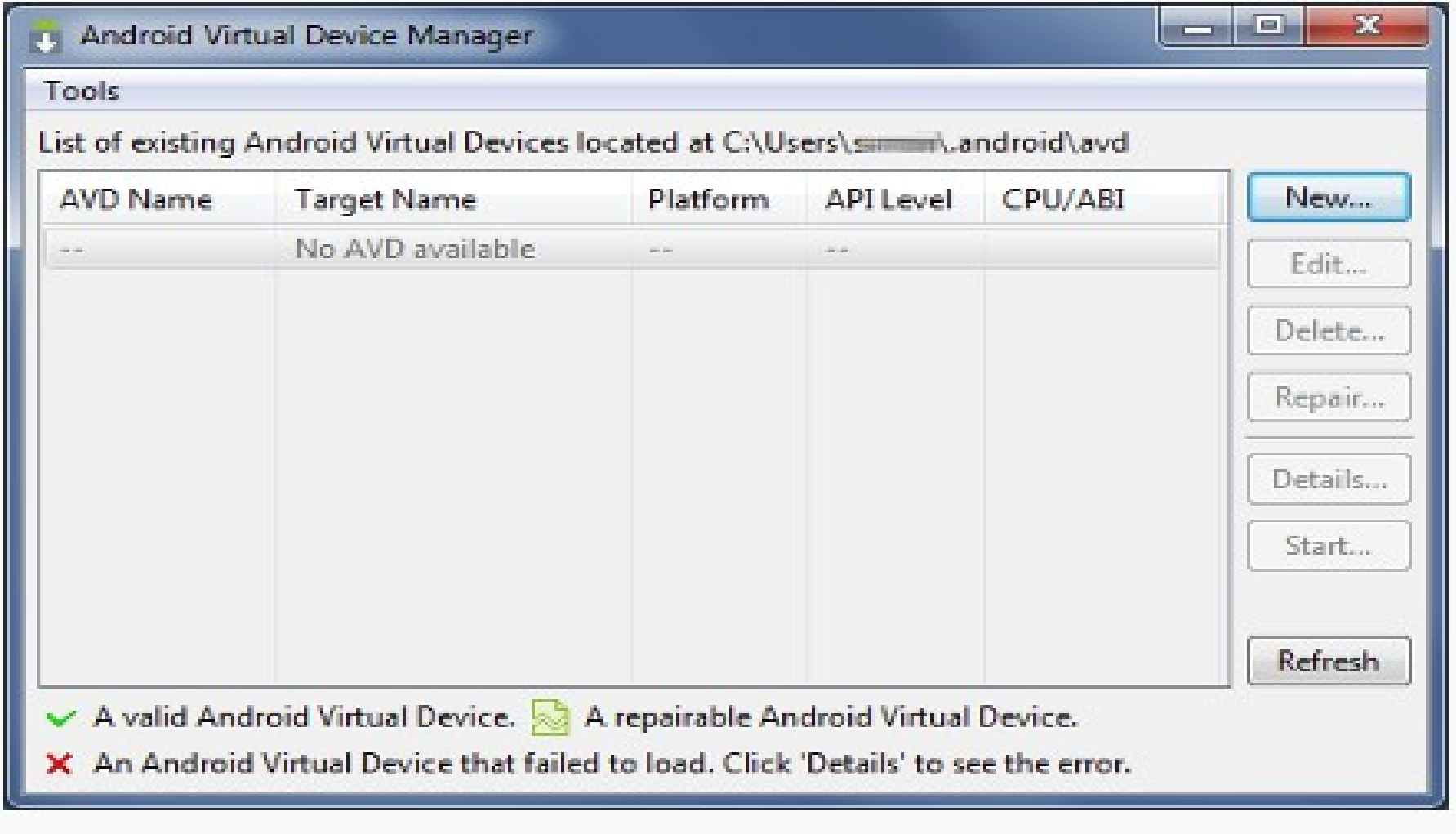
- Install Android SDK
- Install ADT Eclipse plugin
- Create an Android Virtual Device (AVD)
- Create Android Project with Eclipse (Wizard)
- Code it…
- Start it in Android Virtual Device (AVD)

Hello Android Example

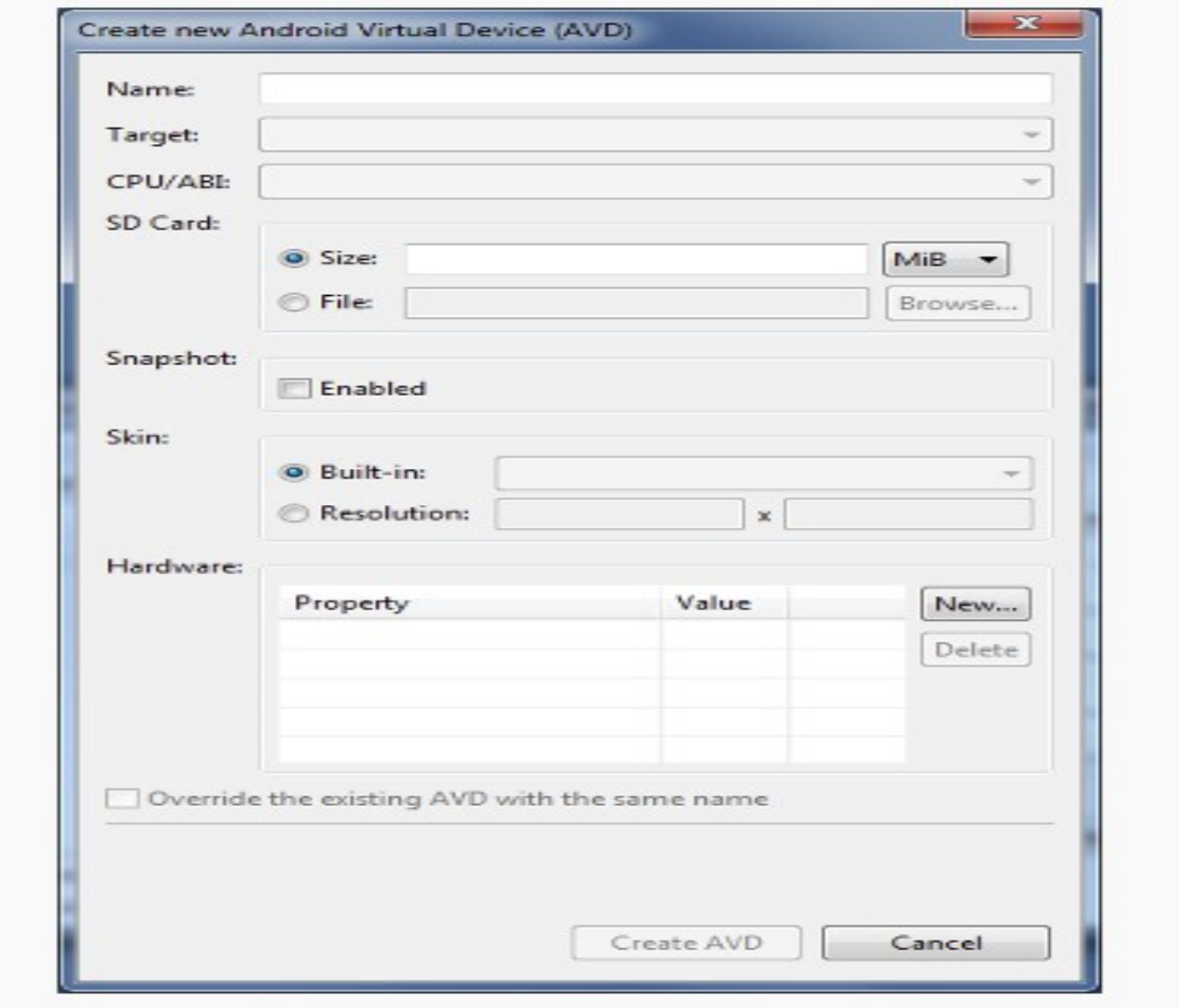
Tools used in this tutorial :

- JDK 1.6
- Eclipse IDE 3.7 , Indigo
- Android SDK

Hello Android Example



Hello Android Example

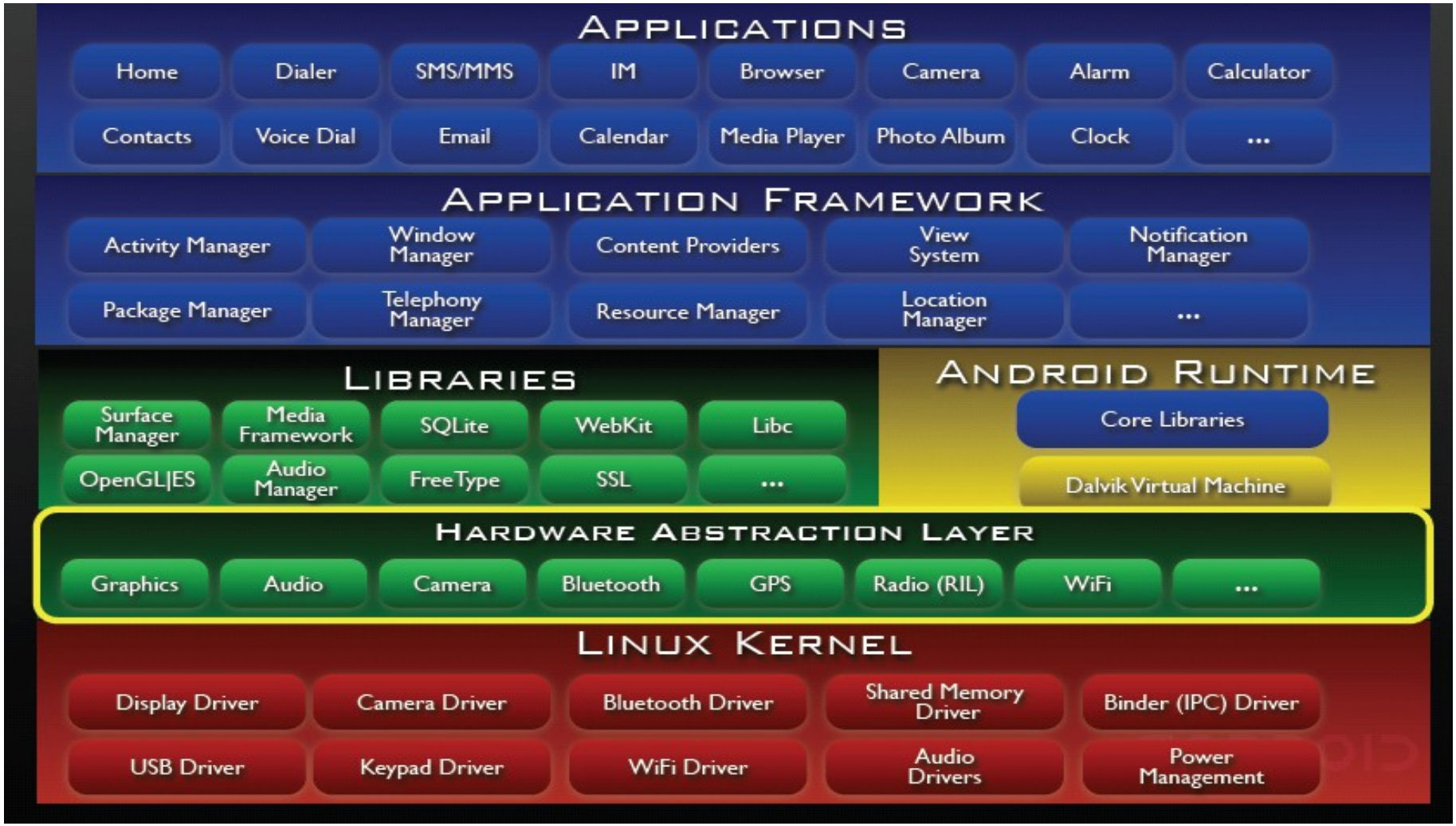


EMERTXE

Hello Android Example

Android Architecture/Software Stack

Android Architecture



Architecture Detail

- Android is build on Linux 2.6.24 kernel
- Does not include the full set of standard Linux utilities
- It has a core capability like
  - Security
  - Memory management
  - Process management
  - Network stack
  - Driver model
  - Abstraction Layer

Architecture Detail

- It has a core capability like:
  - Display Driver
  - Camera Driver
  - Bluetooth Driver
  - Flash Memory Driver
  - USB Driver
  - Keypad Driver
  - WiFi Driver
  - Audio Driver
  - Power Management

Libraries

- Native libraries.
- Written in C/C++ internally
- Called through Java wrapper APIs
- Layer contains
  - Surface Manager (for compositing windows),
  - 2D and 3D graphics,
  - Media codecs (MPEG-4, H.264, MP3, etc.),
  - SQL database (SQLite) and
  - A native web browser engine (WebKit).

Functional Libraries - WebKit

- Open source WebKit browser: http://webkit.org
- Renders pages in full (desktop) view
- Full CSS, Javascript, DOM, AJAX support

SQLite

- Server less, transactional SQL database engine.
- Most widely deployed SQL database engine in the world.
- Light-weight transactional data store
- Back end for most platform data storage
- Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
- Zero-configuration - no setup or administration needed.
- A complete database is stored in a single cross-platform disk file.
- Cross-platform: Unix (Linux and Mac OS X), OS/2, and Windows (Win32 and WinCE) supported.

Dalvik Virtual Machine

Dalvik Virtual Machine

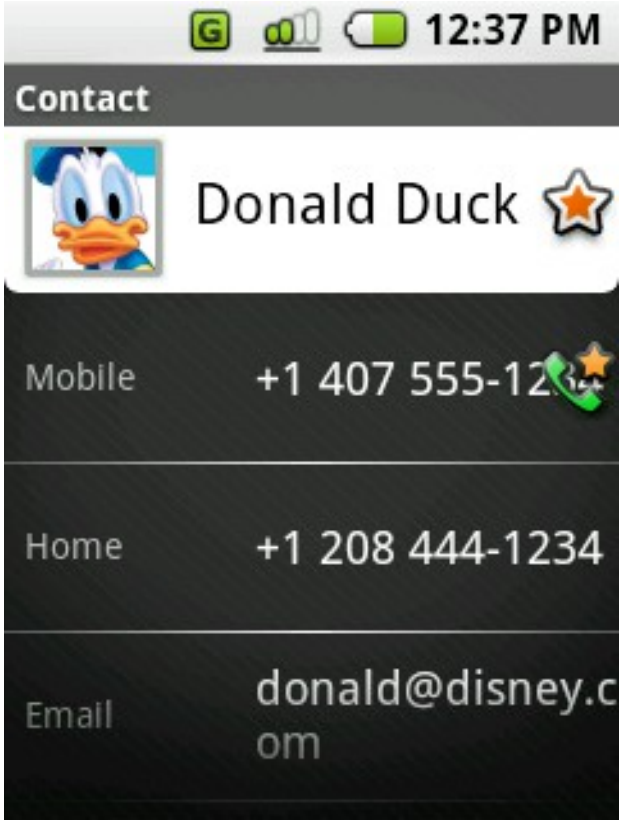
- Highly optimized VM, compiled byte is optimized for mobile devices
- Register based NOT Stack based
- NOT a JVM
- End result is NOT the same byte code as Java,
- dx in the sdk takes compiled Java class files and converts them into .dex
- Built with security and performances (battery life) in mind

Application Framework

- **Content Providers** – access, store and share application data
- **Resource Manager -** access to strings, graphics, and layout files
- **Notification Manager** - Display custom alerts in the status bar
- **Activity Manager** - Manages the life cycle of applications
- **Window Manager-** rendering view on the screen
- **View System -** UI components
- **Package Manager -** Application formation
- **Location Manager -** Location based services

Application Layer

- Top layer
- Code will live here
- Built-in applications such as the Phone and Web Browser.
- Applications Google writes have to go through the same public API that you use.
- You can even tell Android to make your application replace the standard applications, if you like.



ΣMERTXE

Dalvik Runtime

- **Dalvik** is the virtual machine (VM) in Android OS
- Used on mobile devices such as mobile phones, tablets & netbooks.
- Android applications are converted into the compact **Dalvik Executable** (**.dex**)
- Dalvik VM is a register-based architecture.
- A tool called **dx** is used to convert some (but not all) Java .class files into the .dex format.

Dalvik Runtime

- Multiple classes are included in a single .dex file.
- As of Android 2.2, Dalvik has a just-in-time compiler.
- Being optimized for low memory requirements, Dalvik has some specific characteristics that differentiate it from other standard VMs:
  - The VM was slimmed down to use less space
  - It uses its own bytecode, not Java bytecode
  - Moreover, Dalvik has been designed so that a device can run multiple instances of the VM efficiently.

Dalvik vs JVM

***Dalvik Virtual machine** (DVM)*

- Register Architecture
- Designed to run on low memory,
- Uses its own byte code
- Runs .Dex file (Dalvik Executable File

***Java Virtual Machine** (JVM)*

- Stack Architecture
- Uses java byte code
- Runs .class file having JIT.

Android Runtime

- Android application == a process
- A process == a instance of the Dalvik virtual machine.
- Dalvik can run multiple VMs efficiently.
- Executes classes compiled by a Java language compiler that have been transformed into the .dex format.
- DVM executes files in the Dalvik Executable (.dex) format which is optimized.

Android Building Blocks

Objectives

At the end of this module you will be able to:

- Understand building blocks
- Android Activity
- Intent and Intent Filters
- Android Content Providers

Is Android Linux

- NO, android is not linux !
- Android is based on a linux kernel but it's not GNU/Linux
- No glibc support

So Android is Java?

- NO, android is not java !
- Android is not an implementation of any of the Java variants
- Uses the java language
- Implements part of the Java5 SE specification
- Runs on a dalvik virtual machine instead of JVM

EMERTXE

Android Building Blocks

- Recognize the fundamental building blocks
- Use these building blocks to create applications
- Understand applications life cycle.

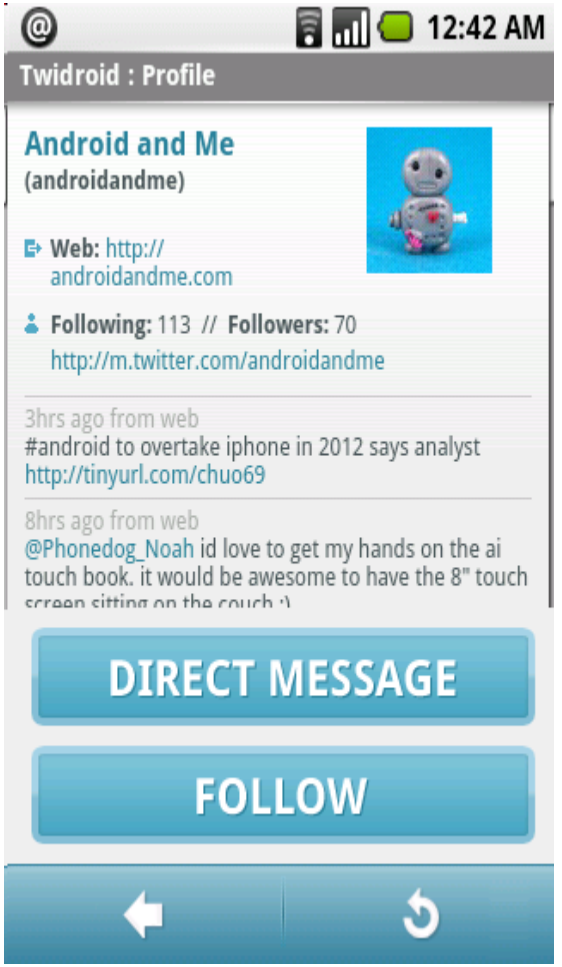
Building Blocks

Activities

- Activities are stacked like a deck of cards
- Only one is visible
- Only one is active
- New activities are placed on top

Application Building Blocks Activity

- Activity - UI component
- One screen.
- For example:
    - List of menu items users can choose.
    - Text messaging application
    - List of contacts to send messages
- Each activity is independent of the others.
- Each one is implemented as a subclass of the Activity base class.



Activities State

- Active
  - At the top of the stack
- Paused
  - Lost focus but still visible
  - Can be killed by LMK (low memory killer)
- Stopped
  - Not at the top of th stack
- Sropped
  - Killed to reclaim its memory

EMERTXE

Application Building Blocks Views

- Views are basic building blocks
- Know how to draw themselves
- Respond to events
- Organized as trees to build up GUIs
- Described in XML in layout resources

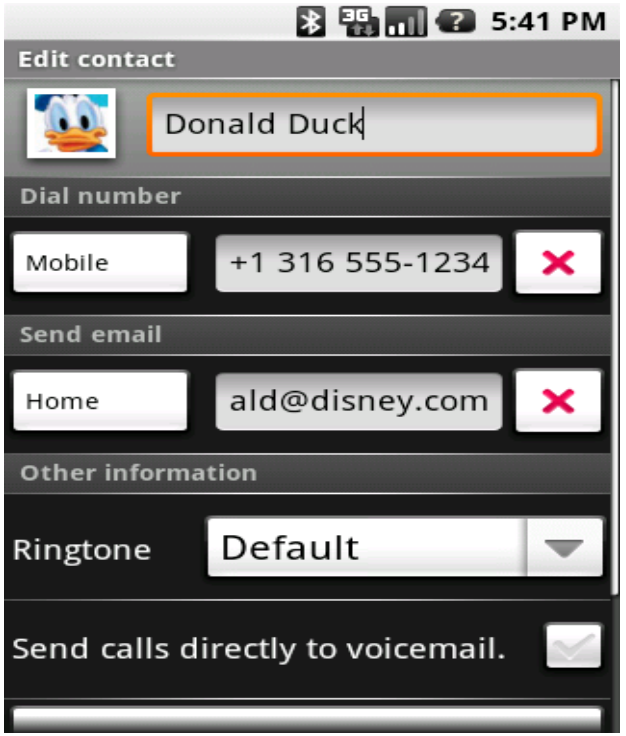
Pattern Load Layout

- Android compiles the XML layout code that is later loaded in code usually by

```
public void onCreate(Bundle
savedInstanceState) {
...
setContentView(R.layout.filename);
...
}
```

Views And View Groups

- Views and View groups trees build up complex GUIs
- Android framework is responsible for:
  - Measuring
  - laying out
  - drawing



Application Building Blocks Intents

- Intents are used to move from Activity to Activity
- Describes what the application wants
- Provides late runtime binding

action	the general action to be performed, such as VIEW, EDIT, MAIN, etc.
data	the data to operate on, such as a person record in the contacts database, as URI

EMERTXE

Intent

- IntentReceiver
- Set and respond to notifications or status changes.
- Can wake up your app.
- Wakes up a predefined action through the external event.
- For example:
  - Alarm notification, when the user receives email.

Application Building Blocks Services

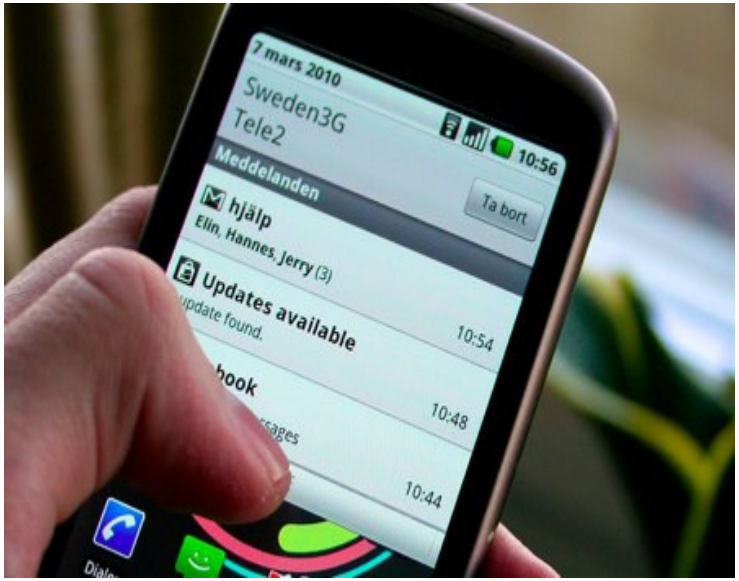
- Services run in the background
- Don't interact with the user
- Run on the main thread of the process
- Is kept running as long as
  - is started
  - has connections

Service

- Faceless task that runs in the background.
- No visual user interface
- Runs in the background certain period of time
- For example: Background music
- Each service extends the Service base class.

