Introduction to Android development

Openium / ISIMA

Thomas Coulange
Olivier Goutet

23 January 2018

Introduction

- Android is the smartphone lead mobile platform
  - End consumer
  - Companies & industries
- Reliable and powerfull operating system
  - Allows a total control of the system

near field communication

  - Advanced fuctionnality (background task, NFC, external hardware support)
- Mobile application development need
  - The perfect understanding of the system
  - Knowledge of all the provided components
  - Creation of robust UI
  - Connectivity management, storage of data
  - Experience

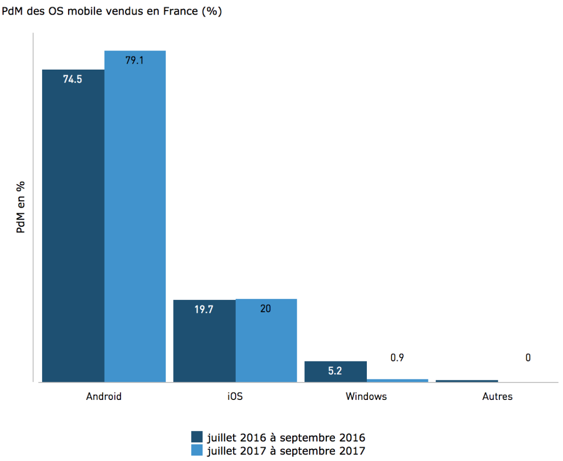
Plan

Plan

Android ?

- Operating system based of Linux kernel
- Developt by the Open Handset Alliance
    - Google, Samsung, HTC, Intel, Motorola, Qualcomm...
    - Google is the main contributor
- Free software
    - Apache licence

Market Share

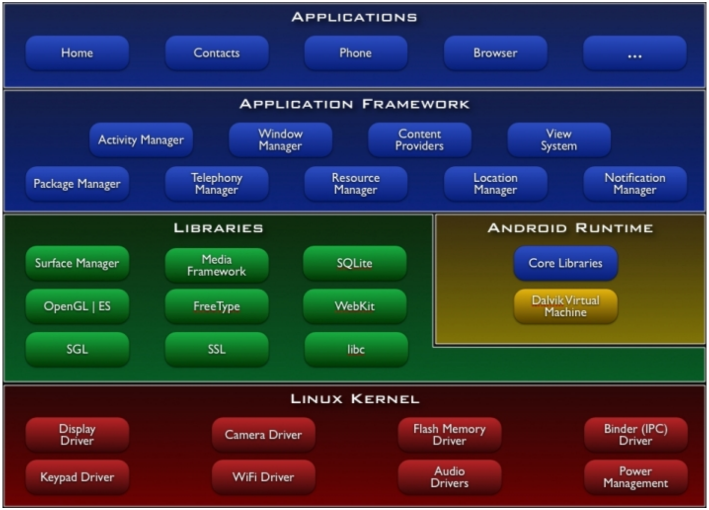


Kantar Worldpanel ComTech

Modified Linux based system

- No X Window
- No glibC ; replaced by Bionic libc
- Generously patched (power management, IPC...)
- Left from Linux
  - Memory and process management
  - Security (permissions)
  - Hadware abstraction layout (HAL)
  - Modules management
  - Community

Architecture



Android Runtime

- Android does not use the standard JVM
- Developement of a specific JVM :
  - Dalvik (2008-2014)
  - ART (2014 +)
    - New runtime since Android 5.0
    - Instead of JIT, it use AOT (Ahead Of Time) : Pre compilation in native code during the app install
    - Since Android 7.0, both JIT and AOT are used
    - No changes from the developer perspective
    - Better memory management, code optimisation, battery optimisation

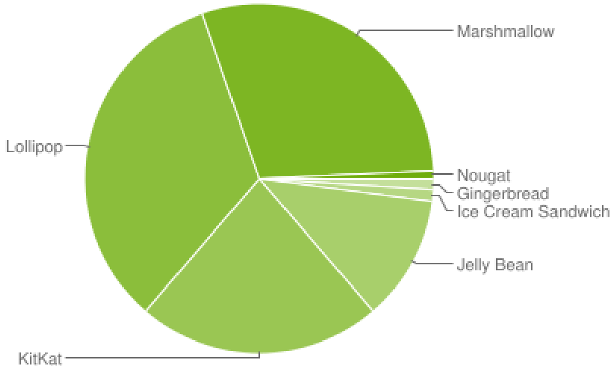
`https://source.android.com/devices/tech/dalvik/jit-compiler`

Fragmentation

- A lot of different versions of Android exist
  - Phone branch : 1.5 (2009), 1.6, 2.0, 2.1, 2.2, 2.3 (2010)
  - Tablet branch : 3.0 (early 2011), 3.1, 3.2 (2012)
  - Common branch : 4.0 (late 2011), 4.1, 4.2, 4.3, 4.4, 5.0, 5.1, 6.0, 7.0, 7.1, 8.0, 8.1 (late 2017)
- Android provide compatibility librairies to allow the use of new API (>4.0) on older versions of Android

So it's not really an issue (but still can be painfull on some things...)