Application Building Blocks Notifications

- Notify the user about events
- Sent through NotificationManager
- Types
  - Persistent icon
  - Sound or vibration



EMERTXE

Application Building Blocks ContentProvider

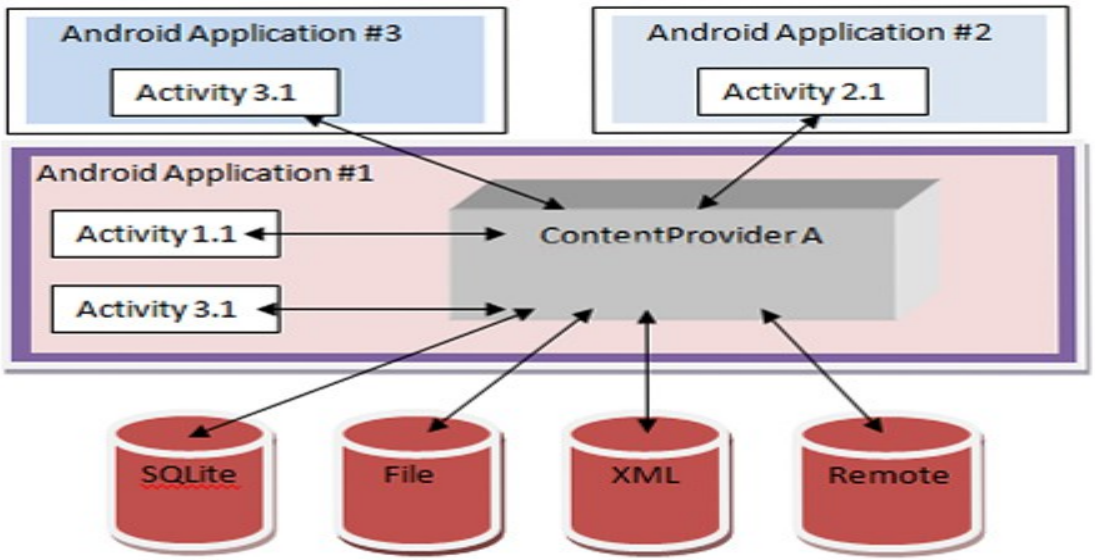
- ContentProviders are objects that can
  - Retrieve data
  - Store data
- Data is available to all applications
- Only way to share data across packages
- Usually the backend is SQLite
- They are loosely linked to clients
- Data exposed as a unique URI

ContentProviders

- Enable applications to share data.
- Makes a specific set of the application's data available to other applications.
- Date stored in the file system, in an SQLite database
- Extends the ContentProvider base class
- Enable other applications to retrieve and store data of the type it controls.

Application Building Blocks

• Components	• Description
Activities	UI Components typically corresponding to One Screen
Intent Receivers	Response to broadcast Intents
Services	Faceless tasks that run in background
Content Providers	Enable Application to share data



EMERTXE

Android Tools

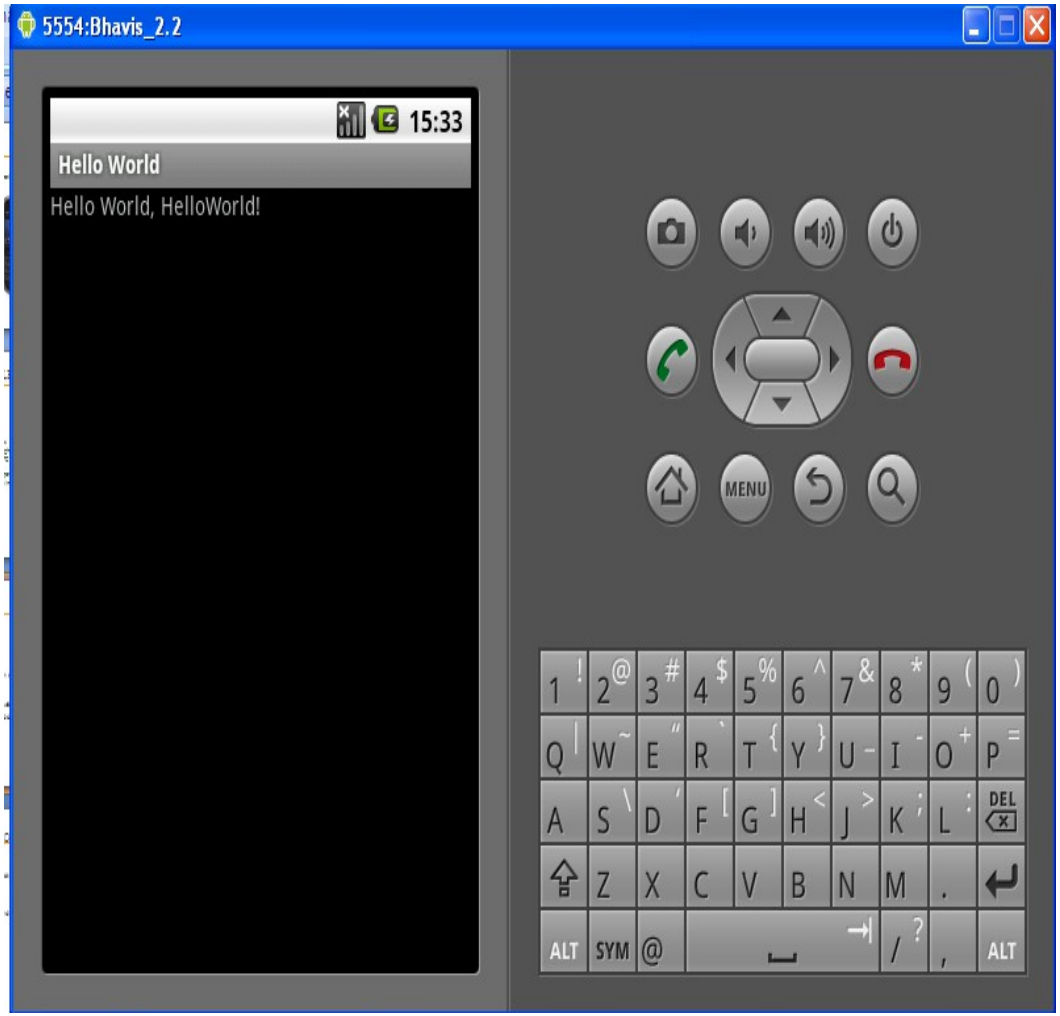
Objectives

At the end of this module you will be able to:

- Understand about Emulator
- Eclipse plugin
- Debugging
- DDMS

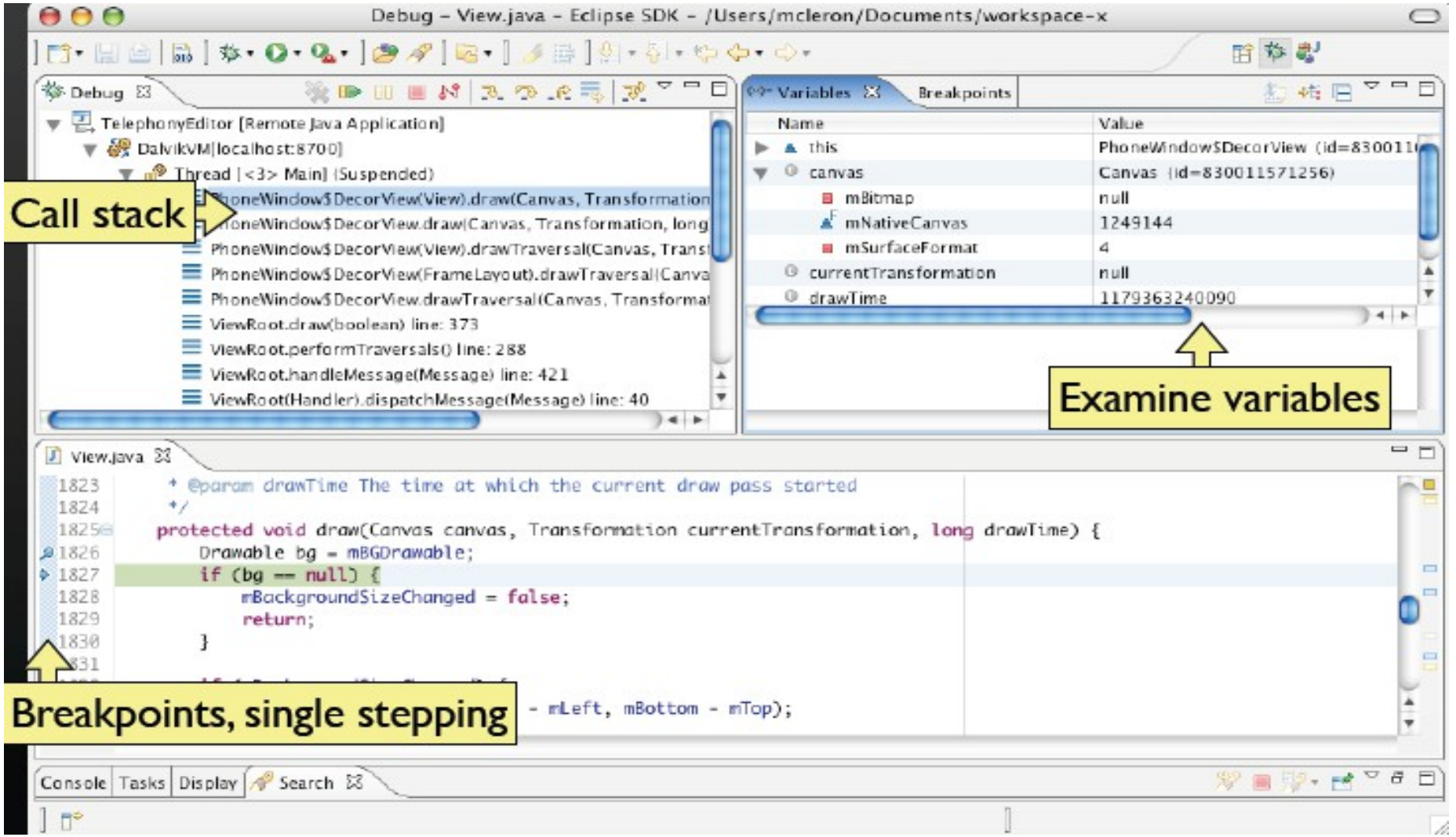
Emulator

- Emulator runs same system image as a device.



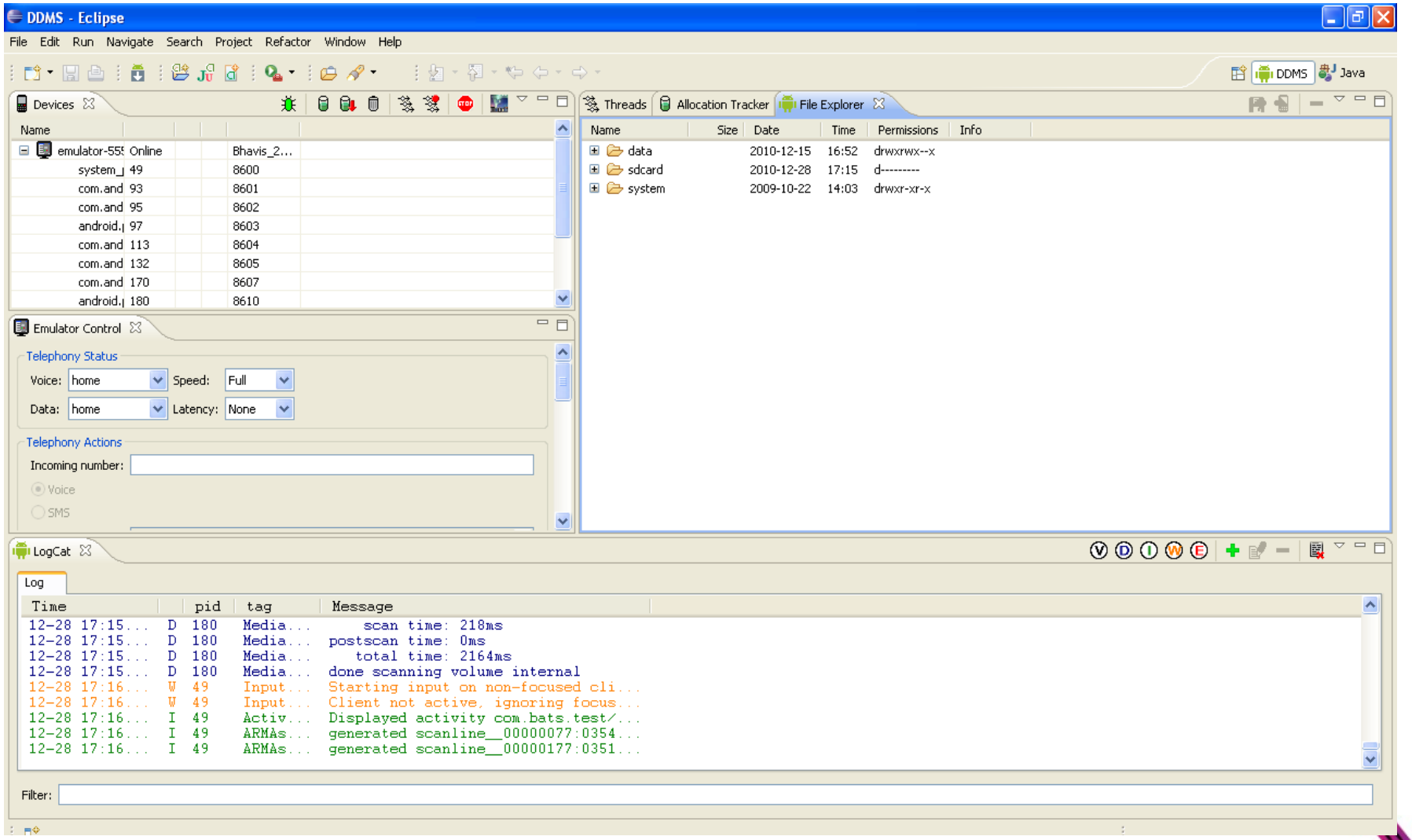
EMERTXE

Debugging



ΣMERTXE

DDMS

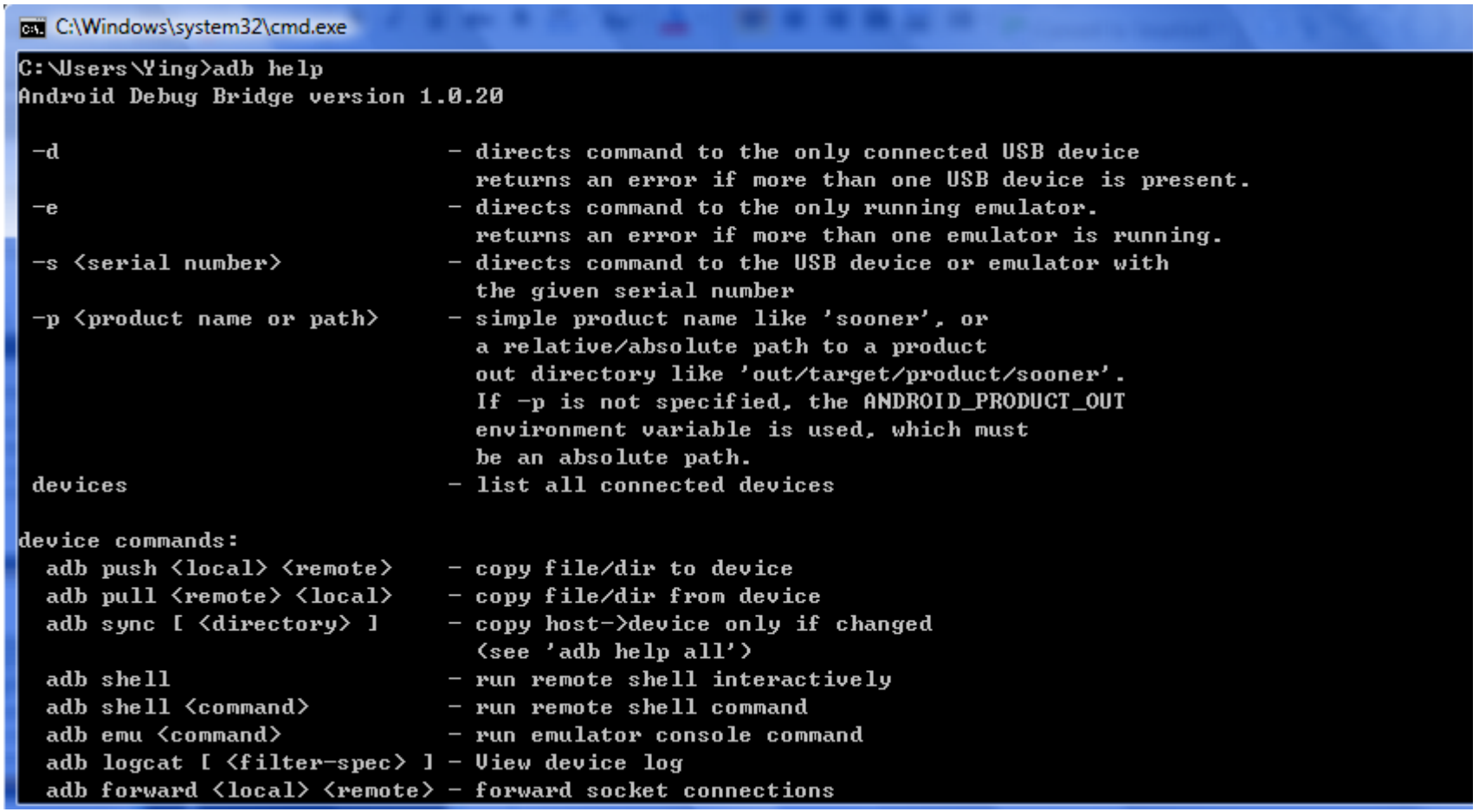


AndroidManifest.xml

- Starts with tag <manifest>
- uses-sdk
- uses-configuration
- uses-feature
- supports-screens
- Application
- service
- Provider
- Receiver
- Uses-permission
- Permission

EMERTXE

ADB


C:\Windows\system32\cmd.exe
C:\Users\Ying>adb help
Android Debug Bridge version 1.0.20
-d - directs command to the only connected USB device
returns an error if more than one USB device is present.
-e - directs command to the only running emulator.
returns an error if more than one emulator is running.
-s <serial number> - directs command to the USB device or emulator with
the given serial number
-p <product name or path> - simple product name like 'sooner', or
a relative/absolute path to a product
out directory like 'out/target/product/sooner'.
If -p is not specified, the ANDROID_PRODUCT_OUT
environment variable is used, which must
be an absolute path.
devices - list all connected devices
device commands:
adb push <local> <remote> - copy file/dir to device
adb pull <remote> <local> - copy file/dir from device
adb sync [<directory>] - copy host->device only if changed
(see 'adb help all')
adb shell - run remote shell interactively
adb shell <command> - run remote shell command
adb emu <command> - run emulator console command
adb logcat [<filter-spec>] - View device log
adb forward <local> <remote> - forward socket connections

EMERTXE

Questions

- What is Android?
- What are Android Features?
- What is the difference between Dalvik and JVM?

User Interface

Agenda

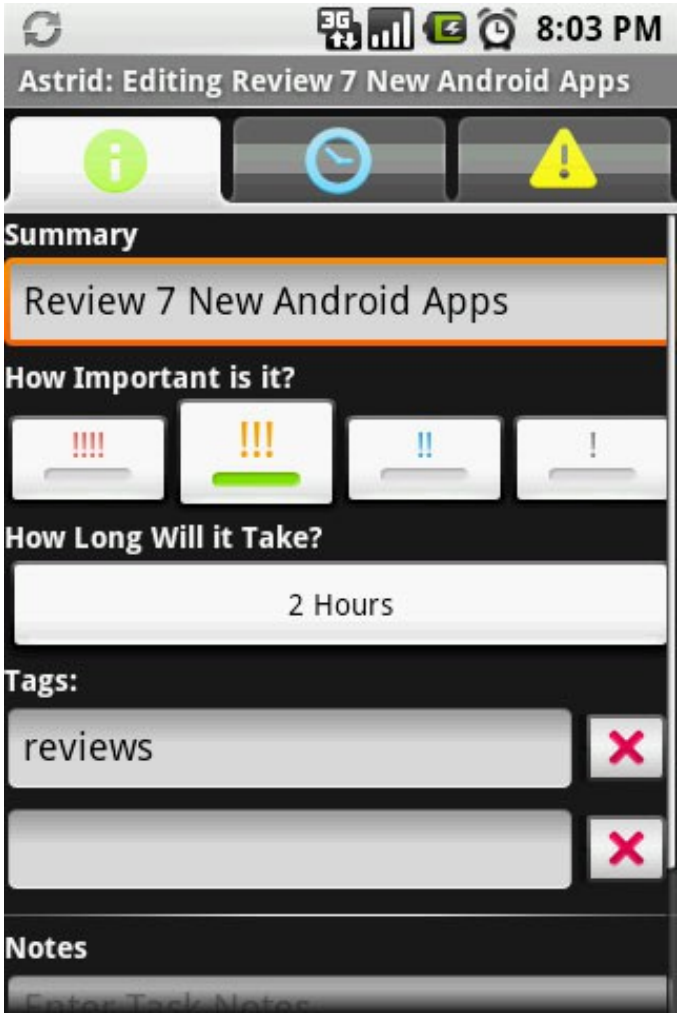
- Why is UI important?
- Bad UI
- Good UI
- Application UI design philosophy
- Android UI features
- Basic android UI elements
- Tools- The Android draw tool
- Layout
- Android advanced UI

Why is UI important?

- Better UI
- Perceived quality + polish
- Better rating
- Better app ranking
- More installs/purchases

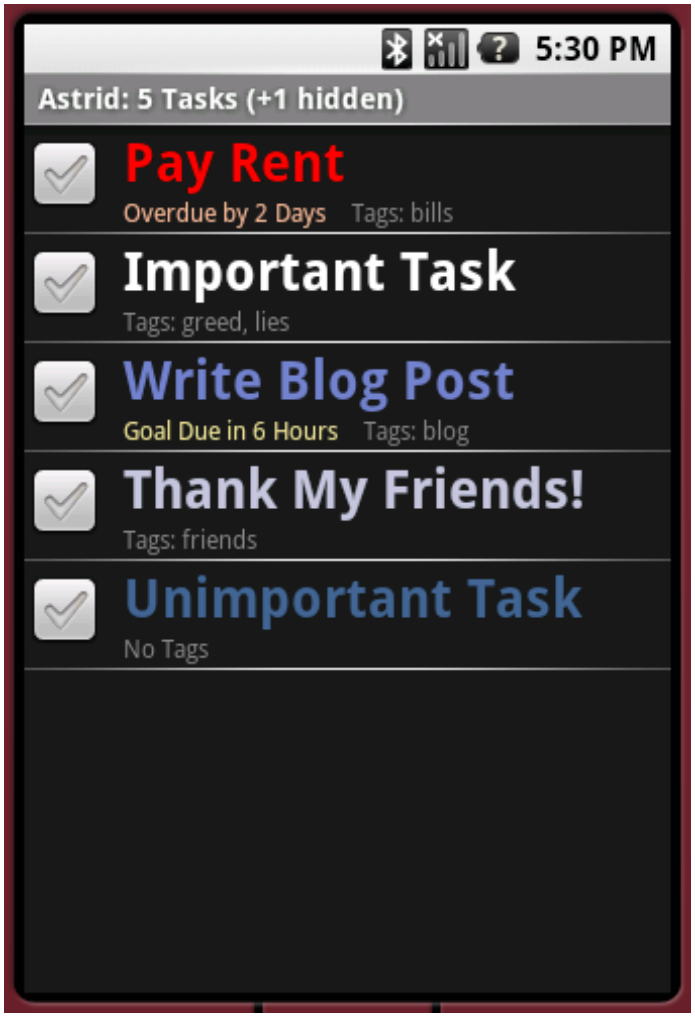
Bad UI

Too many views



Bad UI

Too many colors?



Good UI

Clear design

Application UI Design Philosophy

- “Successful application will offer an outstanding end-user experience”
- Fast
- Simple
- Seamless
- Most iOS apps are poular because they have an outstanding UI and are fast!
- The industry takes user experience seriously!

Android UI Features

- Rich UI model, huge number of components
- Code in Java or XML
- Based on the MVC model
- New APIs, no J2ME, AWT, Swing
- Both keypad and touch modes
- Support for different resolutions
- Unified UI experience across device

Basic Android UI Elements

- Views
- Layouts
- Shapes
- Colors
- Gradients

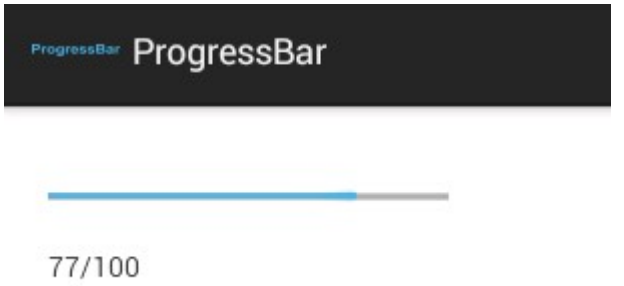
Basic UI components

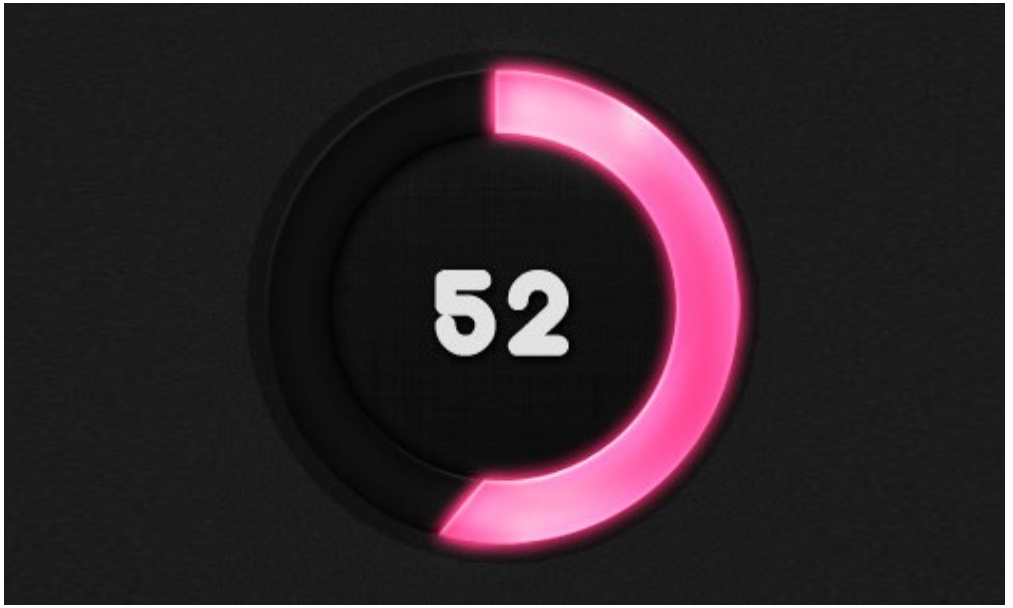
- Button,
- ImageButton,
- CheckBox
- EditText,
- ImageView
- TextView
- ListMenuItemView
- RadioButton
- RadioGroup

EMERTXE

Basic UI Component

- Progress Bar

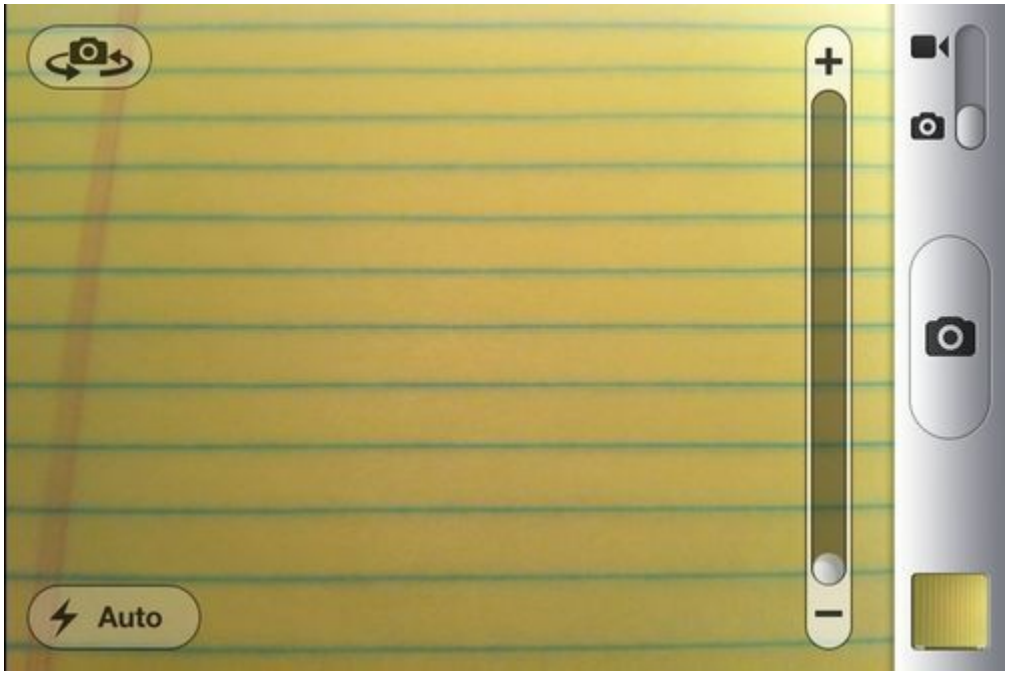




EMERTXE

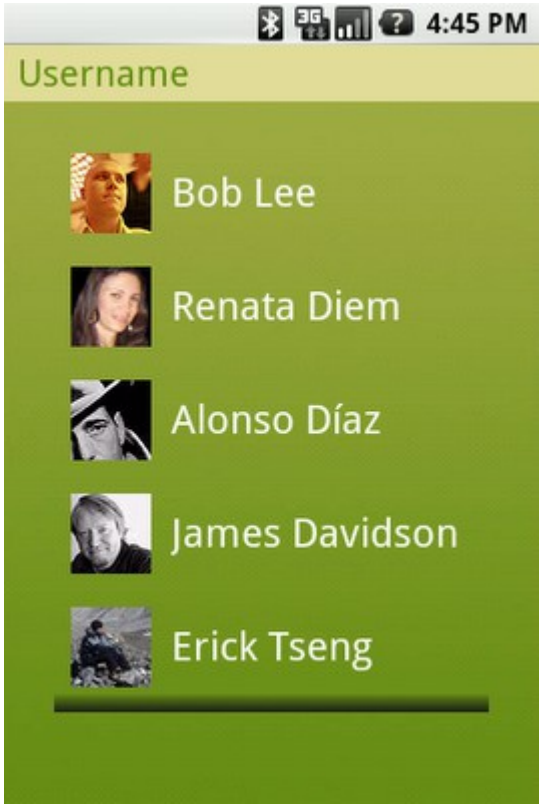
Zoom Slider

- Zoom Slider



ListView

- ListView

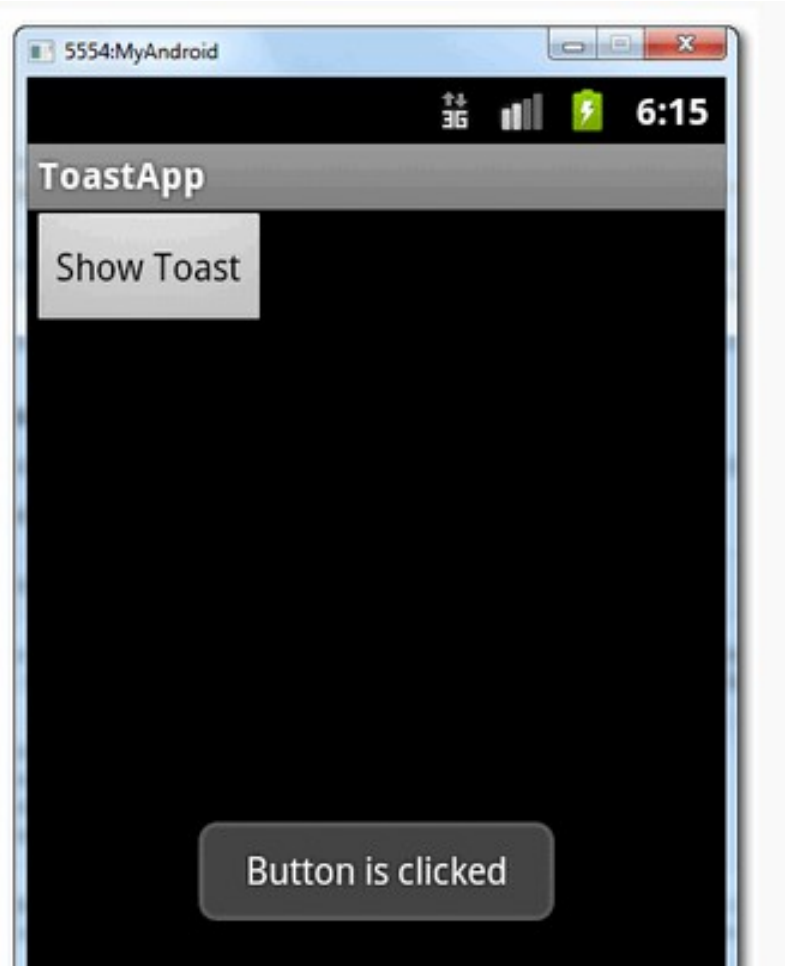


ScrollView

- ScrollView

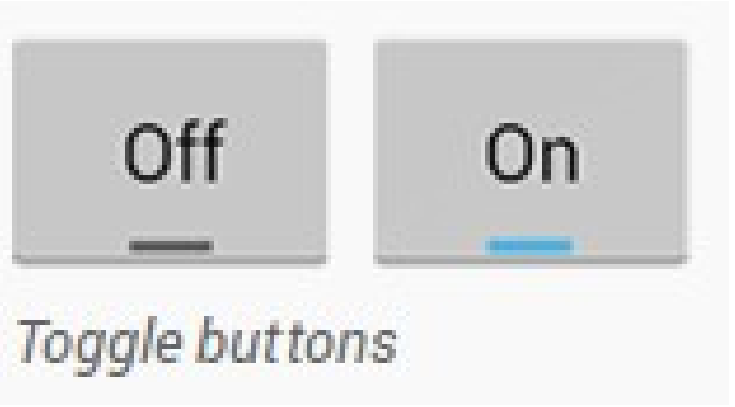
Toast

```
Toast.makeText(getApplicationContext(), "Button is clicked", Toast.LENGTH_LONG).show();
```



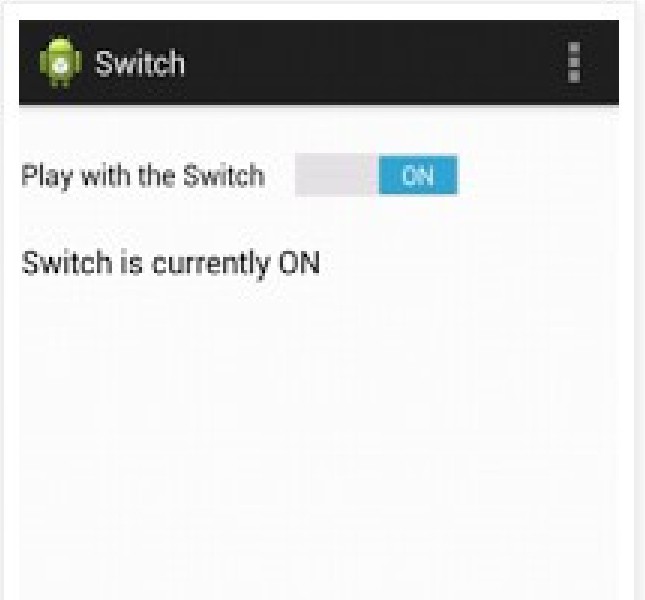
Toggle Button

```
<ToggleButton
android:id="@+id/togglebutton"
android:layout_width="wrap_con
tent"
android:layout_height="wrap_co
ntent"
android:textOn="Vibrate on"
android:textOff="Vibrate off"
android:onClick="onToggleClicke
d"/>
```

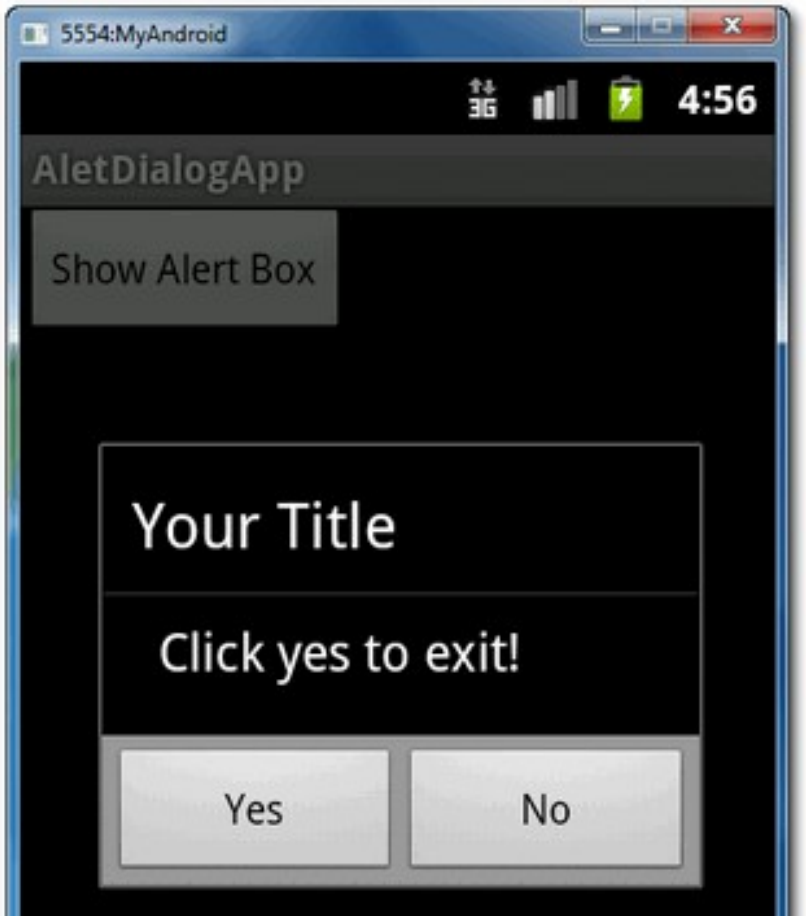


Switch Button

```
<Switch
android:id="@+id/mySwitch"
android:layout_width="wrap_cont
ent"
android:layout_height="wrap_con
tent"
android:layout_alignParentLeft="t
rue"
android:layout_alignParentTop="tr
ue"
android:layout_marginTop="20dp"
android:text="Play with the
Switch" />
```

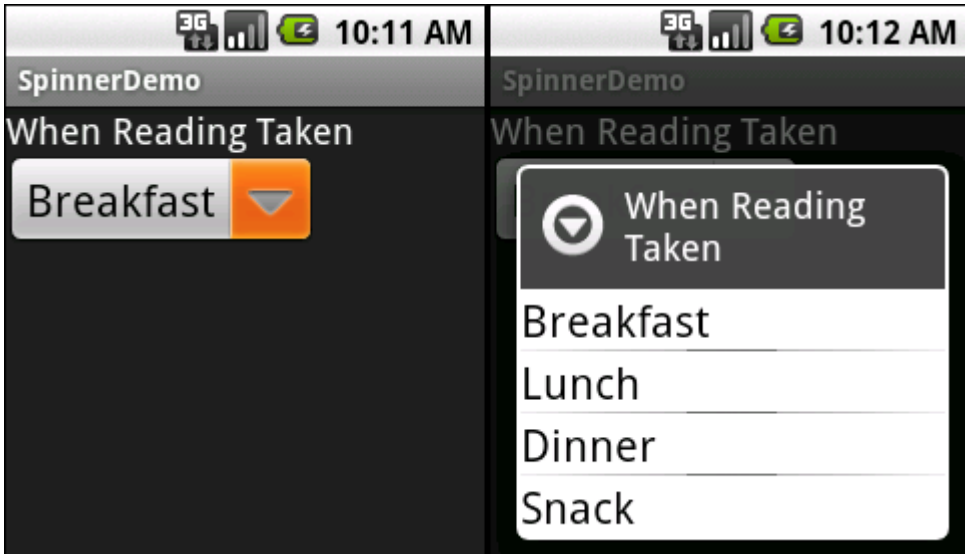


Alert Dialog



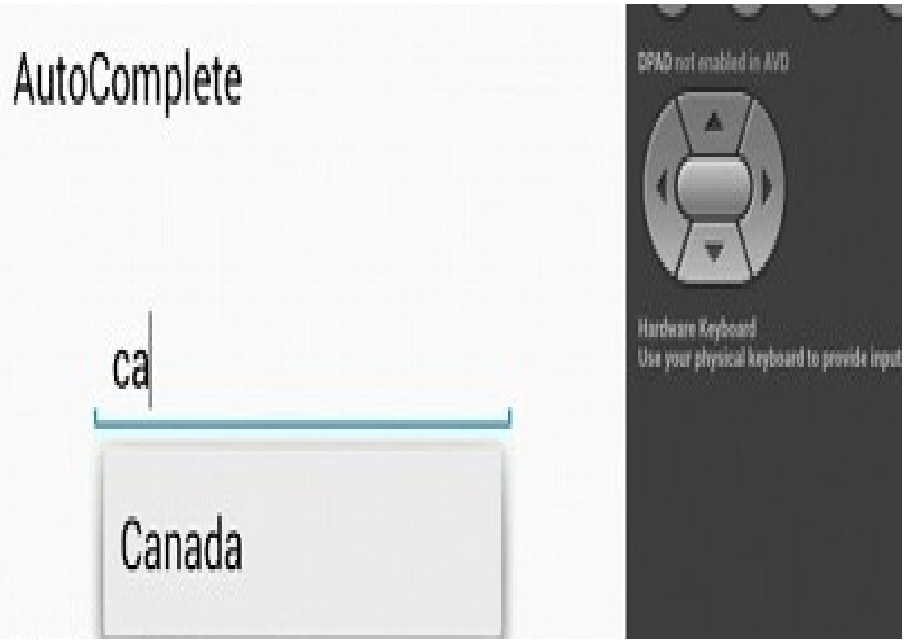
Spinner

```
<Spinner
    android:id="@+id/spi
    nner2"
    android:layout_width
    ="match_parent"
    android:layout_heigh
    t="wrap_content" />
```



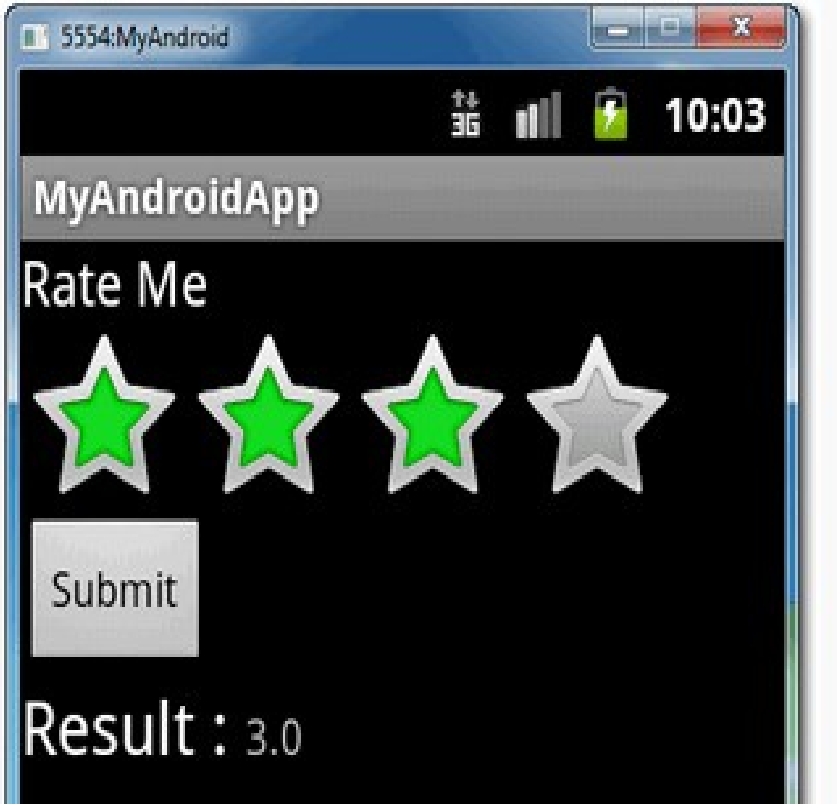
Auto Complete TextView

```
<AutoCompleteTextView
android:id="@+id/autoCompleteTextVie
w1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true"
android:layout_marginTop="65dp"
android:ems="10" >
<requestFocus />
</AutoCompleteTextView>
```



Rating Bar

```
<RatingBar
android:id="@+id/ratingBar"
android:layout_width="wrap_con
tent"
android:layout_height="wrap_co
ntent"
        android:numStars="4"
        android:stepSize="1.0"
        android:rating="2.0" />
```



Date Picker

```
<DatePicker
android:id="@+id/dpResult"
android:layout_width="wrap_con
tent"
android:layout_height="wrap_co
ntent" />
```

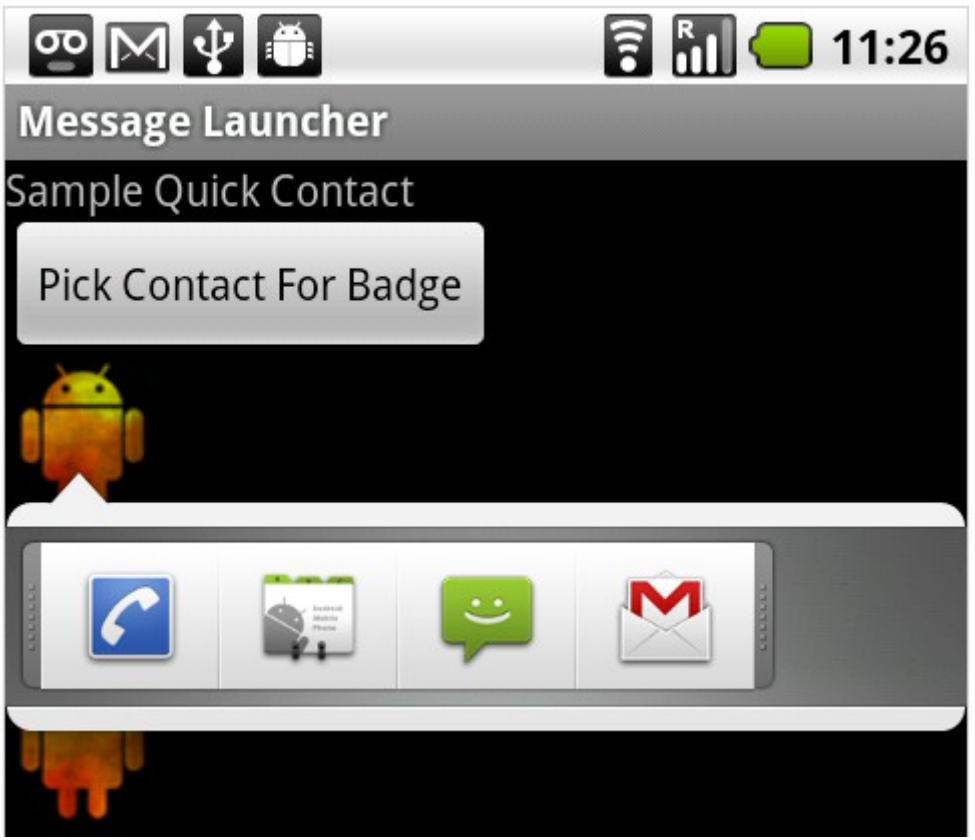
Time Picker

```
<TimePicker
android:id="@+id/timePicker1"

android:layout_width="wrap_con
tent"
android:layout_height="wrap_co
ntent" />
```

EMERTXE

Quick Contact Budge

```
QuickContactBadge badgeSmall
= (QuickContactBadge)
findViewById(R.id.badge_small);

badgeSmall.assignContactFromE
mail("any@gmail.com", true);

badgeSmall.setMode(ContactsCo
ntract.QuickContact.MODE_SMAL
L);
```



EMERTXE

Analog Clock and Digital Clock

- AnalogClock ac = (AnalogClock) findViewById(R.id.analogClock1) ;
- DigitalClock dc = (DigitalClock) findViewById(R.id.digitalClock1);

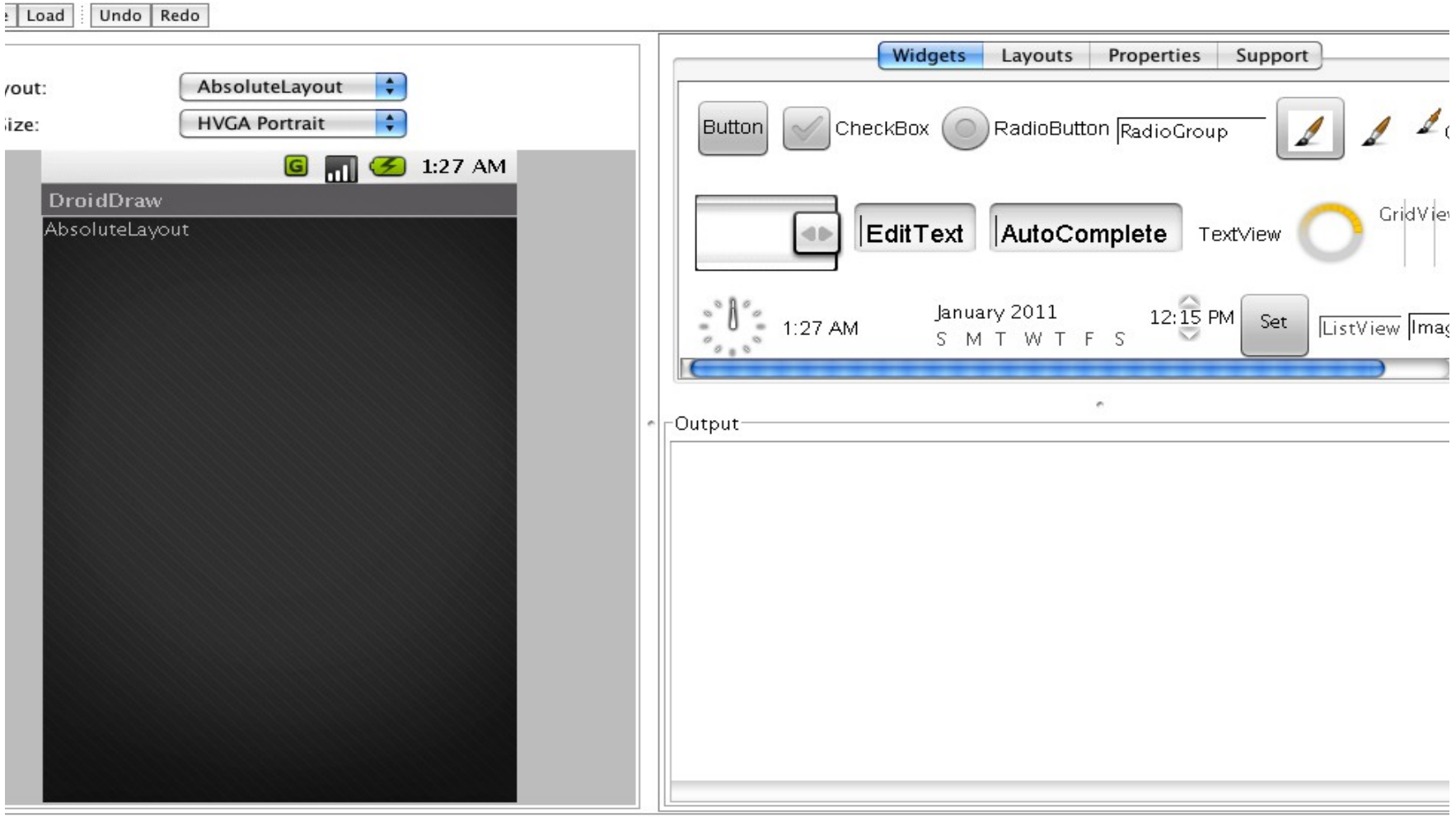
Two UI Approaches

Code	XML
You write Java code Similar to Swing or AWT	You write XML code Similar to HTML of a web page

You can mix and match both styles.