Fragmentation – Market repartition



Data collected during a 7-day period ending on January 8, 2018.

Fragmentation – Market repartition

Version	Date	Codename	API	Distribution
2.3.3-2.3.7	early 2011	Gingerbread	10	0.4%
4.0.3-4.0.4	early 2012	Ice Cream Sandwich	15	0.5%
4.1-4.2-4.3	mid 2013	Jelly Bean	16-18	5.6%
4.4	late 2013	KitKat	19	12.8%
5.0	late 2014	Lollipop	21	5.7%
5.1	early 2015	Lollipop	22	19.5%
6.0	late 2015	Marshmallow	23	28.6%
7.0	mid 2016	Nougat	24	21.1%
7.1	late 2016	Nougat	25	5.2%
8.0	mid 2017	Oreo	26	0.5%
8.1	late 2017	Oreo	27	0.2%

Data collected during a 7-day period ending on January 8, 2018.

Any versions with

Android development – Different framework

- Cordova
- JQuery Mobile
- Xamarin
- Ionic
- Qt Android
- React Native
- Flutter
- ...

Android development – Different languages

- C++
- C#
- Java
- Kotlin
- HTML
- Javascript
- Lua
- Python
- ...

Android development – Official way

- Java / Kotlin
- Android SDK

Plan

Component list

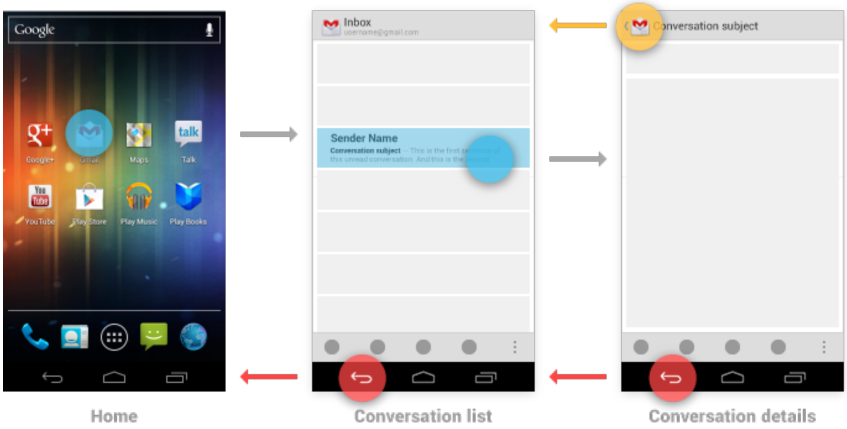
- Activity
- Fragment
- Service
- Content Provider
- Broadcast Receiver

Activity

In the official documentation : *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 [...]*

- Base component of nearly all Android applications
- Graphical display
- Principal class of an application
- Handle events
  - System events
  - User events
- Activity = screen

Activity – Example

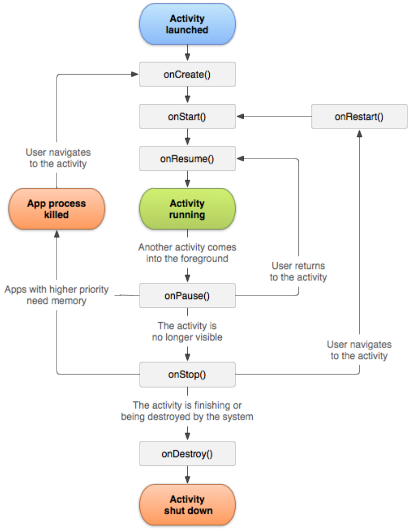


Activity – States

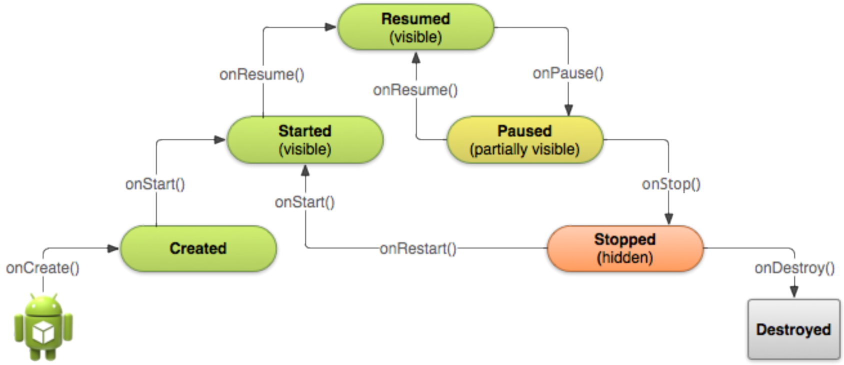
An `Activity` can be in 4 different states