Declarative is preferred: easier and more tools

Draw Tool

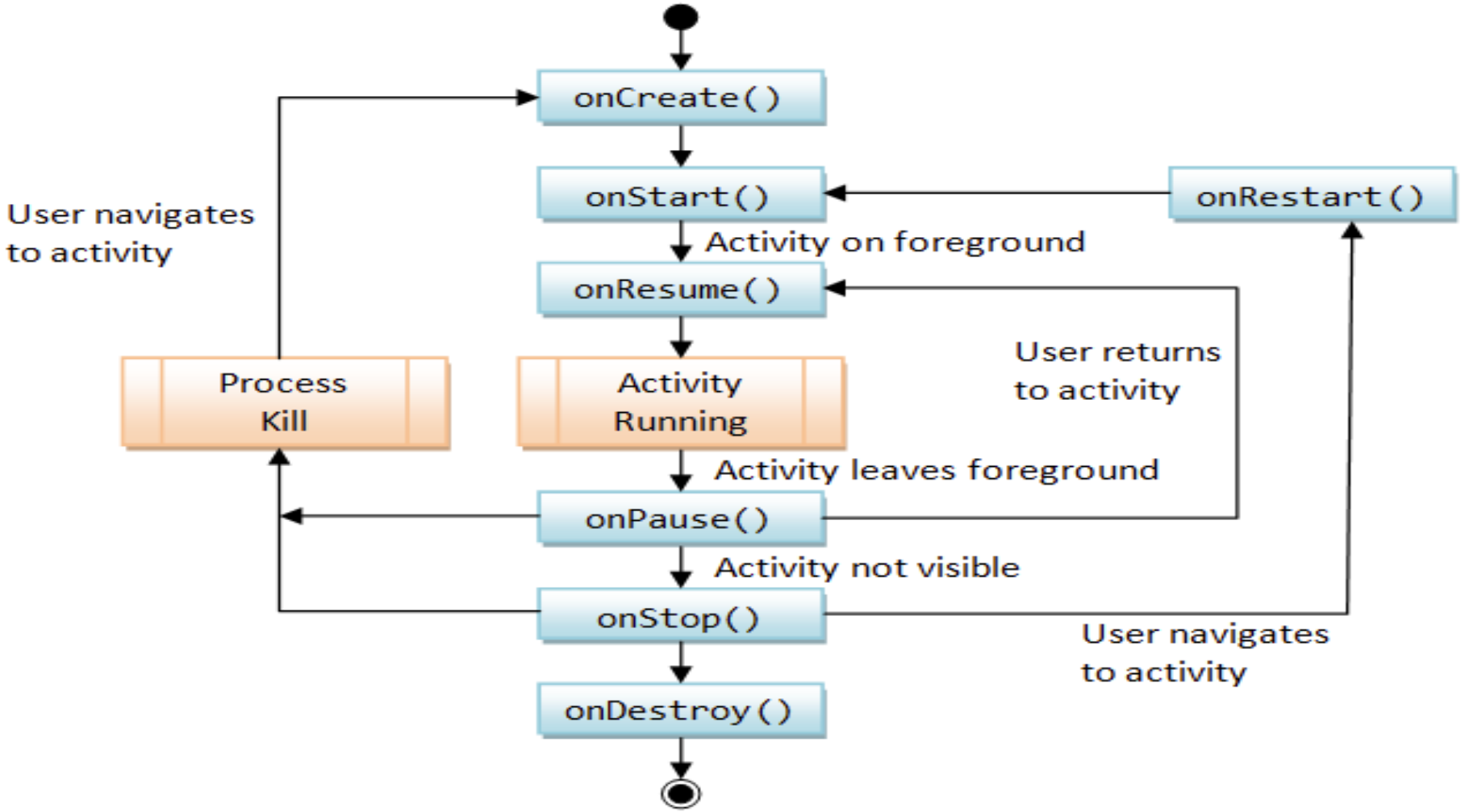

Load
Undo
Redo
AbsoluteLayout
HVGA Portrait
1:27 AM
DroidDraw
AbsoluteLayout
Widgets
Layouts
Properties
Support
Button
CheckBox
RadioButton
RadioGroup
EditText
AutoComplete
TextView
1:27 AM
January 2011
S M T W T F S
12:15 PM
Set
ListView
Output

Activity, Intent & Fragment

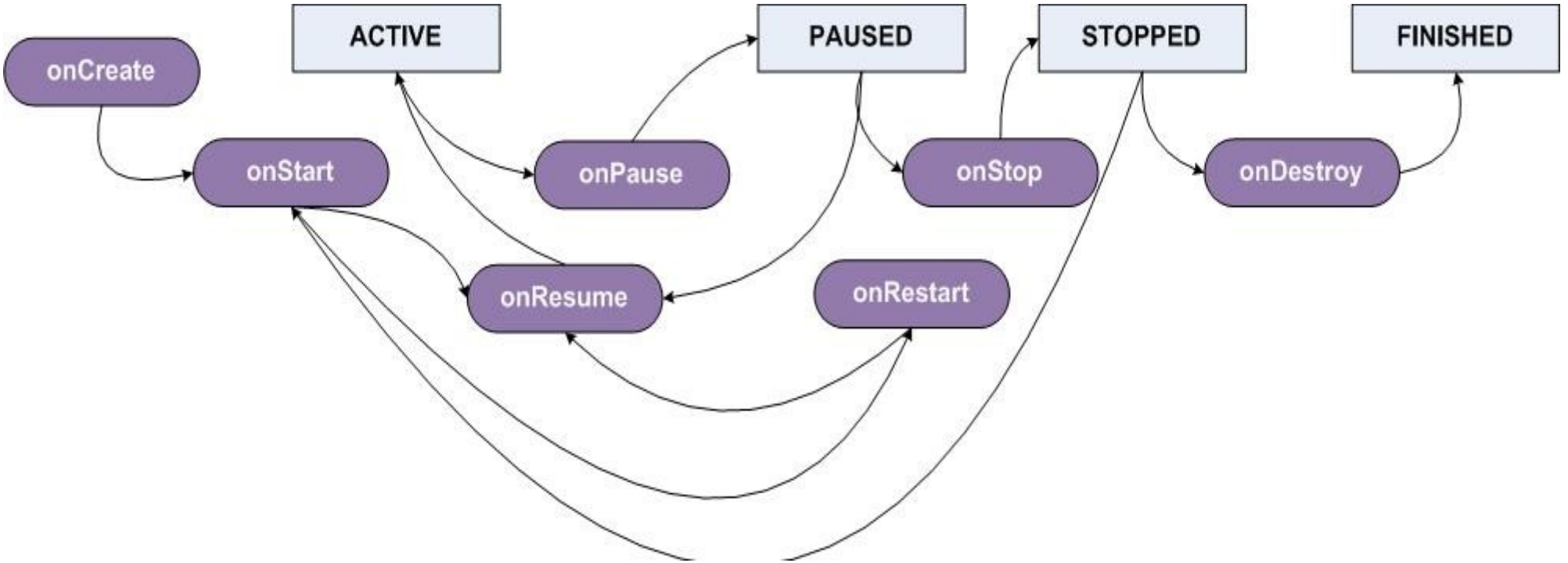
What is Activity

- An activity is a single, focused thing that the user can do.
- Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI.

Life Cycle of Activity



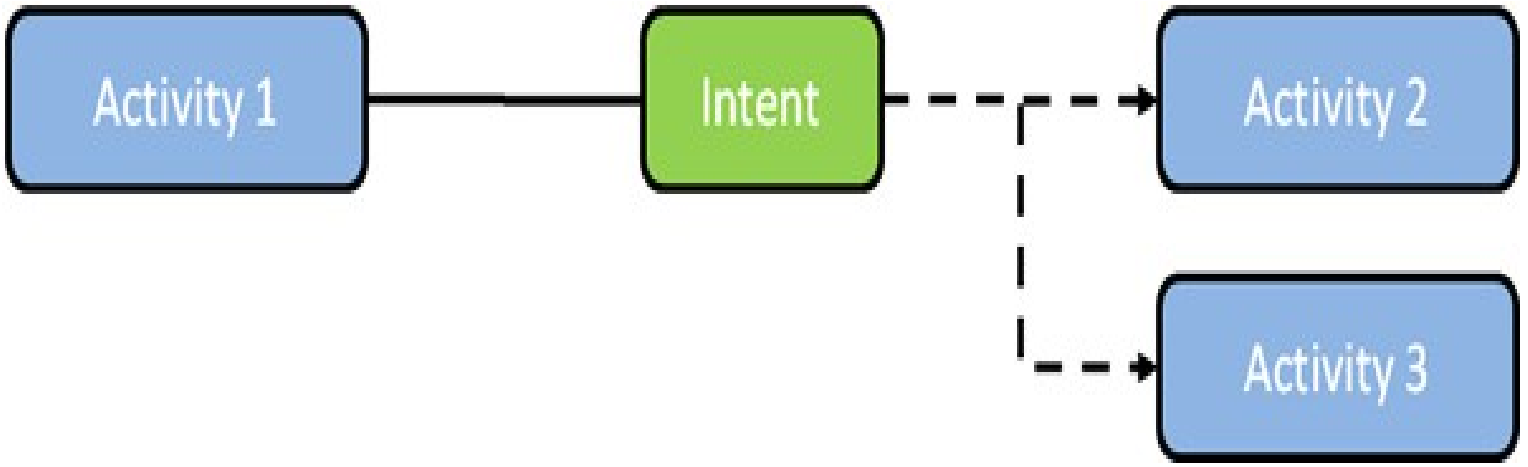
Android Activity Life Cycle Example



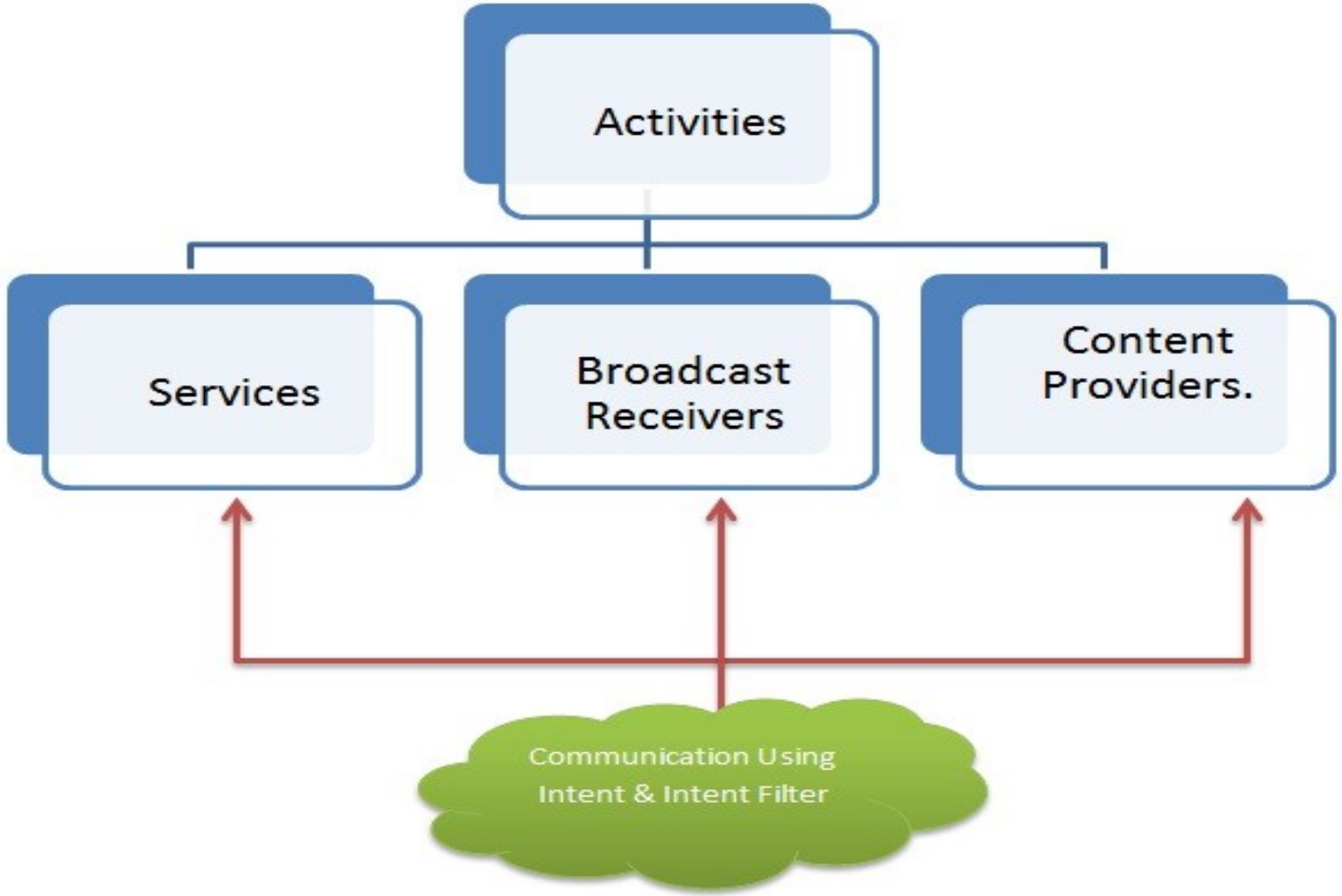
EMERTXE

Intent

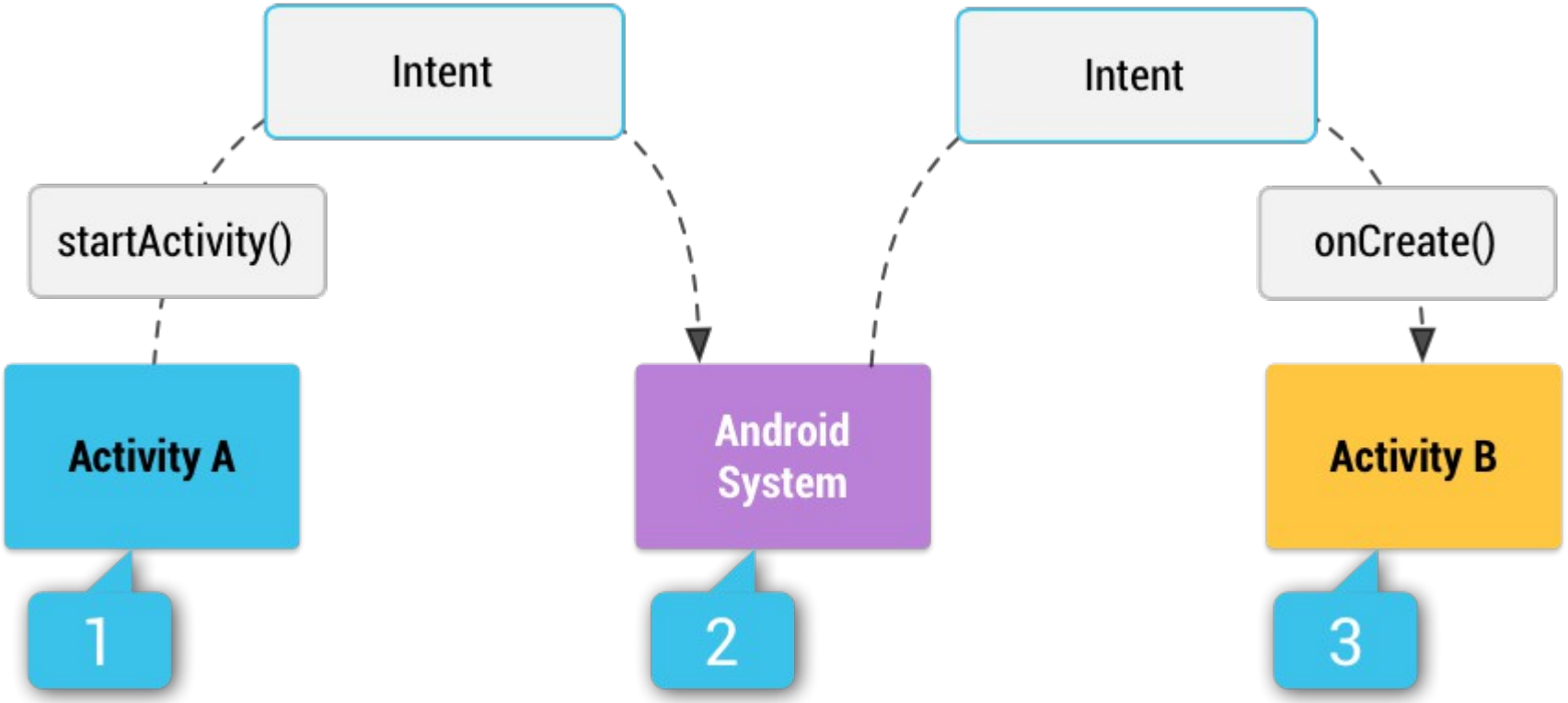
- Implicit Intent
- Explicit Intent



Intent

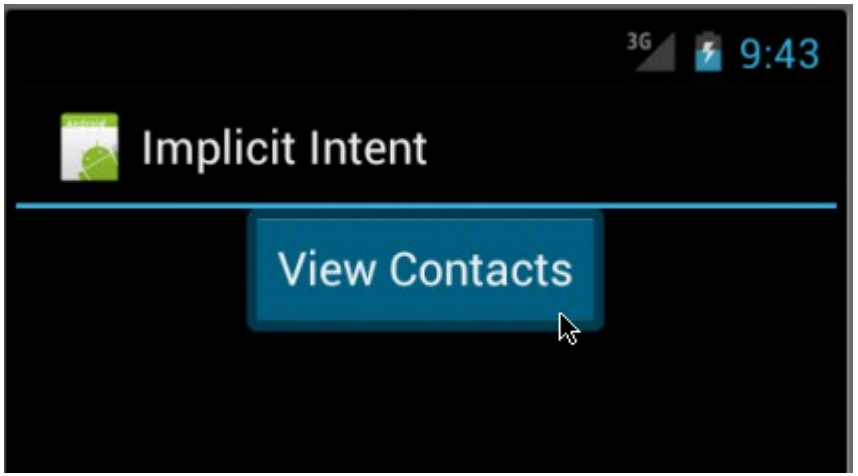


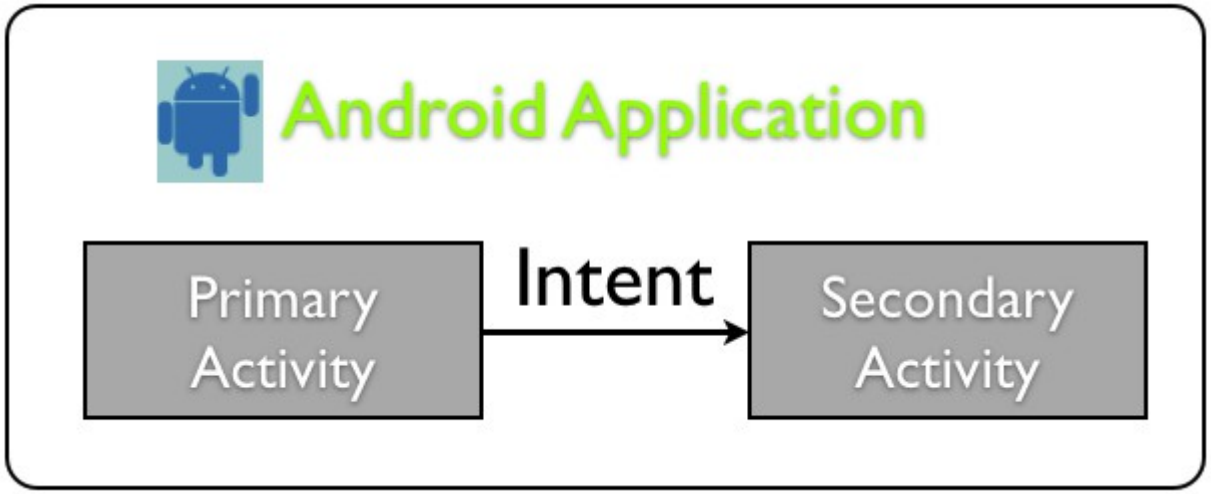
Intent



Intent

- Implicit Intent
- Explicit Intent





Intent Methods

- Intent intent=new Intent(Intent.ACTION_VIEW);

  intent.setData(Uri.parse("http://www.google.com"));