- Running : displayed on screen and in foreground
- Paused : still displayed but not in foreground (dialog in front of it for example)
- Stopped : not visible anymore. State is saved. Can be killed by the system to save memory if needed
- Destroyed : deleted from memory. State can be saved

Activity – Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Activity – Simplified lifecycle



Service

In the official documentation : *A Service is an application component that can perform long-running operations in the background and does not provide a user interface. [...]*

- Background task
- No UI
- Running without the user knownledge
- A service runs on the main Thread (graphical)

Service use

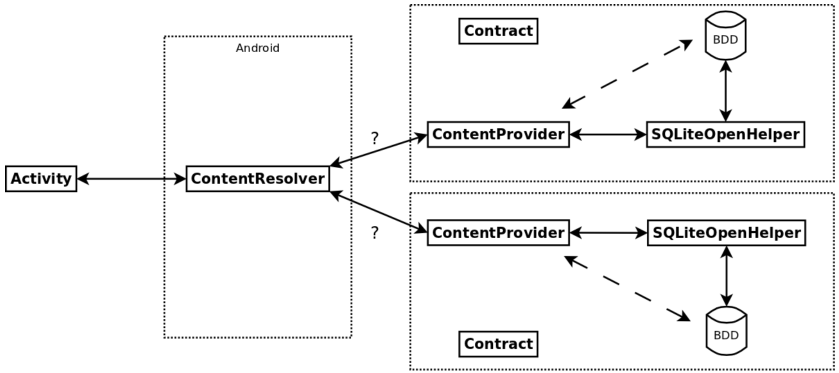
- Activity monitoring
- Periodic data update
- Music playback
- Network connections
    - Shoot and forget (send of data)
    - Allows to refresh some data without asking the user
- Geolocalisation
    - Avoid localisation to be linked with the `Activity` lifecycle

Content Provider

In the official documentation : *Content providers manage access to a structured set of data. [...]*

- Structured interface for data access
- Easy loose coupling of code and data
- Most often used to access to a database
- Allow data sharing between applications

Content Provider



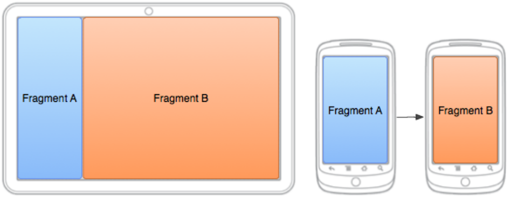
Broadcast Receiver

- Component allowing to received all the broadcast messages on the device
- Examples
    - System start
    - User position
    - Battery level
    - Connectivity change (Wi-Fi, 3G, no network)

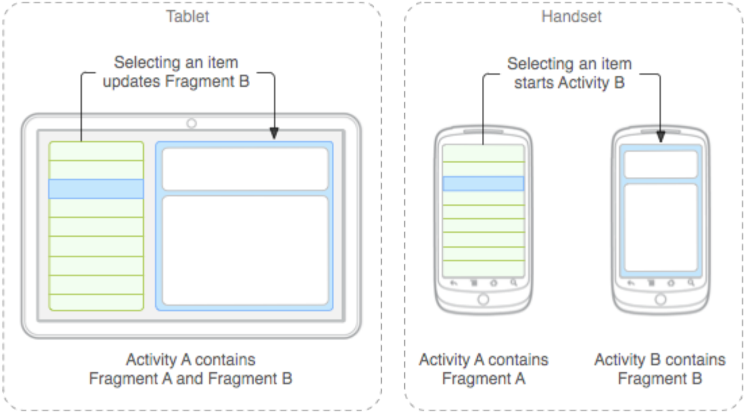
How to handle tablets ?

Activity – Fragment

- Subdivision of an `Activity`
- Created to avoid code duplication in a phone / tablet application
- Specific lifecycle



Activity – Fragment


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Component declaration

The `AndroidManifest.xml` file contains all the information about the application

- List of `Activity`, `Service`, `BroadcastReceiver` and `ContactProvider`
- Version and application name
- Compatibility with the different android versions

Plan

Security

- In the news : recurrent problem on Android
- The system block and manage a lot of cases (except for root user)
- The security system is based on permissions
  - Explicit declaration of functions needed by the application
  - User validation to accept the permissions

Permissions

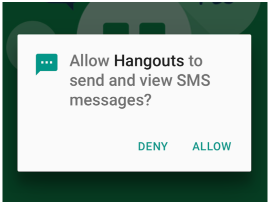
Since the user accepted, you can do what ever you want !

- `ACCESS_FINE_LOCATION`
- `ACCESS_NETWORK_STATE`
- `VIBRATE`
- `DELETE_PACKAGES`
- `READ_CONTACTS`
- `SEND_SMS`

Declaration of the permissions in the `AndroidManifest.xml` file in the project root directory.

Runtime permissions

- Before Android 6.0, the user acccept or decline ALL permissions. If you install the app, you accept all permissions.
- Since Android 6.0, the app need to ask the user before using a feature


Allow **Hangouts** to send and view SMS messages?