  startActivity(intent); //implicit

- Intent i = new Intent(getApplicationContext(),

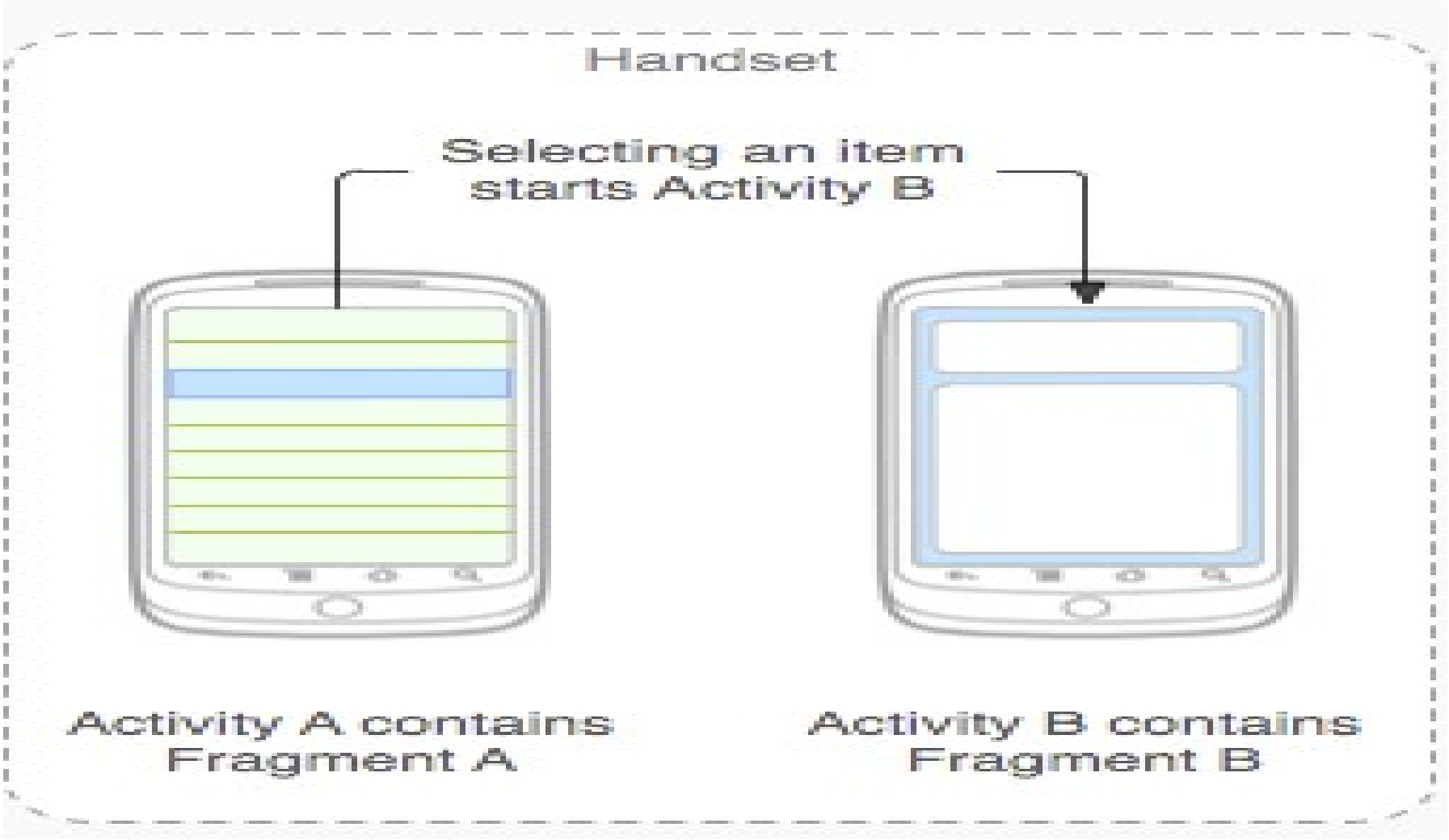
  ActivityTwo.class);

  startActivity(i);   // explicit

Fragments

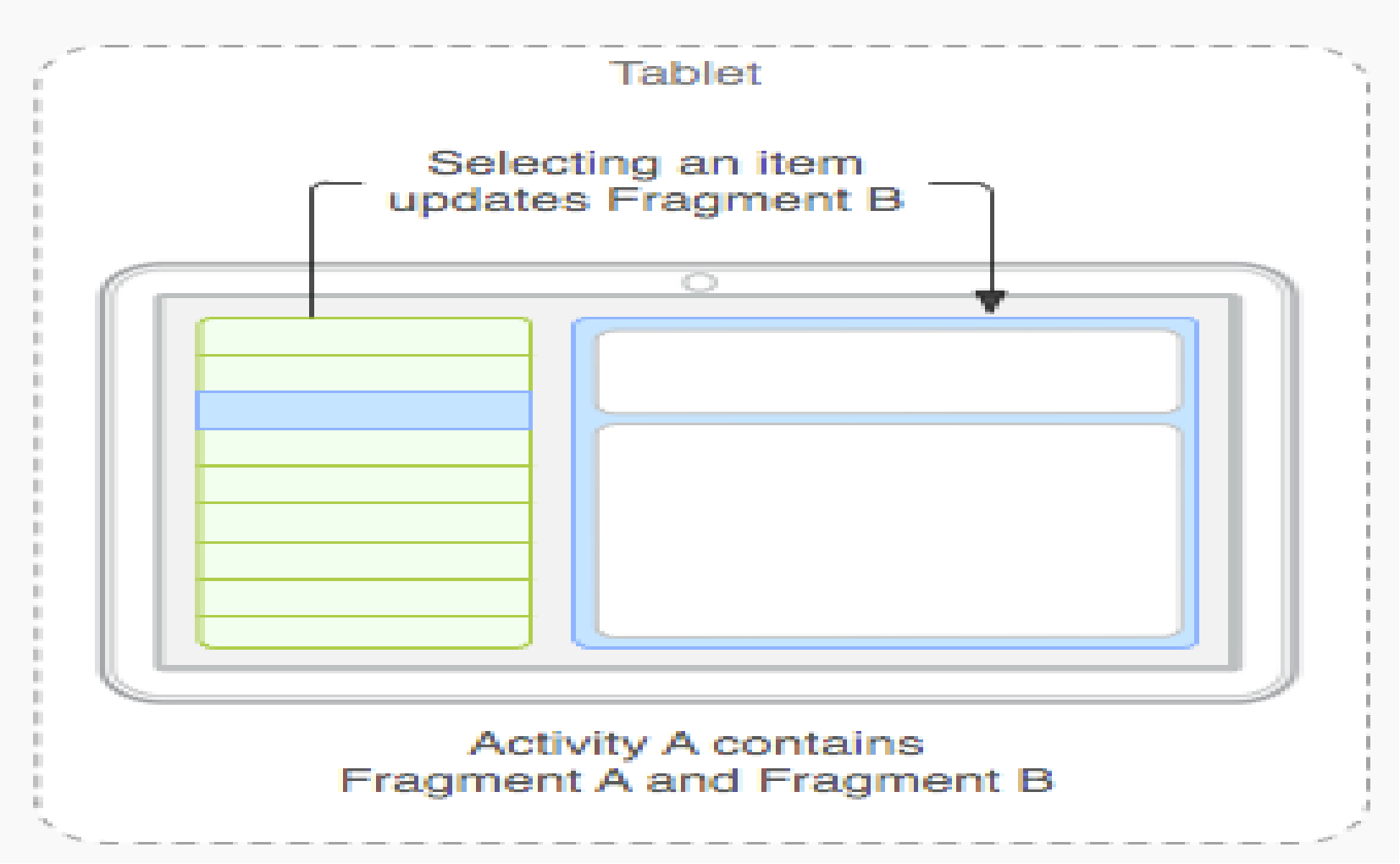
- A Fragment represents a behavior or a portion of user interface in an Activity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.

Fragments

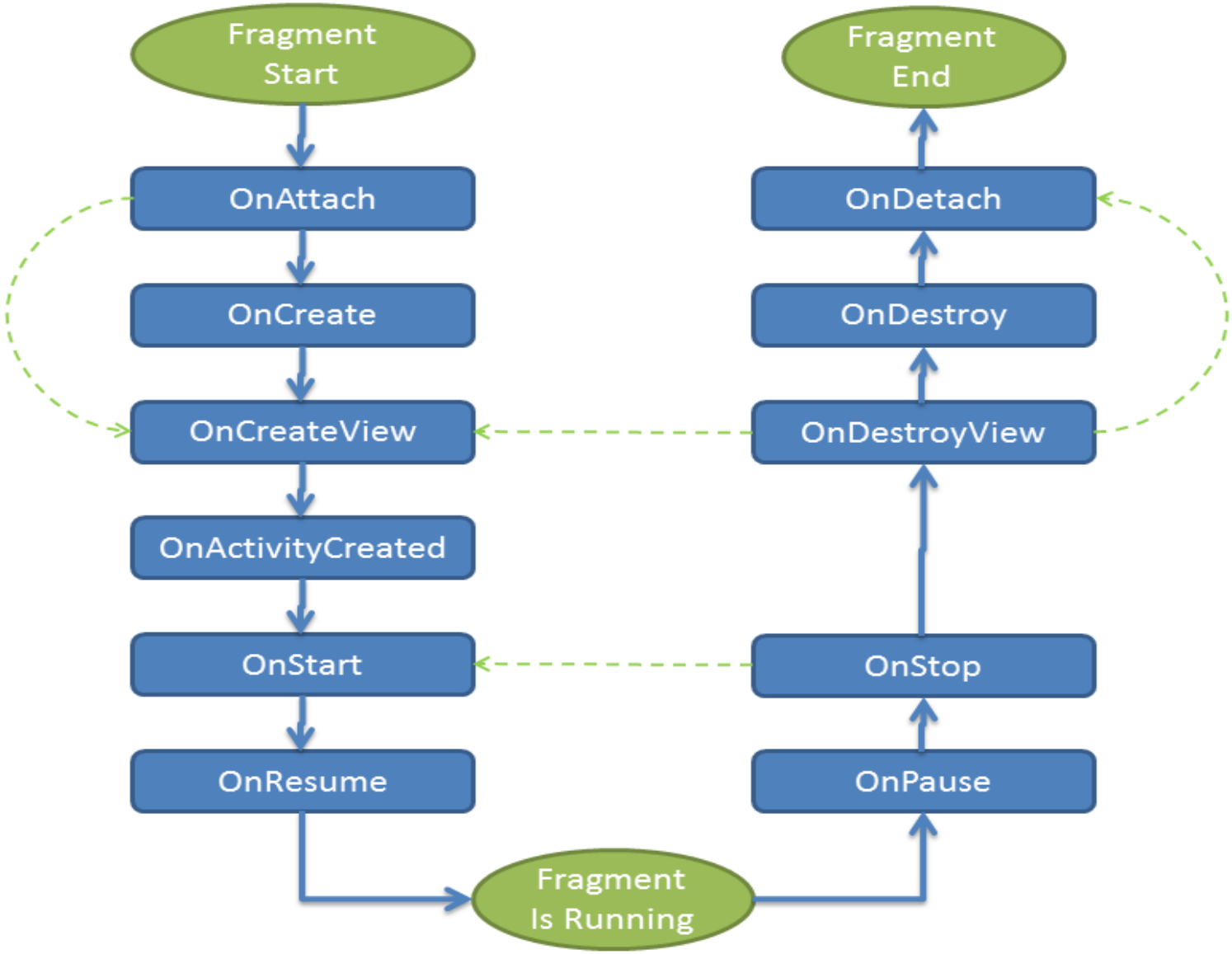


EMERTXE

Fragments

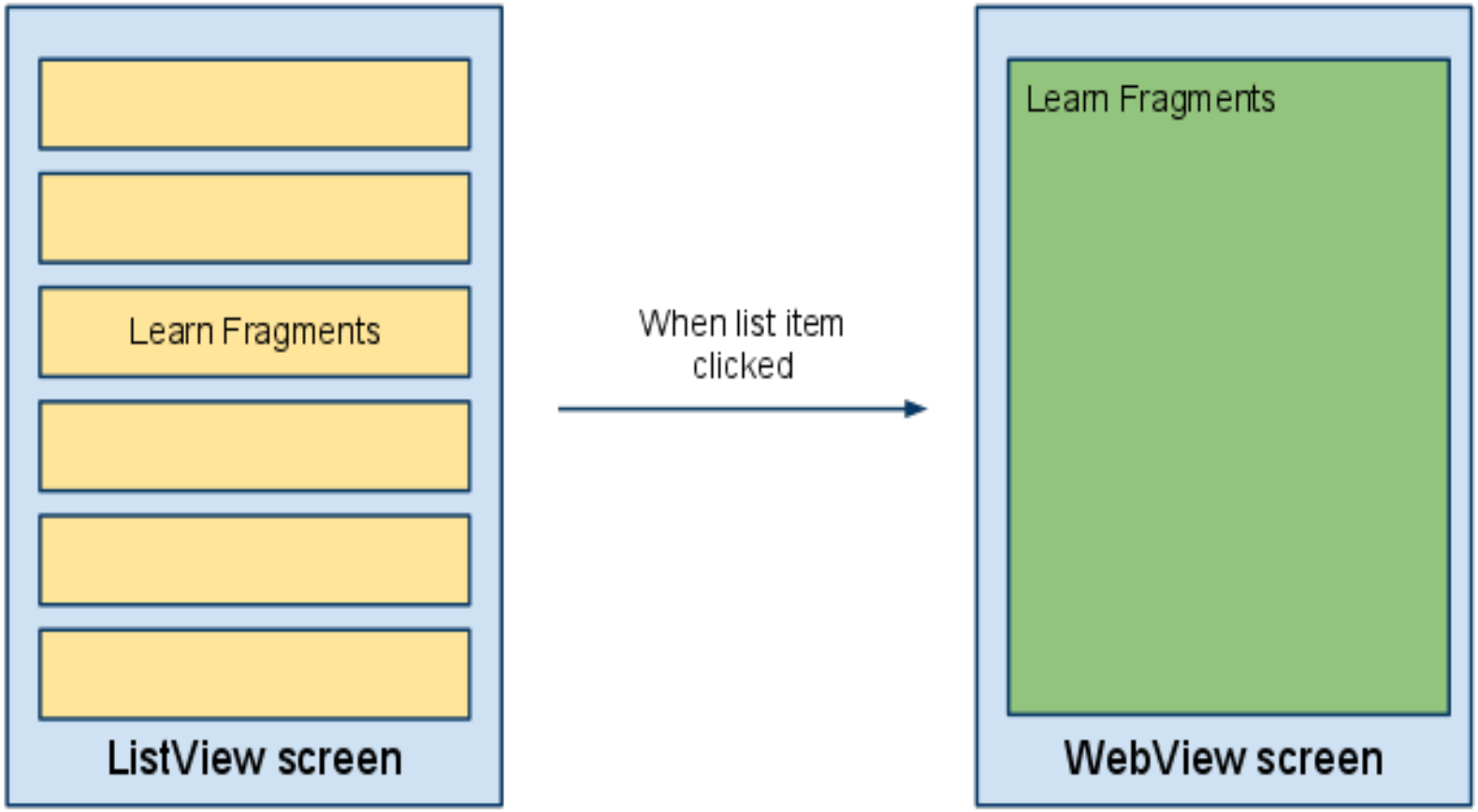


Fragment Life Cycle

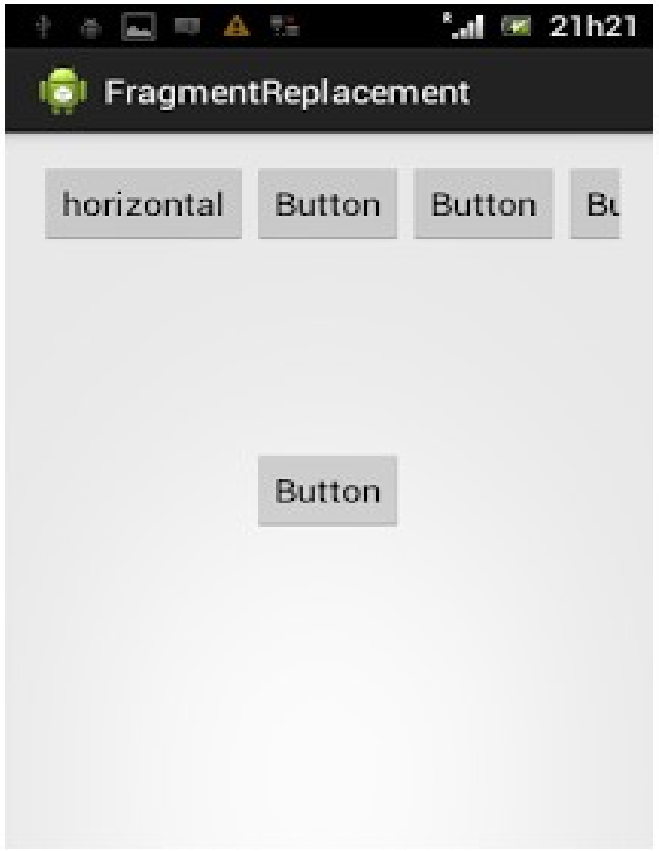


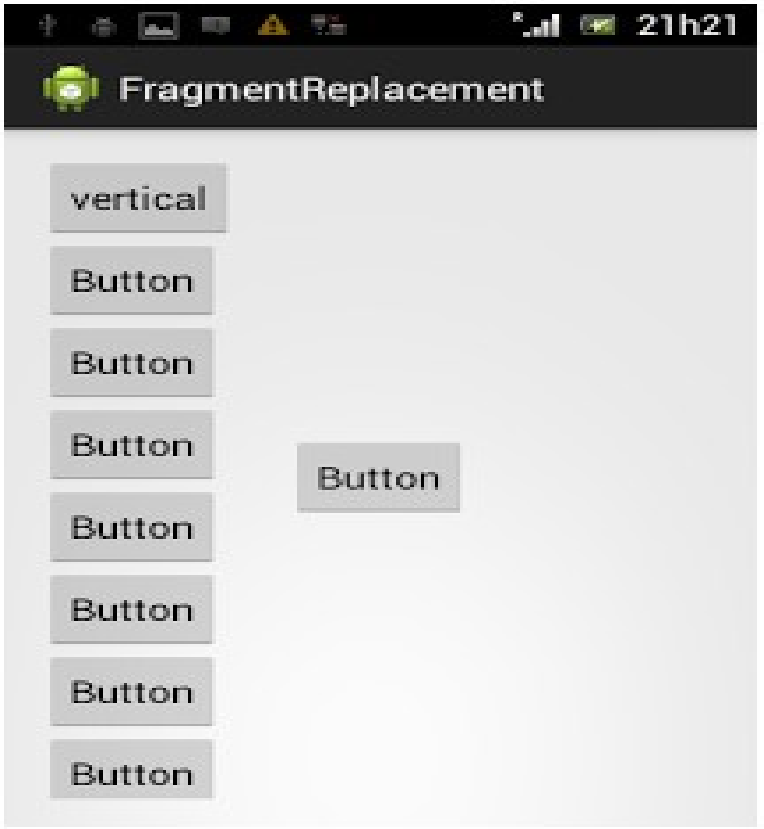
EMERTXE

Fragment Example



Dynamic Fragments





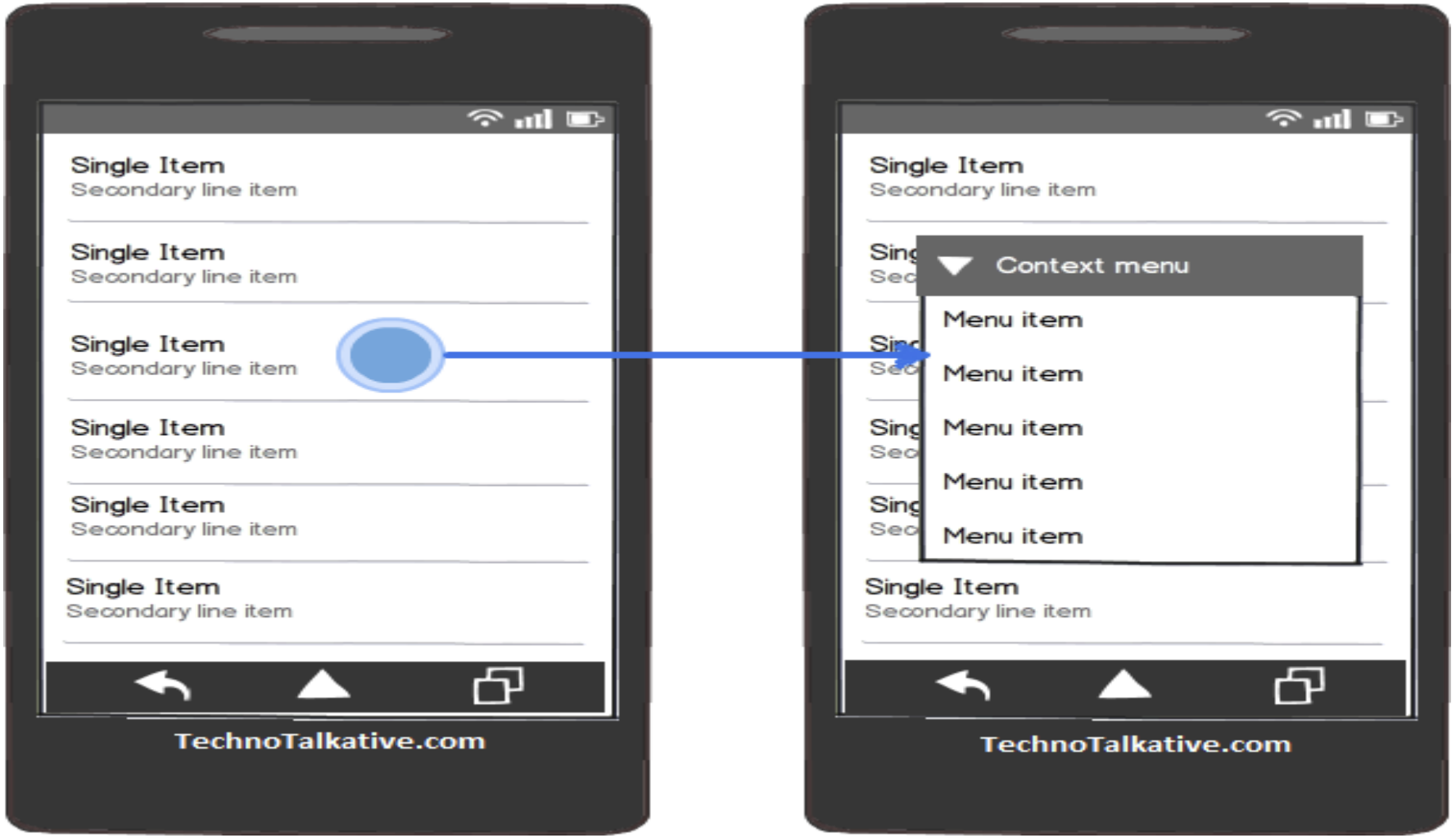
EMERTXE

Android Menu

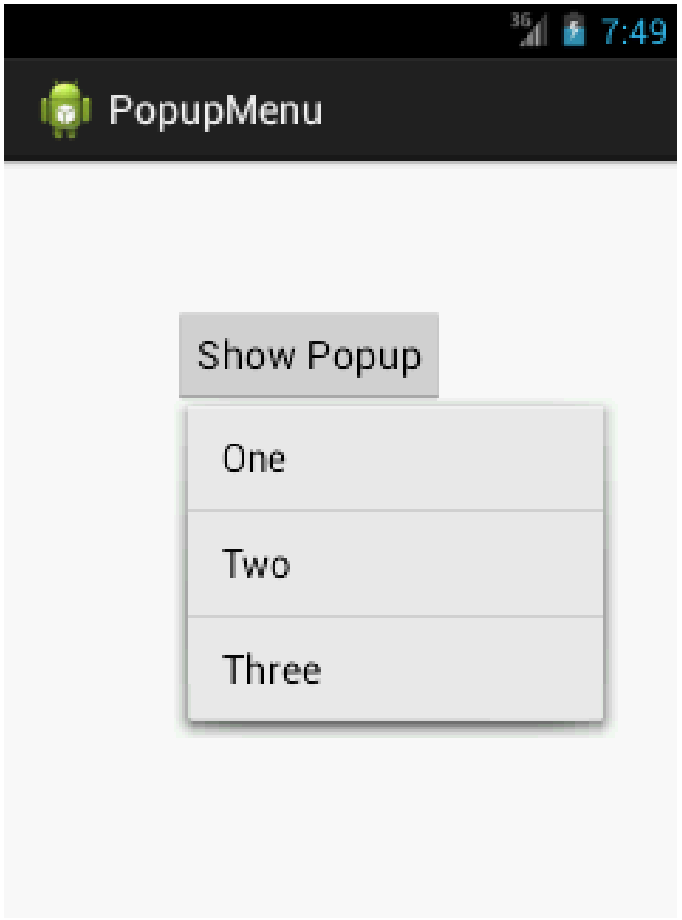
Option Menu

EMERTXE

Context Menu



Pop Up Menu



Layout Manager

Layouts

- Layout Managers are extensions of the ViewGroup class
- Control the position of child controls on a screen.
- Versatile layout classes available:
  - Relative Layout
  - Linear Layout
  - Table Layout
  - Grid Layout

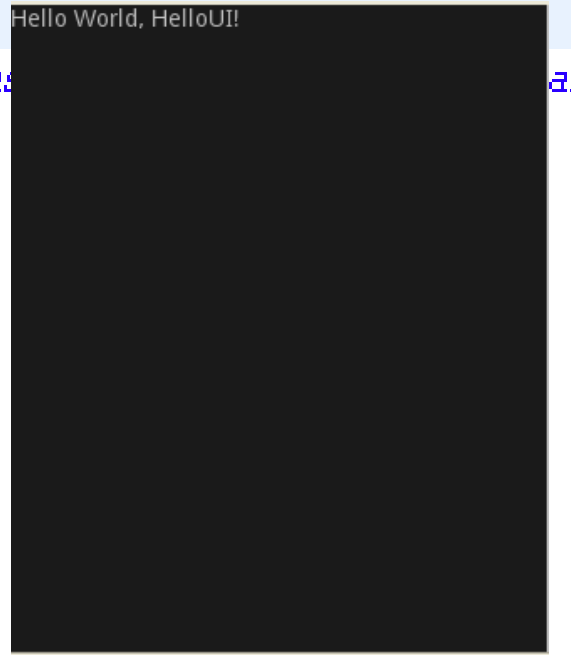
Activities

Android Activity

- Basic unit of an Android application is an Activity.
- May contain widgets like buttons, labels, text boxes, etc.