Plan

Intents

In the official documentation : *An intent is an abstract description of an operation to be performed.*

- Message to an
  - Activity (our application or another)
  - Service
  - BroadcastReceiver
- It allows to :
  - “change screen”
  - Start a service
  - Send a broadcast
  - ...

It’s a fundamental tool on Android. It allows different applications of different developers to communicate and work together in a loose coupling maner.

Intents – Data

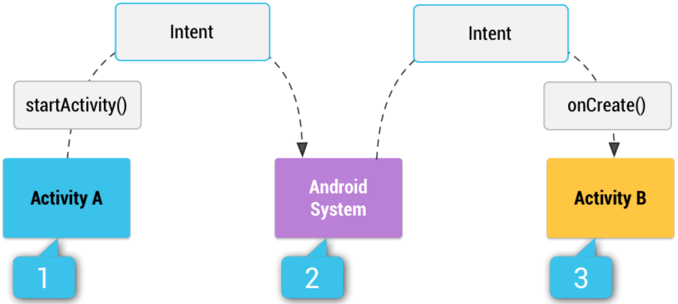
An `intent` can contains a set of data

- Action (`ACTION_VIEW`, `ACTION_EDIT`, `ACTION_MAIN`)
- Data (`Uri`)
- Category (`CATEGORY_LAUNCHER`, `CATEGORY_ALTERNATIVE`)
- Type (picture, contact, text...)
- Extras (Bundle key/value)
- Component

Intents – How it works ?

- Need to be created then “launch”
- The system analyse the intent and then try to find an application corresponding
  - Action/Type/Uri
  - Check in the package manager
- Start of the screen / function corresponding

Intents – How it works ?



Intents – Example

- **Action** ACTION_VIEW **Uri** http ://www.google.fr
- **Action** ACTION_DIAL **Uri** tel :123
- **Action** START_SCAN **Uri** null
- **Action** null **Uri** null **Component** fr.openium.isima.MyActivity

Intents – Types

- Explicit Intent
  - Intent sent to a specific component and not an other
  - The Action is not mandatory
  - Typically to start your own `Activity` `Service` in your own app
- Implicit Intent
  - Generic message sent to the system
  - Multiple application can answer
  - Example : show a shop position on a map

Intents : example

```java
// explicit intent
Intent i = new Intent(mContext, MyActivity.class);
startActivity(i);

// implicit intent
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.google.fr");
startActivity(i);

// data add
Intent i = new Intent(mContext, MyActivity.class)
i.putExtra("id_result", 1234);
i.putExtra("data", anObjet);
startActivity(i);
```

Describe the intents your application handle

For each component you can add filters describing the intents their handle.

```
<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.INSERT"/>
 <category android:name="android.intent.category.DEFAULT"/>
 <data android:mimeType="image/jpeg"/>
</intent-filter>
```

`AndroidManifest.xml` file in the project root directory.

Plan

Libraries – Support Library

- Provide retrocompability and bugs fixes to most Android class (Activity, Fragment, Media)
- Provide new components and view (Design) RecyclerView, Tablayout, BottomNavigationBar

Libraries – Architecture components

- Helps developers to build apps
  - Lifecycle
  - LiveData
  - ViewModel
  - Room
  - Paging
- Made by Google in 2017

Libraries – Play Services And Firebase

- Provide new features based on internet ou research like
  - Location helper
  - Push notification
  - Image processing / Face recognition
  - Realtime Remote Database
  - And much more

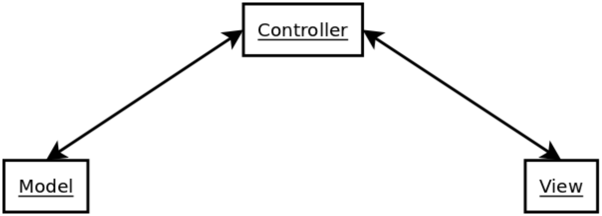
Plan

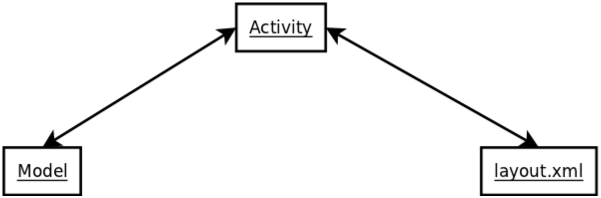
Graphical user interface

Most complex part of android. It needs the

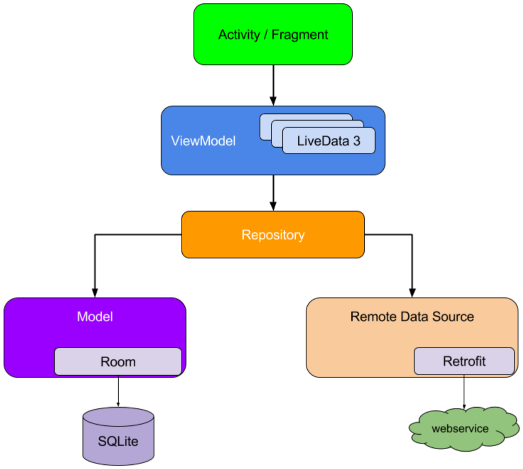
- Knownledge of components
- Handle of multiple screens (size, resolution)
- Display optimisations
- `Activity` lifecycle management

MVC





MVVM


Activity / Fragment
ViewModel
LiveData 3
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice

Layout files

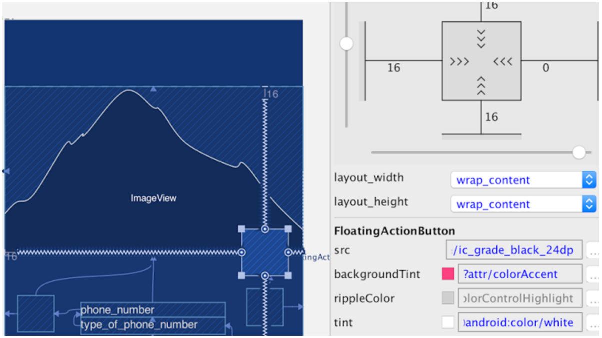
- XML defined interface
- Compiled and compacted by Android
- Parsed and displayed while the Activity creation

Exemple :

```
<RelativeLayout xmlns:android="http://schemas.android..."
  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:text="@string/hello_world">
  </TextView>
</RelativeLayout>
```

Layout WYSIWYG

- Directly in Android Studio
- A lot faster for using `ConstraintLayout`
- Could be better, but it's improved with each Android Studio versions



layout_width	wrap_content
layout_height	wrap_content

FloatingActionButton

src	:/ic_grade_black_24dp
backgroundTint	🟥 ?attr/colorAccent
rippleColor	›lorControlHighlight
tint	›android:color/white

Plan

Base Widgets

Widget : reusable graphical object

- TextView
- EditText
- Button
- ImageView
- Checkbox
- RadioButton

All widget attributes can be customized by XML or JAVA (in fact not all of them...)

Widget = View

All components inherit from `View`. So there own it's attributes.

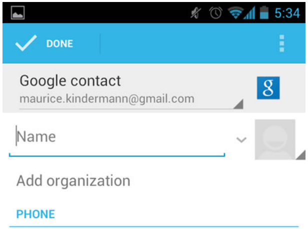
- height / width
- background
- focusable / clickable
- padding / margin
- visibility
- ...
- `http://developer.android.com/reference/android/view/View.html`

TextView

- Text display
- Main properties
    - text
    - textColor
    - textSize
    - ellipsize
    - lines
    - typeface
- Advanced properties
    - linkify
    - span
    - CompoundDrawable

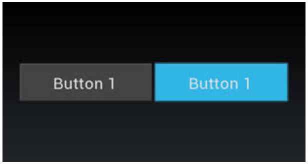
EditText

- Text edit
- Inherit TextView
- In fact, it's just an editable TextView with a specific look
- Main properties
  - hint
  - TextWatcher
  - EditorActionListener



Button

- Inherit from TextView
- In fact, just a clickable TextView



ImageView

- Allow image display
- Main properties
  - src
  - scaleType (center, fitXY, fitStart, centerCrop...)
- Sub class : ImageButton

Others

- Checkbox
  - Inherit from Button (so also from TextView)
  - Display a check and text
- RadioButton
  - Display a round button and some text
  - Link multiple RadioButton with a RadioGroup
  - Only one selectable in the group

Plan

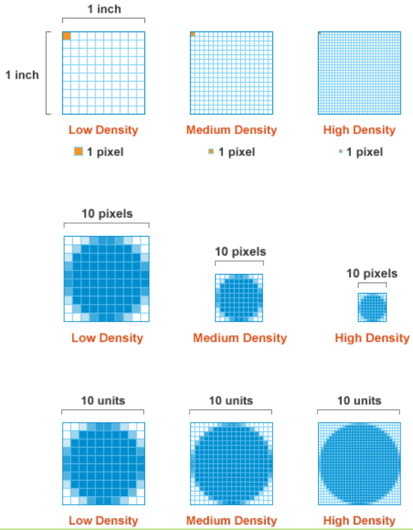
Multiple resolution management

- 11000+ different android device
- Screen size ?
- Resolution ?

Android has been design from scratch to handle the different screen size but it's the developer job to handle it properly.