HelloUI.java | main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas                    android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```



Android Activity Flow

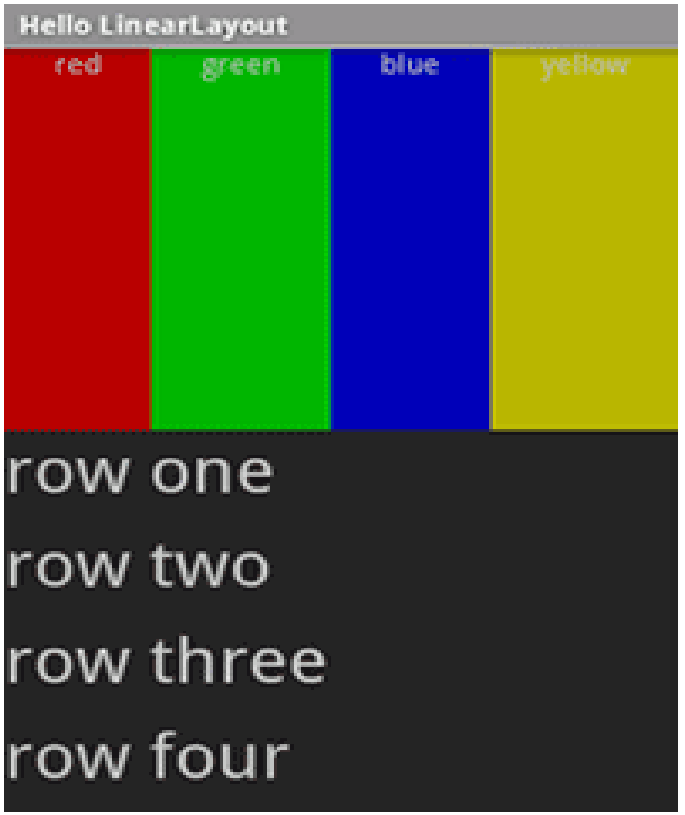
- Load the XML UI in the onCreate() using the setContentView()
- Each element in the XML file is compiled into its equivalent Android GUI class.
- UI of the Activity created when it is loaded.
- Easier to build your UI using a XML file.

Attributes

Attribute	Description
layout_width	Specifies the width of the View or ViewGroup
layout_height	Specifies the height of the View or ViewGroup
layout_marginTop	Specifies extra space on the top side of the View or ViewGroup
layout_marginBottom	Specifies extra space on the bottom side of the View or ViewGroup
layout_marginLeft	Specifies extra space on the left side of the View or ViewGroup
layout_marginRight	Specifies extra space on the right side of the View or ViewGroup
layout_gravity	Specifies how child Views are positioned
layout_weight	Specifies how much of the extra space in the layout to be allocated to the View
layout_x	Specifies the x-coordinate of the View or ViewGroup
layout_y	Specifies the y-coordinate of the View or ViewGroup

Linear Layouts

- The LinearLayout arranges views in a single column or single row.
- Child views can either be arranged vertically or horizontally.



EMERTXE

Linear Layout

Linear Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="horizontal" >

</LinearLayout>
```

Relative Layout

Relative Layout

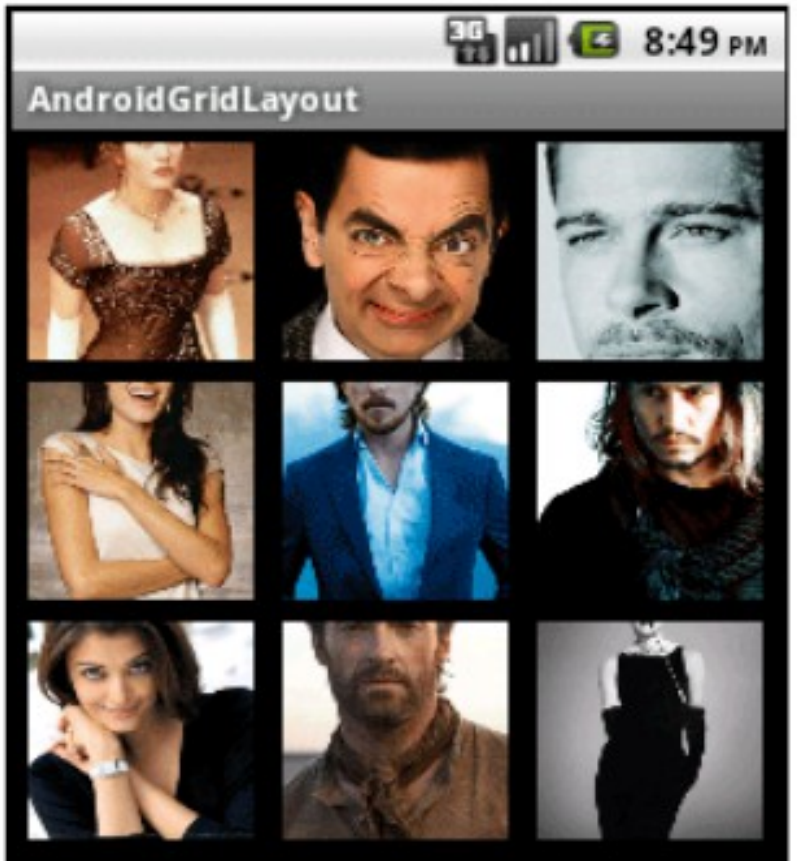
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >

</RelativeLayout>
```

Table Layout

```
<?xml version="1.0"
encoding="utf-8"?>

<TableLayout
xmlns:android="http://schemas.
android.com/apk/res/android"

 android:id="@+id/tableLayout1"
android:layout_width="fill_paren
t"
android:layout_height="fill_pare
nt" >
```

Grid View Layout

```
<GridView
xmlns:android="http://schemas.android.
com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```



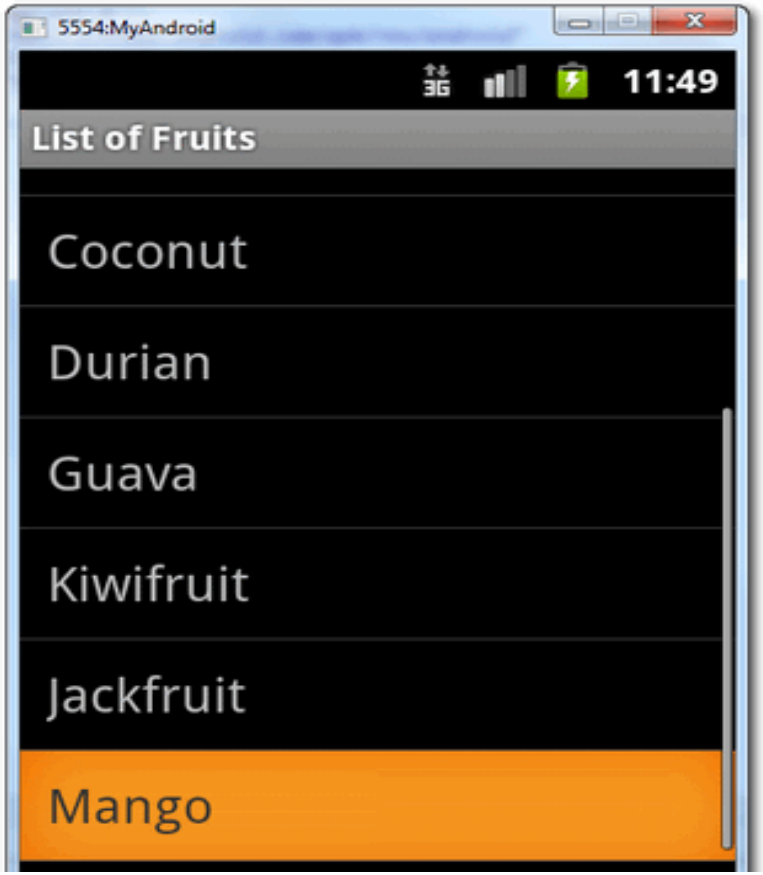
Adapter

Adaptor

- Array Adaptor
- ArrayList Adaptor
- Base Adaptor

Array Adaptor

```
ArrayAdapter<String> adapter =
new ArrayAdapter<String>(this,

R.layout.rowlayout, R.id.label,
values);

setListAdapter(adapter)
```




View

View

Grid View

Web View

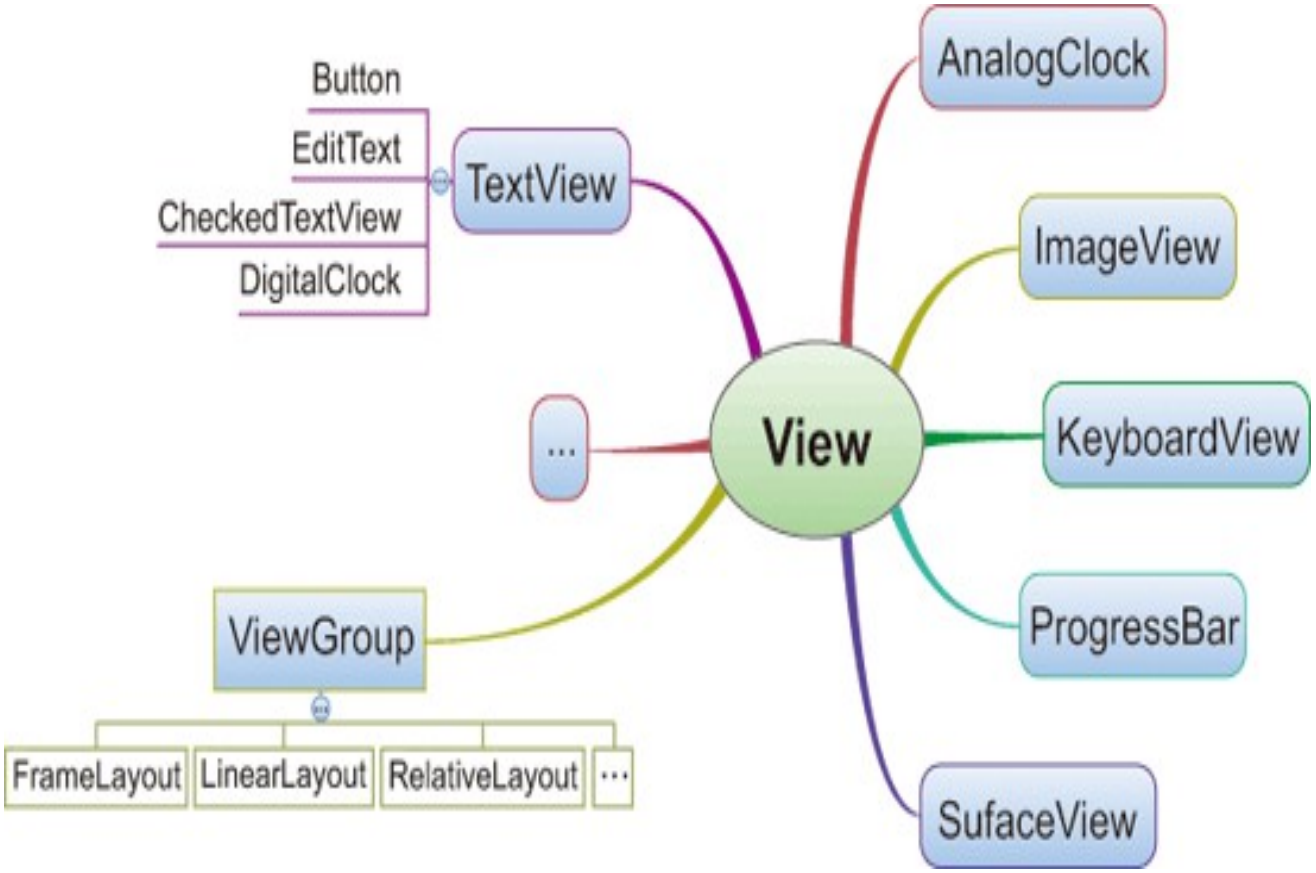
Scroll View

Search View

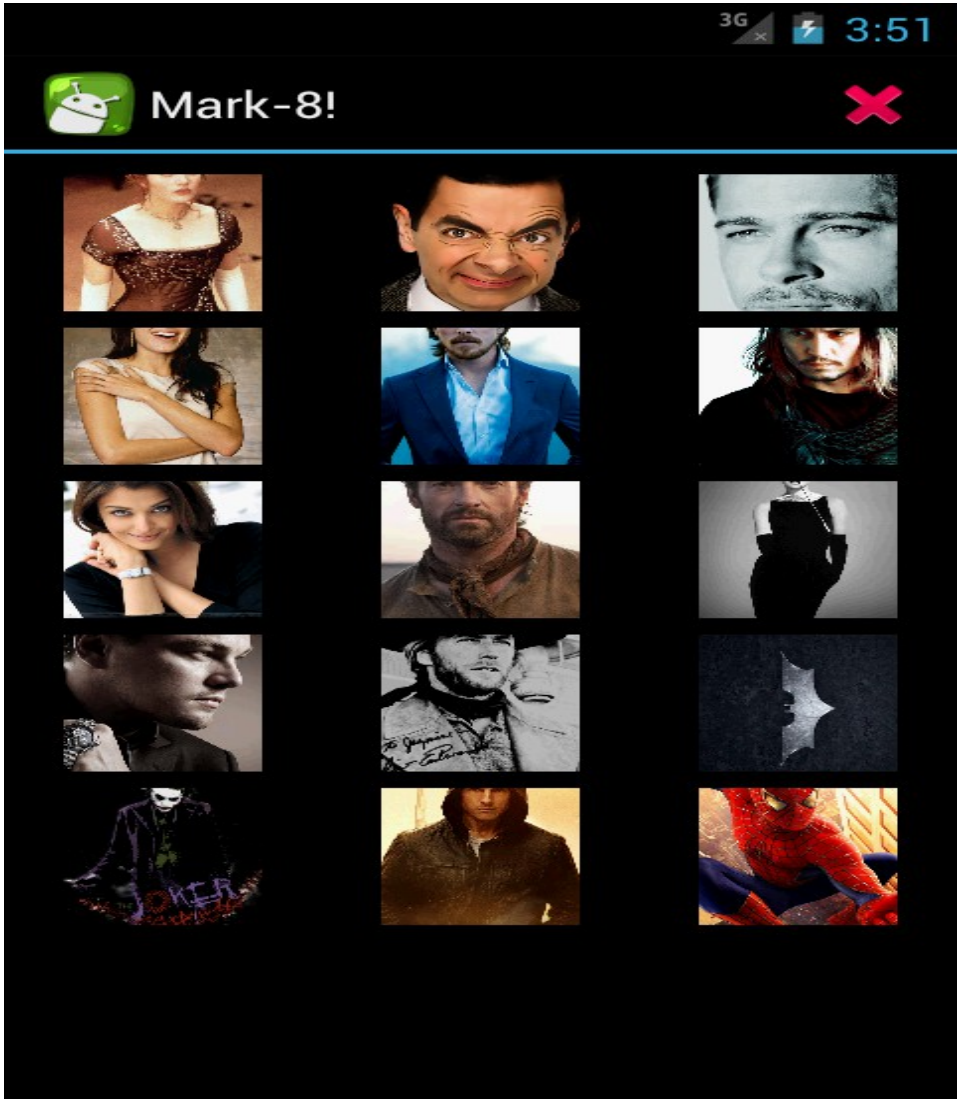
Tab Host

Dynamic List View

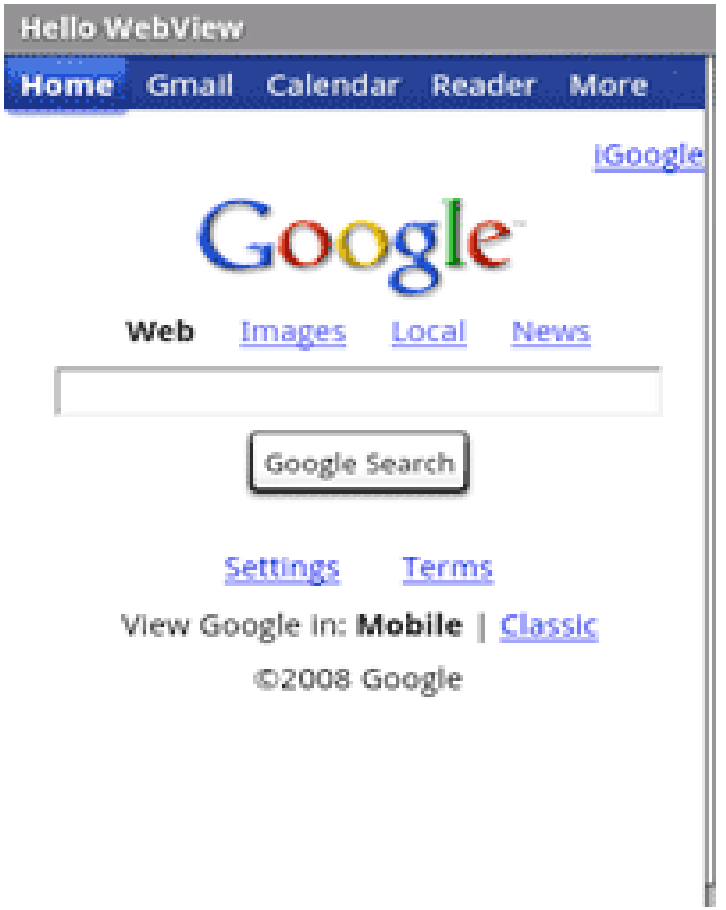
Expanded List View



Grid View

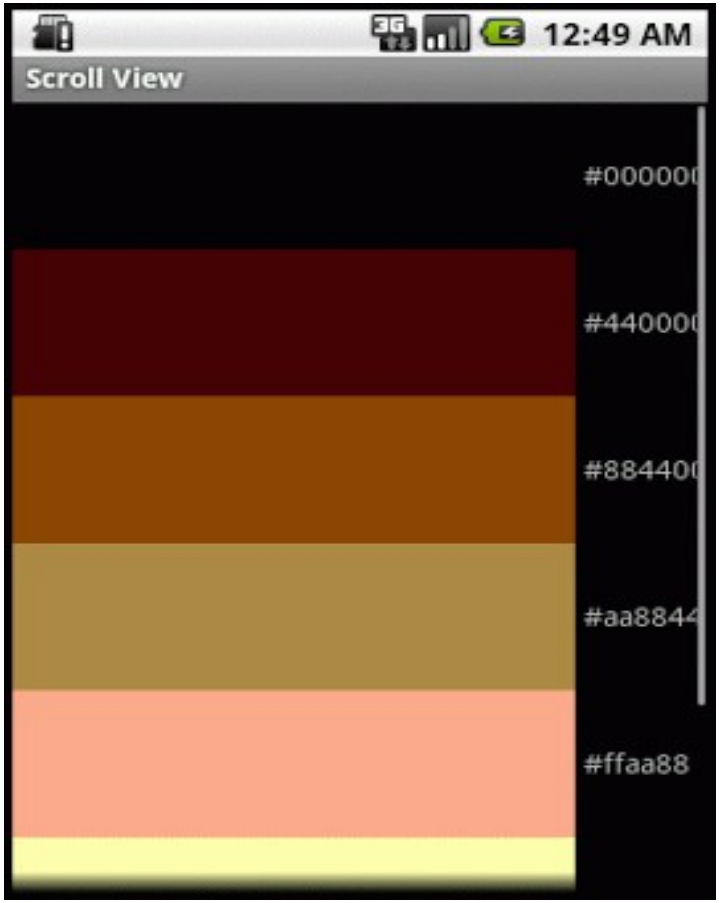


Web View

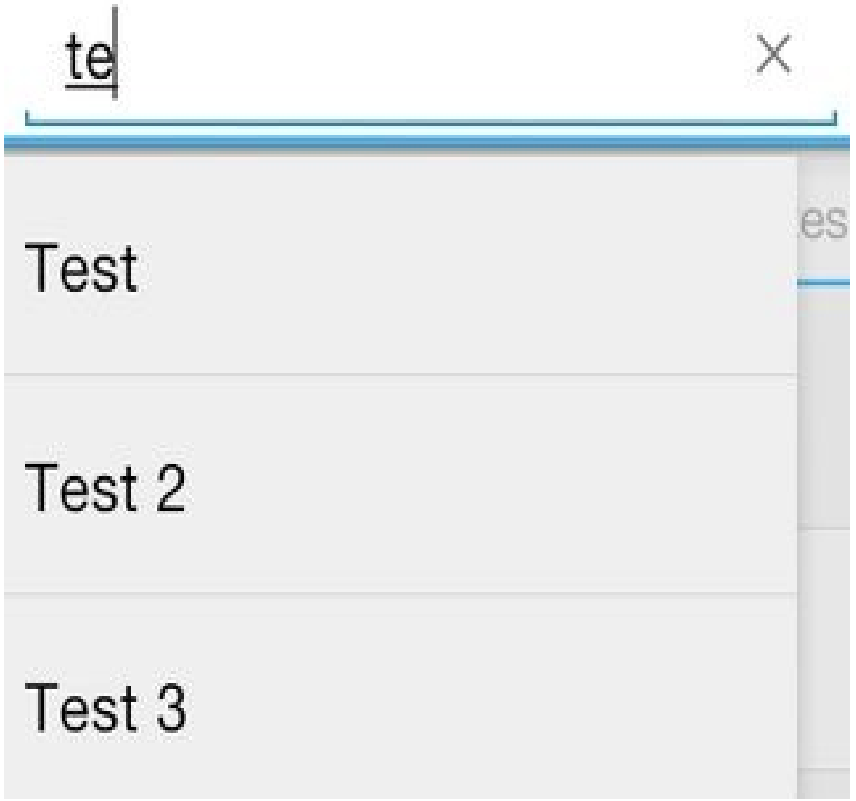


EMERTXE

Scroll View



Search View

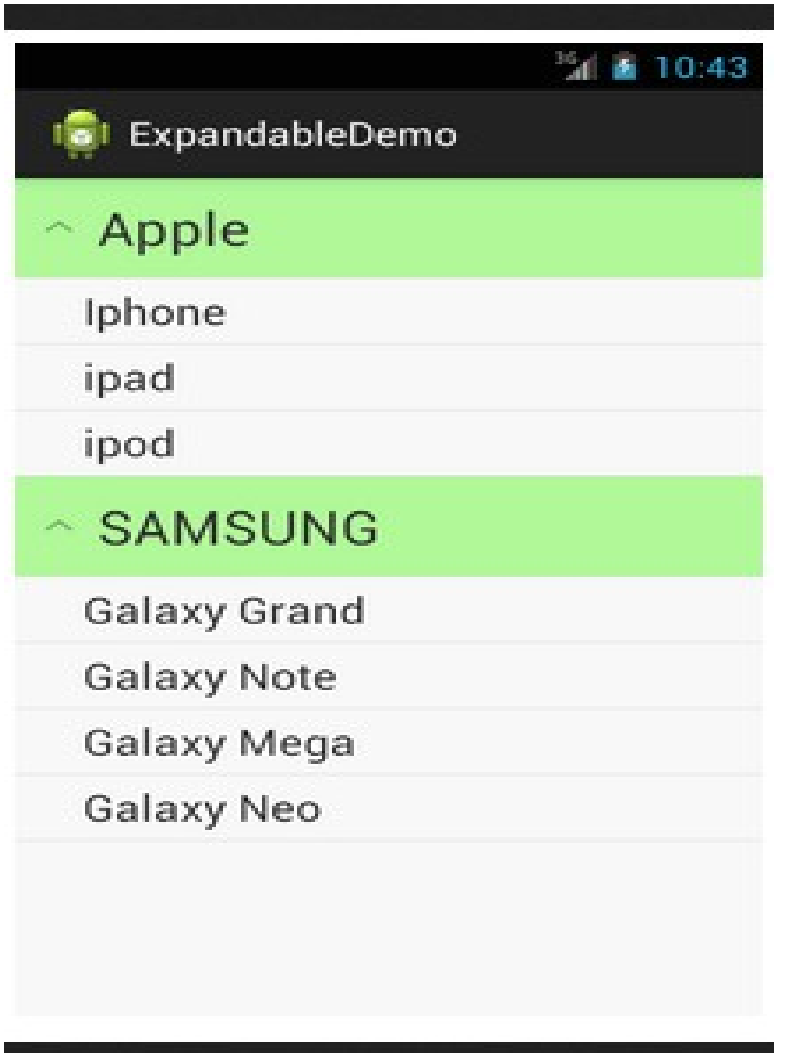


Tab Host

EMERTXE

Dynamic ListView

Expandable ListView



Services

Objectives

- What is a Service?
- Service lifecycles
- Permissions
- Local services
- Remote services

When a Service Used?

- Applications will need to run processes for a long time without any intervention from the user, or very rare interventions.
- These background processes need to keep running even when the phone is being used for other activities / tasks.
- Functionality that should run invisibly, without a user interface.

When a Service Used?

- Services gets higher priority than inactive Activities, so they’re less likely get to be killed.
- Services run without a dedicated GUI, but still execute in the main thread of the application’s process.
  - Updating Content Providers
  - Firing Intents
  - Triggering Notifications.

Two Types of Services

- Local service
  - Not accessible from other applications
  - Runs in the same process of the application that starts it

- Remote service
  - Accessible from other applications as well
  - Exposes to other applications through AIDL (s Definition Language)

Service and Notification

- Service is a long lived component and does not implement any user interface of its own.
  - Typically a user interacts with a service through an activity that has a UI.
- The service “notifies” the user in case of any need for user intervention.
  - Through the notification (such as status bar notification), the service can give a user interface for binding to the service or viewing the status of the service or any other similar interaction with the service.

Creating a Service

- To define a Service, create a new class that extends Service. You'll need to override onBind and onCreate,

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
public class MyService extends Service {
@Override
public void onCreate() {
// TODO: Actions to perform when service is created.
}
@Override
public IBinder onBind(Intent intent) {
// TODO: Replace with service binding implementation.
return null;
}
}
```