- Element position
- Element size
- Adapt all elements to available size

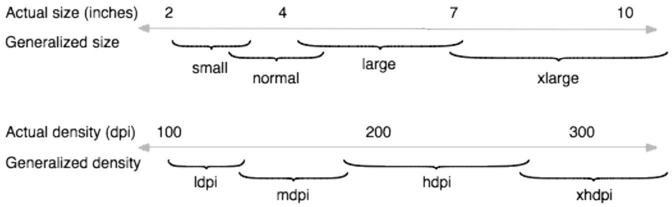
Screen density



Screen categories

All devices screens has been divided in categories

- mdpi 160 dip (ADP 2, Galaxy Tab 3)
- hdpi 240 dip (Nexus S, Galaxy S2)
- xhdpi 320 dip (Nexus 4, Galaxy S3, Nexus 7 2013, Nexus 9)
- xxhdpi 480dip (Nexus 5, Galaxy S4, Galaxy S5, Nexus 10)
- xxxhdpi 640dpi (LG G5, Galaxy S7, Nexus 6P)



Dimensions

With Android, you **never** use pixel sizes. Metrics has been created to define an element size without having to know the real pixel size or resolution of the screen.

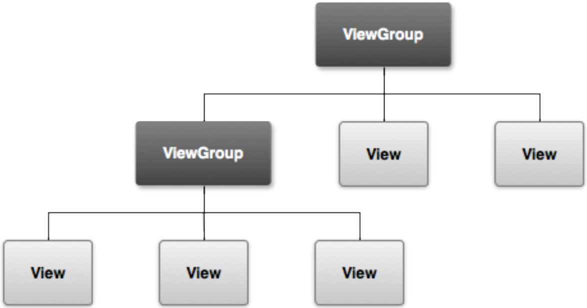
- Density Independent Pixel (dip or dp)
  - Physical size is identical whatever the screen size is
  - The number of pixel used to display the component varies with the screen density
  - Size of elements
- Scale-independent Pixels (sp)
  - Close of dip
  - Used exclusively for text size
  - Change of the text size with the user preference (disabled people)

Plan

Layouts

- View container
- Allows to place the different Views on the screen
  - Vertical / horizontal arangement
  - One above another
  - In relation to one another
- 3 principal layout
  - LinearLayout
  - RelativeLayout
  - FrameLayout

ViewGroup

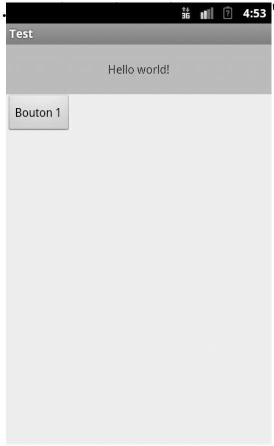


LinearLayout

- Vertical / horizontal arangement
- Element repartition
    - 50%-50%
    - 20%-80%
    - stuck to the bottom or right

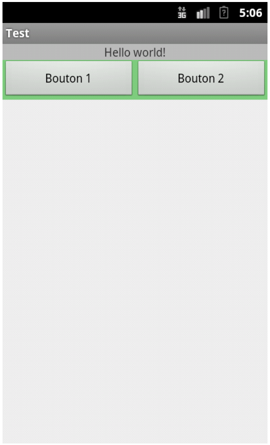
LinearLayout – simple example

```
<LinearLayout xmlns:android="http://schemas.android.
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EEEEEE"
    android:orientation="vertical" >
    <TextView
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:background="#BBBBBB"
        android:gravity="center"
        android:text="@string/hello_world" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button␣1" >
    </Button>
</LinearLayout>
```



LinearLayout – example weight

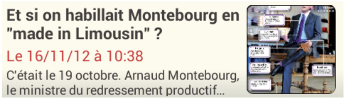
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#EEEEEE">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/hello_world" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:background="#7700AA00">
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button␣1" >
        </Button>
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button␣2" >
        </Button>
    </LinearLayout>
</LinearLayout>
```



RelativeLayout – example

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="90dip">
    <ImageView
        android:id="@+id/home_list_item_ImageView_Preview"
        android:layout_width="90dip"
        android:layout_height="90dip"
        android:layout_alignParentRight="true">
    </ImageView>
    <TextView
        android:id="@+id/home_list_item_TextView_Title"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@id/home_list_item_ImageView_Preview"
        android:ellipsize="end"
        android:maxLines="2"
        android:textStyle="bold" >
    </TextView>

    <TextView
        android:id="@+id/home_list_item_TextView_Date"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@id/home_list_item_TextView_Title"
        android:layout_toLeftOf="@id/home_list_item_ImageView_Preview"
        android:textColor="@color/lm_listview_cell_date" >
    </TextView>
    ...
</RelativeLayout>
```

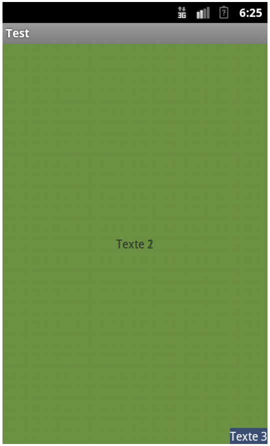

Et si on habillait Montebourg en "made in Limousin" ?
Le 16/11/12 à 10:38
C'était le 19 octobre. Arnaud Montebourg, le ministre du redressement productif...

FrameLayout

- Items One above another
- In relation to the layout it self (bottom, left, center...)

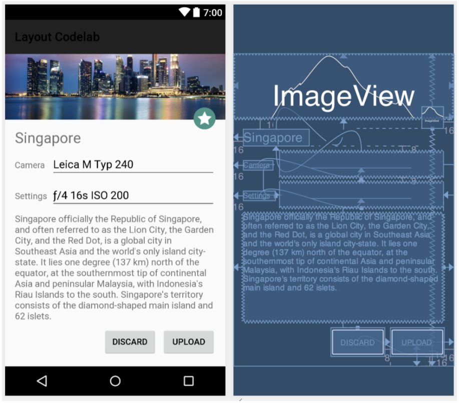
FrameLayout – example

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EEEEEE">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Texte␣1"
        android:background="#77AA0000">
    </TextView>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Texte␣2"
        android:background="#7700AA00">
    </TextView>
     <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|right"
        android:text="Texte␣3"
        android:background="#770000AA"
        android:textColor="@android:color/white">
    </TextView>
</FrameLayout>
```



ConstraintLayout

- Allows you to create large and complex layouts with a flat view hierarchy
- Similar to RelativeLayout (all views are laid out according to other views)



Singapore officially the Republic of Singapore, and often referred to as the Lion City, the Garden City, and the Red Dot, is a global city in Southeast Asia and the world's only island city-state. It lies one degree (137 km) north of the equator, at the southernmost tip of continental Asia and peninsular Malaysia, with Indonesia's Riau Islands to the south. Singapore's territory consists of the diamond-shaped main island and 62 islets.

Singapore officially the Republic of Singapore, and often referred to as the Lion City, the Garden City, and the Red Dot, is a global city in Southeast Asia and the world's only island city-state. It lies one degree (137 km) north of the equator, at the southernmost tip of continental Asia and peninsular Malaysia, with Indonesia's Riau Islands to the south. Singapore's territory consists of the diamond-shaped main island and 62 islets.

Other layout

- AbsoluteLayout
  - Fixed position elements
  - **Do not use !** Does not handle the different screen size
- GridLayout
- TableLayout
- SwipeRefreshLayout
- DrawerLayout

Plan

Event management

Every `View` can interact with the user

- Click / `OnClickListener`
- Focus / `OnFocusChangedListener`
- Touch ...
- For `TextView`
  - text modification / `TextWatcher`
  - keyboard validation / `OnEditorActionListener`

To listen thoses events, you need to register listeners

View – OnClickListener – Method 1

```
Button myButton = (Button) findViewById(R.id.ok);
myButton.setOnClickListener(new OnClickListener() {
  @Override
  public void onClick(View v) {
    // click button
  }
});
```

View – OnClickListener – Method 2

```
public class MainActivity extends Activity
                              implements OnClickListener {
  private Button mButton;

  protected void onCreate(Bundle savedInstanceState) {
    // ...
    mButton = (Button) findViewById(R.id.ok);
    mButton.setOnClickListener(this);
    // ...
  }
  public void onClick(View v) {
    if (v.equals(mButton)) {
      // click button
    } else {
      // ...
    }
  }
}
```

Plan

Lists

- Display a list of scrollable elements
- `ListView`
- One of the most complex widget in Android
- Simple to use
- Use an `Adapter` to load each of it's lines
- Specific event on line selection
  - `OnItemClickListener`

Adapter

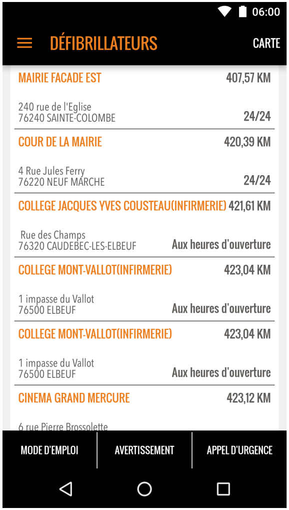
- In charge of providing data to the `ListView`
- Data can be stored in different types of data
  - Arrays : `ArrayAdapter`
  - Database : `CursorAdapter` and `SimpleCursorAdapter`
  - Other : `BaseAdapter`
- The `ArrayAdapter` and `SimpleCursorAdapter` class facilitate the Adapter integration for simple cells
- More explanations during the practical sessions

Plan

What is a ressource

- Text
- Color
- Image
- Dimensions
- ...

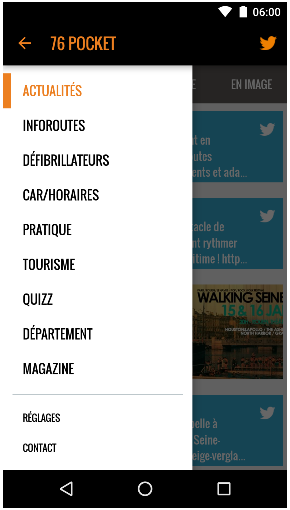
String ressources



06:00

DÉFIBRILLATEURS CARTE

MAIRIE FACADE EST	407,57 KM
240 rue de l'Eglise 76240 SAINTE-COLOMBE	24/24
COUR DE LA MAIRIE	420,39 KM
4 Rue Jules Ferry 76220 NEUF MARCHE	24/24
COLLEGE JACQUES YVES COUSTEAU(INFIRMERIE)	421,61 KM
Rue des Champs 76320 CAUDEBEC-LES-ELBEUF	Aux heures d'ouverture
COLLEGE MONT-VALLOT(INFIRMERIE)	423,04 KM
1 impasse du Vallot 76500 ELBEUF	Aux heures d'ouverture
COLLEGE MONT-VALLOT(INFIRMERIE)	423,04 KM
1 impasse du Vallot 76500 ELBEUF	Aux heures d'ouverture
CINEMA GRAND MERCURE	423,12 KM
6 rue Pierre Brossolette	



06:00

76 POCKET

t en
utes
ents et ada...

acle de
nt rythmer
itime ! http...

elle à
Seine-
eige-vergla...

String.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Pocket76</string>
    <string name="menu_settings">Settings</string>
    <string name="defibrilateur">Defibrilateur</string>
    <string-array name="encoding">
        <item>US-ASCII</item>
        <item>ISO-8859-1</item>
        <item>UTF-8</item>
    </string-array>
    <string name="deconnexion">disconnection of %s</string>
    <string name="user">User number %d</string>
</resources>
```