Controlling Services

- Override onStartCommand().
- Is called whenever the Service is started with a call to startService, so it may be executed several times within a Service's lifetime.
- onStartCommand it enables you to tell the system how to handle restarts if the Service is killed by the system prior to an explicit call to stopService or stopSelf.
- Services are launched on the main Application thread
- Any processing done in the onStartCommand handler will happen on the main GUI thread.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
// TODO Launch a background thread to do processing.
return Service.START_STICKY;
}
```

What Facilities Service Provide?

- A facility for the application to tell the system about something it wants to be doing in the background (even when the user is not directly interacting with the application).
- This corresponds to calls to Context.startService(),which ask the system to schedule work for the service, to be run until the service or someone else explicitly stop it.
- A facility for an application to expose some of its functionality to other applications.
- This corresponds to calls to Context.bindService(),which allows a long-standing connection to be made to the service in order to interact with it.

Binding to a Service

- Once a service has begun and is running in the background, activities can “bind” to the service.

- In fact, if we want to bind to a service that has not started, calling to bind could initiate the service.

What Facilities Service Provide?

- A facility for the application to tell the system about something it wants to be doing in the background (even when the user is not directly interacting with the application).

- This corresponds to calls to Context.startService(),which ask the system to schedule work for the service, to be run until the service or someone else explicitly stop it.

What Facilities Service Provide?

- A facility for an application to expose some of its functionality to other applications.
- This corresponds to calls to Context.bindService(),which allows a long-standing connection to be made to the service in order to interact with it.

Starting a Service

- If someone calls Context.startService() then the system will retrieve the service (creating it and calling its onCreate() method if needed) and then call its onStartCommand(Intent, int, int) method with the arguments supplied by the client.

- The service will at this point continue running until Context.stopService() or stopSelf() is called.

Modes of Operations

For started services, there are two additional major modes of operation they can decide to run in, depending on the value they return from onStartCommand():

- START_STICKY is used for services that are explicitly started and stopped as needed,

- START_NOT_STICKY or START_REDELIVER_INTENT are used for services that should only remain running while processing any commands sent to them.

Getting a Connection to a Service

- Clients can also use Context.bindService() to obtain a persistent connection to a service.

- This likewise creates the service if it is not already running (calling onCreate() while doing so), but does not call onStartCommand()

- The client will receive the IBinder object that the service returns from its onBind(Intent) method, allowing the client to then make calls back to the service.

Permissions

- Global access to a service can be enforced when it is declared in its manifest's <service> tag.
- By doing so, other applications will need to declare a corresponding <uses-permission> element in their own manifest to be able to start, stop, or bind to the service.
- In addition, a service can protect individual IPC calls into it with permissions, by calling the checkCallingPermission(String) method before executing the implementation of that call.

Local Service

- One of the most common uses of a Service is as a secondary component running alongside other parts of an application, in the same process as the rest of the components.
- All components of an .apk run in the same process unless explicitly stated otherwise, so this is a typical situation.
- When used in this way, by assuming the components are in the same process, you can greatly simplify the interaction between them.
- Clients of the service can simply cast the Ibinder they receive from it to a concrete class published by the service.

Remote Messenger Service

- If you need to be able to write a Service that can perform complicated communication with clients in remote processes (beyond simply the use of Context.startService to send commands to it), then you can use the Messenger class instead of writing full AIDL files.

Preferences

SharedPreferences

- The SharedPreferences class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types.

- You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).

SharedPreferences

- To get a SharedPreferences object for your application, use one of two methods:
  - **getSharedPreferences()** - Use this if you need multiple preferences files identified by name, which you specify with the first parameter.
  - **getPreferences()** - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

SharedPreferences

- To write values:
  - Call edit() to get a SharedPreferences.Editor.
    Add values with methods such as putBoolean() and putString().
    Commit the new values with commit()

Other Useful Methods

- getFilesDir()
- Gets the absolute path to the filesystem directory where your internal files are saved.
- getDir()
- Creates (or opens an existing) directory within your internal storage space.
- deleteFile()
- Deletes a file saved on the internal storage.
- fileList()
- Returns an array of files currently saved by your application.

Data Storage

Agenda

- Android Database
- SQLite Database
- Content Providers
- Cursors and Content Values
- SQLiteOpenHelper
- Querying a Database
- Add, Update, Delete records
- Access DB through command line
- SQLite database file

Android Databases

Structured data persistence in Android :

- **SQLite Databases :** Offers the SQLite relational database library
- **Content Providers:** offer a generic, well-defined interface for using and sharing data between applications

SQLite Databases

- Independent relational databases for applications.
- Store and manage complex, structured application data.
- DB stored in the /data/data/<package_name>/databases folder on your device (or emulator).

SQLite Databases

- All databases are private. Standard database best practices apply in Android.
- Normalize your data to reduce redundancy.

SQLite

- SQLite has a reputation for being extremely reliable and is the database system of choice for many :
  - Consumer electronic devices
  - Including several MP3 players
  - iPhone, and the iPod Touch.

SQLite

- *SQLite is a well regarded relational database management system (RDBMS). It is*:
  - Open-source
  - Standards-compliant
  - Lightweight
  - Single-tier
  - C library
- SQLite database is an integrated part of the application.

SQLite

- SQLite has a reputation for being extremely reliable and is the database system of choice for many :
  - Consumer electronic devices
  - Including several MP3 players
  - iPhone, and the iPod Touch.

Cursors And Content Values

- ContentValues used to insert new rows into tables.

- Queries in Android returned as Cursor objects.

- The startManagingCursor method integrates the Cursor's lifetime into the calling Activity's.When you've finished with the Cursor, call stopManagingCursor to do just that.

Cursors And Content Values

Navigation functions including:

- moveToFirst Moves the cursor to the first row in the query result
- moveToNext Moves the cursor to the next row
- moveToPrevious Moves the cursor to the previous row
- getCount Returns the number of rows in the result set
- getColumnName Returns the name of the specified column index
- moveToPosition Moves the Cursor to the specified row
- getPosition Returns the current Cursor position

SQLiteOpenHelper

- SQLiteOpenHelper is an abstract class :
  - Used for creating,
  - Opening, and upgrading databases.
- onCreate, and onUpgrade - creation of a new database.
- Create a new instance
- Passing in the context
- Database name
- Current version
- Call getReadableDatabase or getWritableDatabase to open and return a readable/writable instance of the underlying database.

SQLiteOpenHelper

```
dbHelper = new myDbHelper(context, DATABASE_NAME, null,
DATABASE_VERSION);

SQLiteDatabase db;

try
{
      db = dbHelper.getWritableDatabase();
}
catch (SQLiteException ex)
{
      db = dbHelper.getReadableDatabase();
}
```

Opening And Creating Databases

Opening and creating databases without SQLiteHelper:

- Create and open databases by using the openOrCreateData base method from the application Context.
- Setting up a database is a two-step process:
  - call openOrCreateDatabase to create the new database.
  - call execSQL to run the SQL commands

Opening And Creating Databases Without SQLiteHelper

```
private static final String DATABASE_NAME = "myDatabase.db";
private static final String DATABASE_TABLE = "mainTable";
private static final String DATABASE_CREATE =
"create table " + DATABASE_TABLE + " ( _id integer primary key
autoincrement," +
"column_one text not null);";

SQLiteDatabase myDatabase;

private void createDatabase()
{
      myDatabase = openOrCreateDatabase(DATABASE_NAME,
     Context.MODE_PRIVATE, null);

      myDatabase.execSQL(DATABASE_CREATE);
}
```

Querying A Database

- Database query returns a Cursor.

- To execute needs :
  - Optional Boolean – if result set should contain only unique values.
  - The name of the table to query.
  - Lists the columns to include in the result set.
  - “where” clause
  - Defines the rows to be returned.
  - A “group by” clause that defines how the resulting rows will be grouped.
  - A “having” filter that defines which row groups to include if you specified a *group by clause*.
  - Describes the order of the returned rows.
  - Limit the number of returned rows.

Querying A Database

```
// Return all rows for columns one and three, no duplicates

String[] result_columns = new String[] {KEY_ID, KEY_COL1, KEY_COL3};

Cursor allRows = myDatabase.query(true, DATABASE_TABLE, result_columns,
null, null, null, null, null, null);
```

```
// Return all columns for rows where column 3 equals a set value
// and the rows are ordered by column 5.

String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor myResult = myDatabase.query(DATABASE_TABLE, null, where,
null, null, null, order);
```

Extracting Results From Cursor

- First use the moveTo<location> methods described earlier to position the cursor at the correct row of the result Cursor.

```
String columnValue = myResult.getString(columnIndex);
```

Extracting Results From Cursor

```
int GOLD_HOARDED_COLUMN = 2;
Cursor myGold = myDatabase.query("GoldHoards", null, null, null, null, null, null);
float totalHoard = 0f;

// Make sure there is at least one row.
if (myGold.moveToFirst())
 {

        // Iterate over each cursor.
        do
         {
              float hoard = myGold.getFloat(GOLD_HOARDED_COLUMN);
              totalHoard += hoard;
          } while(myGold.moveToNext());
}
float averageHoard = totalHoard / myGold.getCount();
```

Inserting New Rows

- Construct a ContentValues object
- Put methods to provide a value for each column.

```
// Create a new row of values to insert.
ContentValues newValues = new ContentValues();

// Assign values for each row.
newValues.put(COLUMN_NAME, newValue);
[ ... Repeat for each column ... ]

// Insert the row into your table
myDatabase.insert(DATABASE_TABLE,  newValues);
```

Updating A Row

```
// Define the updated row content.
ContentValues updatedValues = new ContentValues();

// Assign values for each row.
newValues.put(COLUMN_NAME, newValue);
[ ... Repeat for each column ... ]

String where = KEY_ID + "=" + rowId;

// Update the row with the specified index with the new
values.
myDatabase.update(DATABASE_TABLE, newValues, where,
null);
```

Deleting Rows

```
myDatabase.delete(DATABASE_TABLE, KEY_ID + "=" +
rowId, null);
```

Content Providers

Content Provider

- Content providers store and retrieve data and make it accessible to all applications.They're the only way to share data across applications !

- Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications

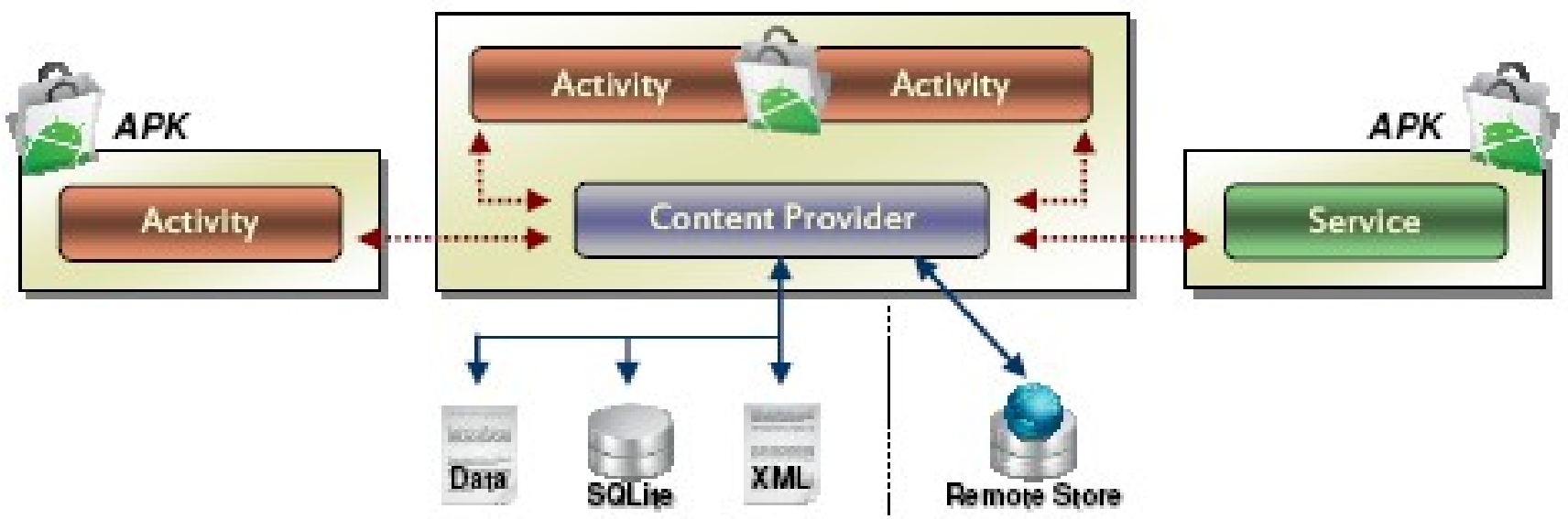
Content Provider	Intended Data
Browser	Browser bookmarks, browser history, etc.
CallLog	Missed calls, call details, etc
Contacts	Contact details
MediaStore	Media files such as audio, video and images
Settings	Device settings and preferences

Content Provider

- Create, retrieve, update, delete data of a particular type
- Exposes URI for each data set
- Content resolver
  - given URI, create, retrieve, update and delete data
- Table model
  - Cursor, ContentValues

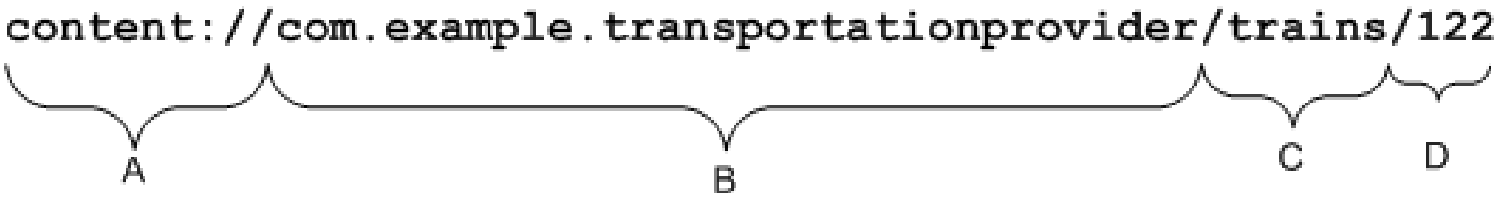
```
ContentResolver content = getContentResolver();

Cursor cursor =        content.query(Contacts.People.CONTENT_URI,

        PEOPLE_PROJECTION, null, null,
Contacts.People.DEFAULT_SORT_ORDER);
```

Content Provider



Content Provider URI

- A Content Provider exposes a unique URI used to query, add, update and delete data:
  - Standard prefix (“content://”)
  - The authority part, it identifies the content provider
  - To determine what kind of data is being requested
  - The ID of the specific record being requested



Ways of Using Content Provider

- Use the existing content provider as is.
- Modify the existing content provider [based on the requirement].
- Write a completely new content provider.

Creating a New Content Provider

- To create a new Content Provider, extend the abstract ContentProvider class.
- Override the onCreate method to create (and initialize) the underlying data source you're planning to publish with this provider.

Creating a New Content Provider

```
import android.content.*;
import android.database.Cursor;
import android.net.Uri;
import android.database.SQLException;
public class MyProvider extends ContentProvider
{
      @Override
      public boolean onCreate()
     {
       // TODO Construct the underlying database.
       return true;
      }
}
```

Content Provider URI

- You should expose a public static CONTENT_URI property that returns the full URI of this provider.

- A Content Provider URI must be unique to the provider, so it's good practice to base the URI path on your package name. The general form for defining a Content Provider's URI is:

```
content://com.<CompanyName>.provider.<ApplicationName>/<DataPath>
```

Content Provider

- Content URIs can represent either of two forms:
  - URI represents a request for all values
  - A trailing /<rownumber>, represents a request for a single record
- It's good practice to support access to your provider for both of these forms.

```
content://com.emertxe.provider.myapp/elements
```

```
content://com.emertxe.provider.myapp/elements/5
```

EMERTXE

Registering Your Provider

- Once you have completed your Content Provider, it must be added to the application manifest.

- Use the authorities tag to specify its base URI, as shown in the following XML snippet.

```
<provider android:name="MyProvider"
android:authorities="com.paad.provider.myapp"/>
```

Content Resolvers

- Content Resolver includes a number of methods to modify and query Content Providers.
- Each method accepts a URI that specifies the Content Provider to interact with.
- A Content Provider's URI is its *authority as defined by its manifest node*.

```
ContentResolver cr = getContentResolver();
```

Content Resolvers

```
ContentResolver cr = getContentResolver();

// Return all rows
Cursor allRows = cr.query(MyProvider.CONTENT_URI, null, null, null, null);

// Return all columns for rows where column 3 equals a set value
// and the rows are ordered by column 5.
String where = KEY_COL3 + "=" + requiredValue;
String order = KEY_COL5;
Cursor someRows = cr.query(MyProvider.CONTENT_URI,
null, where, null, order);
```

Querying for Content

- Content Provider queries take a form very similar to that of database queries.
- Query results are returned as Cursors over a result set
- Using the query method on the ContentResolver object, pass in:
  - The URI of the Content Provider data you want to query.
  - Lists the columns you want to include in the result set.
  - A *where clause that defines the rows to be returned*.
  - order of the returned rows.

Using the Media Storage Provider

- Media Store is a managed repository of audio, video, and image files.
- Whenever you add a new multimedia file to the file system, it should also be added to the Media Store.
- This will expose it to other applications, including the default media player.
- The MediaStore class includes Audio, Video, and Images subclasses, which in turn contain subclasses that are used to provide the column names and content URIs for each media provider.

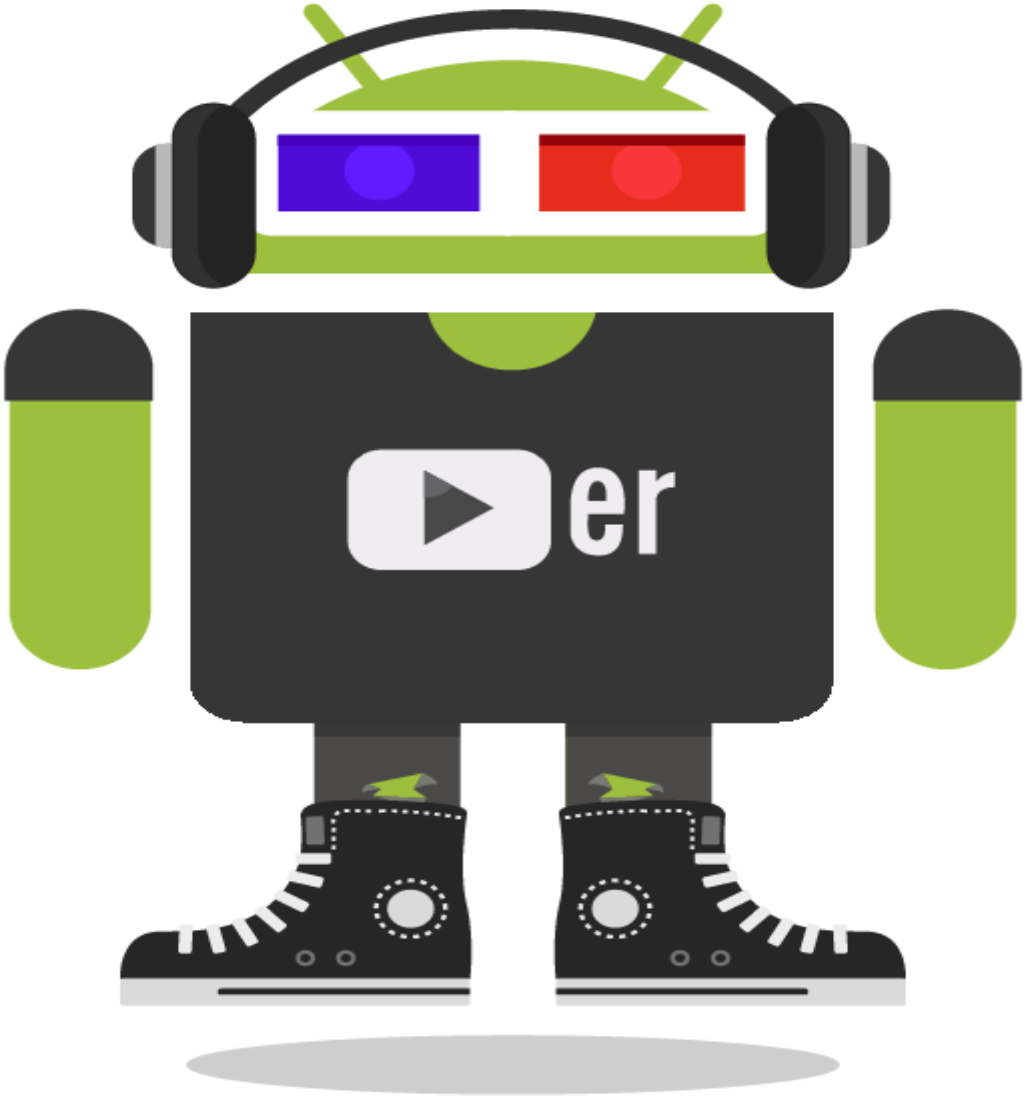
Using the Media Storage Provider

- The Media Store segregates media kept on the internal and external volumes of the host device.
- Each of the Media Store subclasses provides a URI for either the internally or externally stored media using the forms:

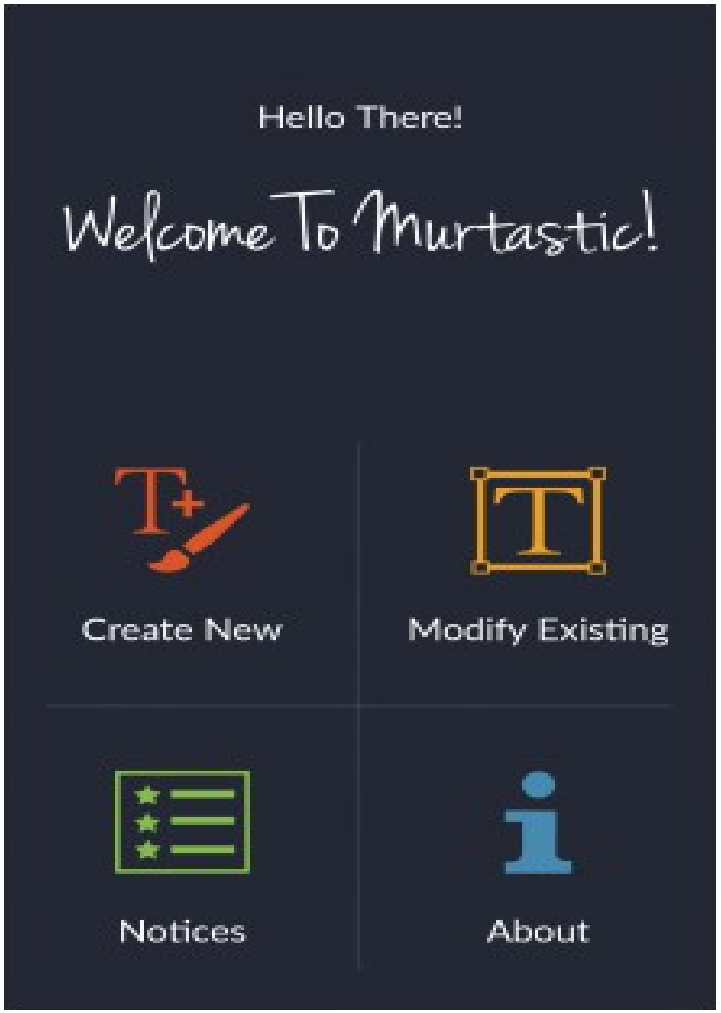
  - MediaStore.<mediatype>.Media.EXTERNAL_CONTENT_URI
  - MediaStore.<mediatype>.Media.INTERNAL_CONTENT_URI

Android Multimedia

Android Multimedia

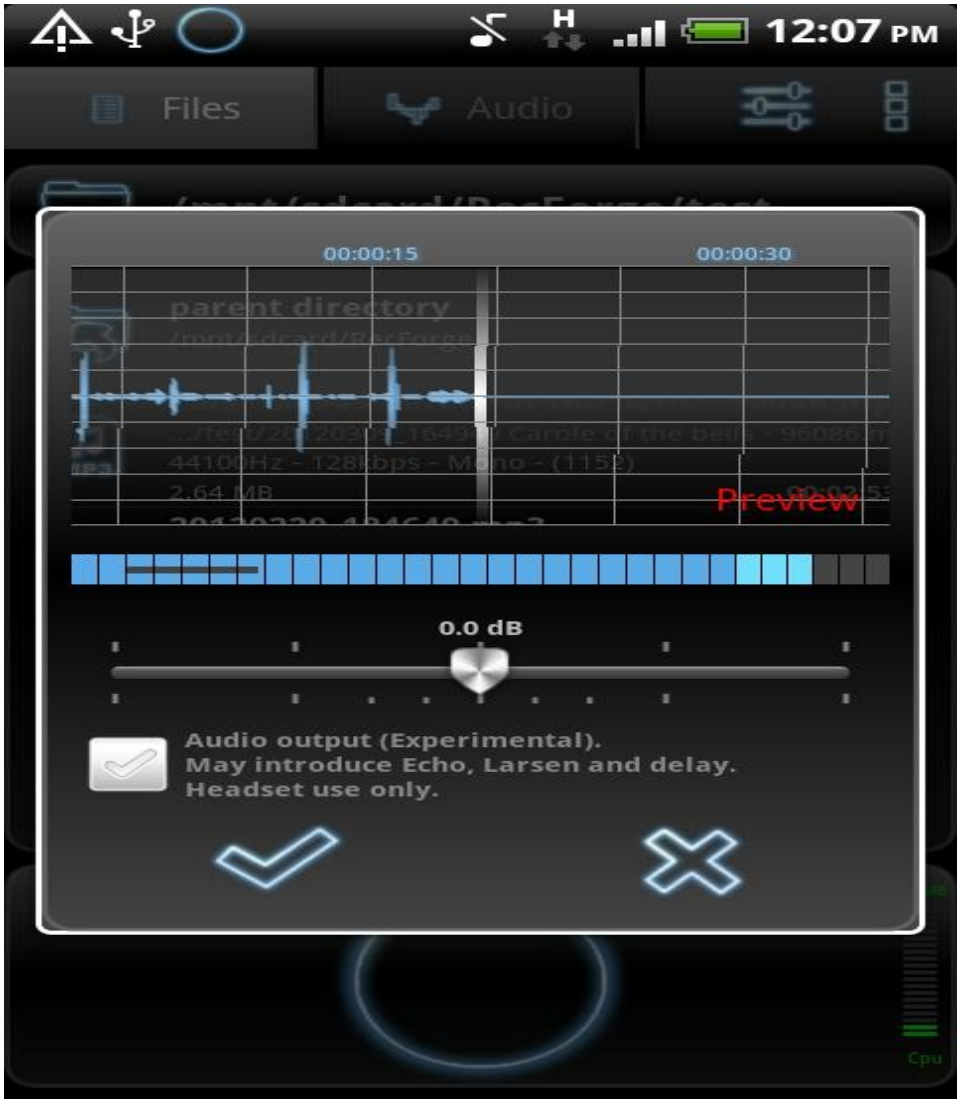


Live Wallpaper



Audio And Video

Playing Audio In Android



EMERTXE

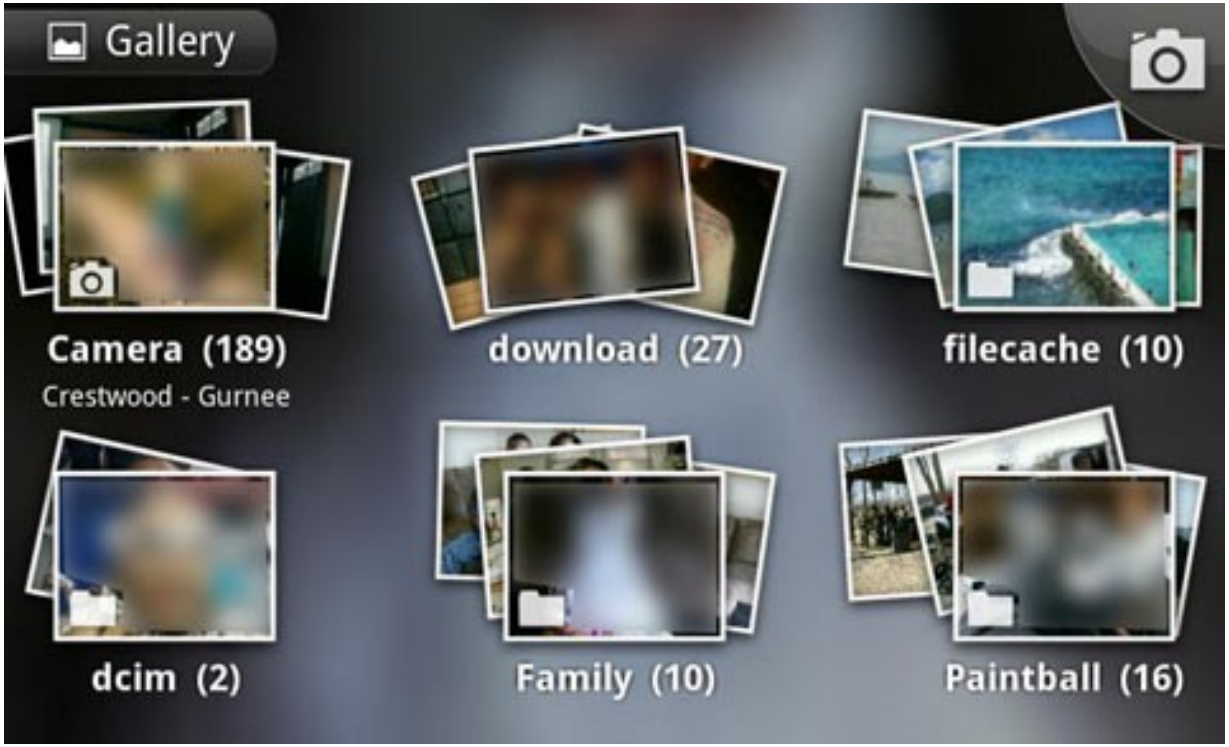
Playing Video in Android

EMERTXE

Android Alarm Manager

EMERTXE

Android Gallery



Android Speech API

Android Speech API

- Text to Speech
- Speech to Text

Android Telephony API

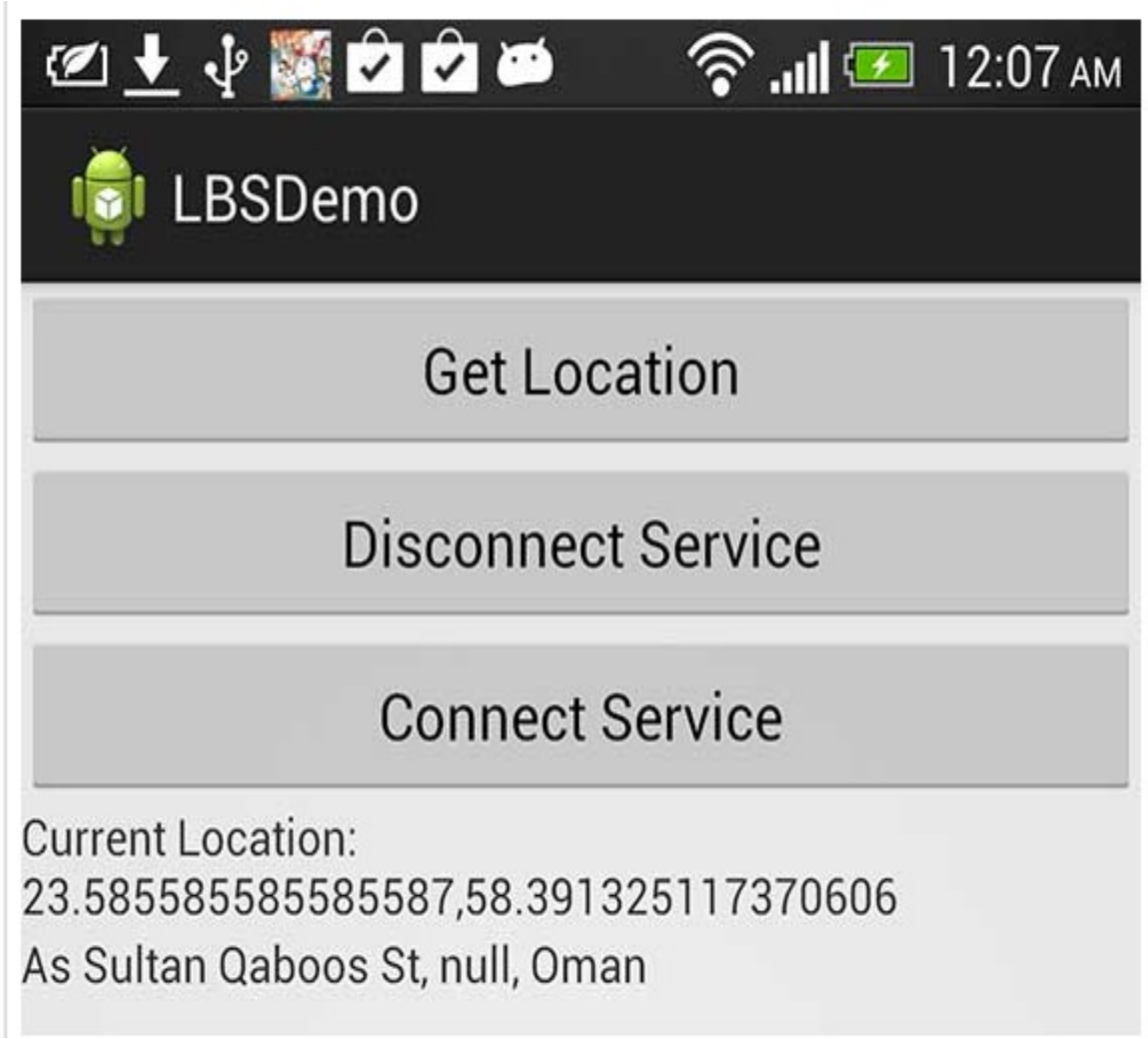
Android Telephony Manager

Telephony Manager Example

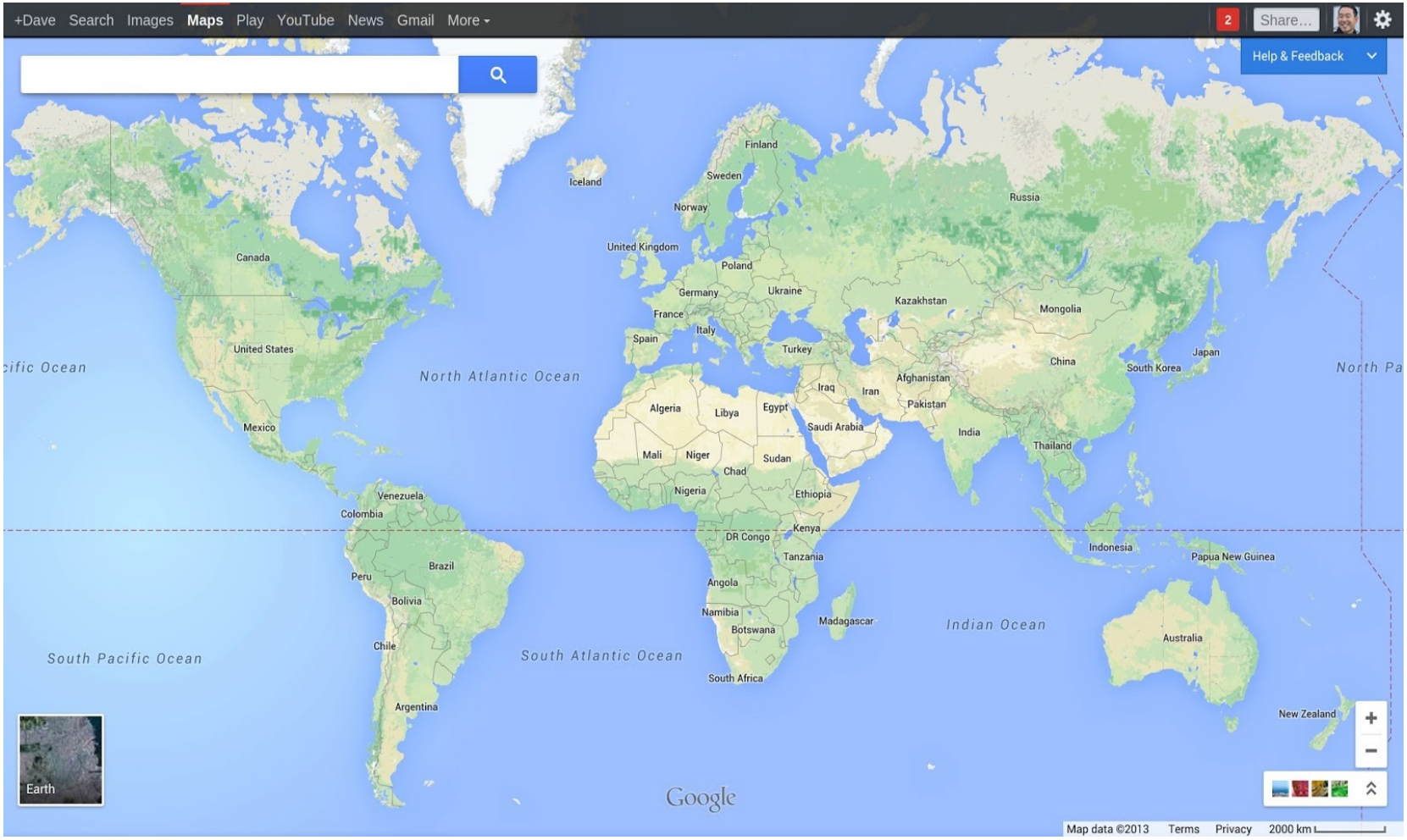
- Example of TelephonyManager that prints information of the telephony services.
- Determining whether phone is ringing or phone is in a call or phone is neither ringing nor in a call.
- Creating a basic android app that speaks the incoming number while phone is in ringing mode.
- Making a phone call in android via intent.
- Making a phone to send SMS and Email

Location and Maps

Android Location API



Google Maps


+Dave Search Images Maps Play YouTube News Gmail More
2
Share...
Help & Feedback
Finland
Sweden
Iceland
Norway
Russia
United Kingdom
Canada
Poland
Germany
Ukraine
Kazakhstan
Mongolia
France
Italy
Spain
Turkey
China
Japan
South Korea
United States
North Atlantic Ocean
Afghanistan
Iraq
Iran
Pakistan
Algeria
Libya
Egypt
Saudi Arabia
India
Thailand
Mexico
Mali
Niger
Sudan
Chad
Nigeria
Ethiopia
Venezuela
Colombia
Kenya
DR Congo
Tanzania
Indonesia
Papua New Guinea
Brazil
Peru
Angola
Bolivia
Namibia
Botswana
Madagascar
Indian Ocean
Australia
Chile
South Pacific Ocean
South Atlantic Ocean
South Africa
Argentina
New Zealand
Earth
Google
Map data ©2013
Terms
Privacy
2000 km

Android Device Connectivity

Agenda

- How to connect to the Wifi service
- Scanning all available wifi networks
- Bluetooth Services
- How to connect to a bluetooth adapter

Wifi Service

Connecting to the Wifi Service

- To define a Service, create a new class that extends Service. You'll need to override onBind and onCreate,

```
WifiManager wifi;

wifi=(WifiManager) getSystemService(Context.WIFI_SERVICE);

WifiInfo info-wifi.getConnectionInfo();

List<WifiConfiguration> configs=wifi.getConfiguredNetworks();
```

Sample Code to Scan Wifi Networks

```
WifiManager wifi;
wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

WifiInfo info = wifi.getConnectionInfo();
textStatus.append("\n\nWiFi Status: " + info.toString());

List<WifiConfiguration> configs = wifi.getConfiguredNetworks();
for (WifiConfiguration config : configs)
{
    textStatus.append("\n\n" + config.toString());
}

wifi.startScan();
```

Register a Broadcast Receiver

```
if (receiver == null)
receiver = new WiFiScanReceiver(this);

registerReceiver(receiver,
new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
```

In the Broadcast Receiver

In the onReceive method():

```
List<ScanResult> results = wifiDemo.wifi.getScanResults();

//wifiDemo is the original class
    ScanResult bestSignal = null;
    for (ScanResult result : results)
 {
    if (bestSignal == null  || WifiManager.compareSignalLevel(bestSignal.level,
result.level)<0)
 bestSignal = result;

}
```

Permissions

- Global access to a service can be enforced when it is declared in its manifest's <service> tag.
- By doing so, other applications will need to declare a corresponding <uses-permission> element in their own manifest to be able to start, stop, or bind to the service.
- < uses-permission android:name="android.permission.ACCESS_WIFI_STATE" / >
- < uses-permission android:name="android.permission.UPDATE_DEVICE_STATS"/ >
- < uses-permission android:name="android.permission.CHANGE_WIFI_STATE" / >

Questions

- Create a program that will scan all available wifi networks and display it on the screen
- When you click a scan button, it will scan the networks again and display the updated results on the screen
- Use the concept of BroadcastReceiver to update the screen once the scan is complete

Bluetooth

Bluetooth Adapter

- Represents the local Bluetooth adapter (Bluetooth radio).
- The **BluetoothAdapter** is the entry-point for all Bluetooth interaction.
- Using this, you can discover other Bluetooth devices, query a list of bonded (paired) devices, instantiate a **BluetoothDevice** using a known MAC address, and
  create a **BluetoothServerSocket** to listen for communications from other devices.

Bluetooth Socket

- Represents the interface for a Bluetooth socket (similar to a TCP **Socket**).

- This is the connection point that allows an application to exchange data with another Bluetooth device via InputStream and OutputStream.

Bluetooth Server Socket

- Represents an open server socket that listens for incoming requests (similar to a TCP **ServerSocket**).
- In order to connect two Android devices, one device must open a server socket with this class.
- When a remote Bluetooth device makes a connection request to the this device, the **BluetoothServerSocket** will return a connected **BluetoothSocket** when the connection is accepted.

Bluetooth Class

- Describes the general characteristics and capabilities of a Bluetooth device.
- This is a read-only set of properties that define the device's major and minor device classes and its services.
- However, this does not reliably describe all Bluetooth profiles and services supported by the device, but is useful as a hint to the device type.

Connecting to the Bluetooth Adapter

- Here you must get a BluetoothAdapter and then enable bluetooth.

1. Get BluetoothAdapter

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // Device does not support Bluetooth
}
```

2. Enable bluetooth

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

Android Camera

Android Camera

Android provides the facility to work on camera by 2 ways:

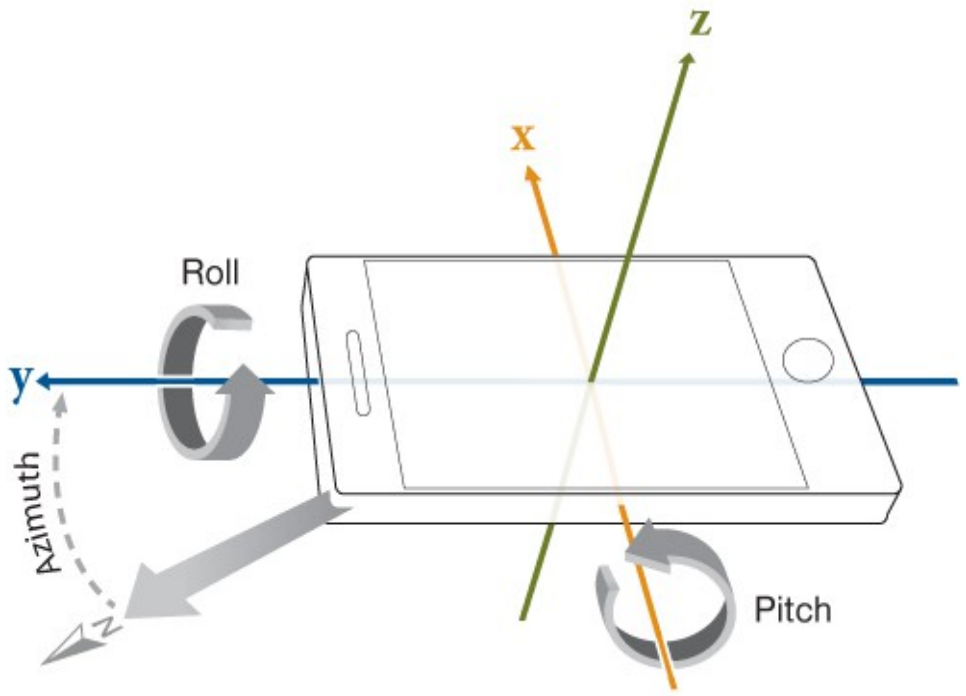
- By Camera Intent
- By Camera API

Android Sensors

Android Sensor

Types of Sensors:

- Motion Sensor
- Position Sensor
- Environmental Sensor



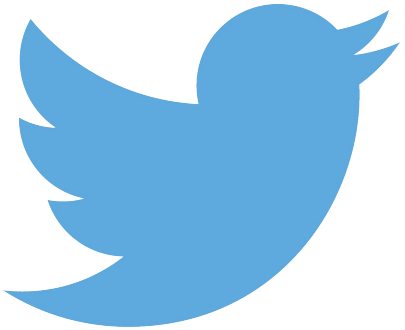
EMERTXE

Stay connected

About us: Emertxe is India's one of the top IT finishing schools & self learning kits provider. Our primary focus is on Embedded with diversification focus on Java, Oracle and Android areas

Emertxe Information Technologies,
No-1, 9th Cross, 5th Main,
Jayamahal Extension,
Bangalore, Karnataka 560046
T: +91 80 6562 9666
E: training@emertxe.com

https://www.facebook.com/Emertxe



https://twitter.com/EmertxeTweet

https://www.slideshare.net/EmertxeSlides

EMERTXE

Thank You