Strings – Localisation

- Localisation depends of the folder name
  - values-fr/ : French
  - values-pt/ : Portuguese
  - values-de/ : German
  - values-zh/ : Chinese (zhong guo)
  - values/ : default language
- The same strings have to be in the different directories

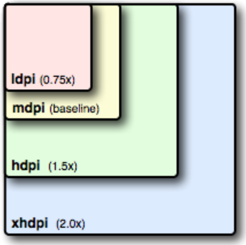
Values – Colors

Fichier `res/values/color.xml`

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="title_text_color">#FFFF0000</color>
    <color name="black">@android:color/black</color>
    <color name="item_text_color">@color/black</color>
    <color name="cell_background">#2e2e2e</color>
</resources>
```

Drawable – Bitmap

- Pre defined ratios
- Example of the application icon
    - 36x36 ldpi
    - 48x48 mdpi
    - 72x72 hdpi
    - 96x96 xhdpi
    - 144x144 xxhdpi
    - 192x192 xxxhdpi



Drawable – Screen size

- Specific ressources folder for each screen category
  - drawable-ldpi/
  - drawable-mdpi/
  - drawable-hdpi/
  - drawable-xhdpi/
  - drawable-xxhdpi/
  - drawable-xxxhdpi/
  - drawable/ shortcut to drawable-mdpi/
- If a ressource is missing for one screen size it's automatically created by Android

Values – Dimensions

Fichier `res/values/dimens.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="text_size_small">10sp</dimen>
    <dimen name="text_size_medium">15sp</dimen>
    <dimen name="text_size_big">20sp</dimen>
    <dimen name="cell_height">60dp</dimen>
    <dimen name="command_main_padding">25dp</dimen>
    <dimen name="command_padding_elements">
      @dimen/command_main_padding
    </dimen>
</resources>
```

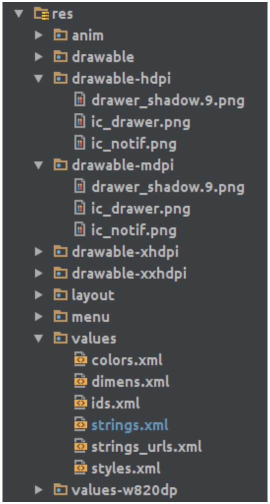
Values – Use

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginTop="@dimen/command_padding_elements"
  android:background="@color/black" >

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="@dimen/cell_height"
    android:text="@string/adresse_ip"
    android:textSize="@dimen/text_size_medium" >
  </TextView>

</LinearLayout>
```

Ressource structure



```
▼ res
    ▶ anim
    ▶ drawable
    ▼ drawable-hdpi
        drawer_shadow.9.png
        ic_drawer.png
        ic_notif.png
    ▼ drawable-mdpi
        drawer_shadow.9.png
        ic_drawer.png
        ic_notif.png
    ▶ drawable-xhdpi
    ▶ drawable-xxhdpi
    ▶ layout
    ▶ menu
    ▼ values
        colors.xml
        dimens.xml
        ids.xml
        strings.xml
        strings_urls.xml
        styles.xml
    ▶ values-w820dp
```

References used for the course

- http://developer.android.com
- http://android-developers.blogspot.fr
- http://developers.google.com/events/io
- http://fr.wikipedia.org
- http://stackoverflow.com
- http://android.cyrilmottier.com
- Développez pour Android (Cyril Mottier et Ludovic Perrier, éditions Digit Books)