SCIENCES SORBONNE UNIVERSITÉ

How to build a static application?

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Android Applicative Stack

ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL

ACTIVITY MANAGER • CONTENT PROVIDERS • LOCATION MANAGER • NOTIFICATION MANAGER • PACKAGE MANAGER • RESOURCE MANAGER • TELEPHONY MANAGER • VIEW SYSTEM • WINDOW MANAGER

AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT

CORE LIBRARIES • ART • DALVIK VM

AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV

AUDIO DRIVERS • BINDER (IPC) DRIVERS • BLUETOOTH DRIVER • CAMERA DRIVER • DISPLAY DRIVER • KEYPAD DRIVER • POWER MANAGEMENT • SHARED MEMORY DRIVER • USB DRIVER • WIFI DRIVER

Basics

One sandbox per Android Application

- One application = One linux user
- One application = One process
- One application = One virtual machine

Permissions

- Declared in AndrdoidManifest.xml
- The user may actively authorize some permissions (later in the lecture)

Two application can share UID to exchange information

Applications & Activities

One application is composed of activities

- For instance, a mail reader :
  - **One activity for displaying emails**
  - **One activity for reading emails**
  - **One activity for writing emails**
  - **…**

An activity is:

- A simple screen with an user interface
- Must inherits from Activity class

An activity may launched from another application

- A mail reader can launch the camera

How activities are working ?

Model-View-Controller scheme

GUI is defined through XML files

- res/layout: for views
- res/menu: for the associated menus

Controller derived from the Activity class

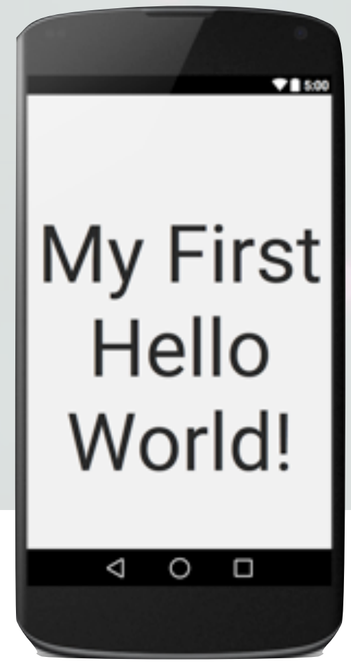
- Load the GUI
- Register to events
- Update the view

Model is updated by the controlling according to user actions

Building a GUI

Easy, use a WISIWYG editor

- Use simple drag-and-drops



res/layout/activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:baselineAligned="false"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="635dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="My First Hello World!"
        android:id="@+id/textView"
        android:layout_weight="1.05"
        android:gravity="center_vertical|center_horizontal"
        android:textSize="100dp" />
</LinearLayout>
```

Building a GUI

Easy, use a WISIWYG editor

- Use simple drag-and-drops

res/layout/activity_main.xml:

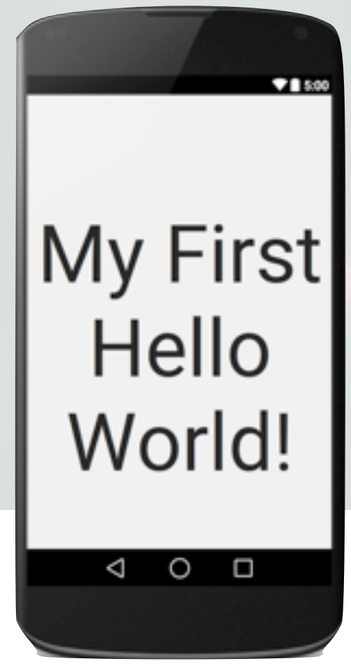
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:baselineAligned="false"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="635dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="My First Hello World!"
        android:id="@+id/textView"
        android:layout_weight="1.05"
        android:gravity="center_vertical|center_horizontal"
        android:textSize="100dp" />
</LinearLayout>
```

Building a GUI

Easy, use a WISIWYG editor

- Use simple drag-and-drops



res/layout/activity_main.xml:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:baselineAligned="false"
    android:weightSum="1">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="635dp"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="My First Hello World!"
        android:id="@+id/textView"
        android:layout_weight="1.05"
        android:gravity="center_vertical|center_horizontal"
        android:textSize="100dp" />
</LinearLayout>
```

Layouts

Setup for the GUI structure

- ViewGroup implementation

Three main layout

- Linear Layout
  - **Elements are ordered horizontally or vertically**
  - **A scrollbar is generated for invisible elements**

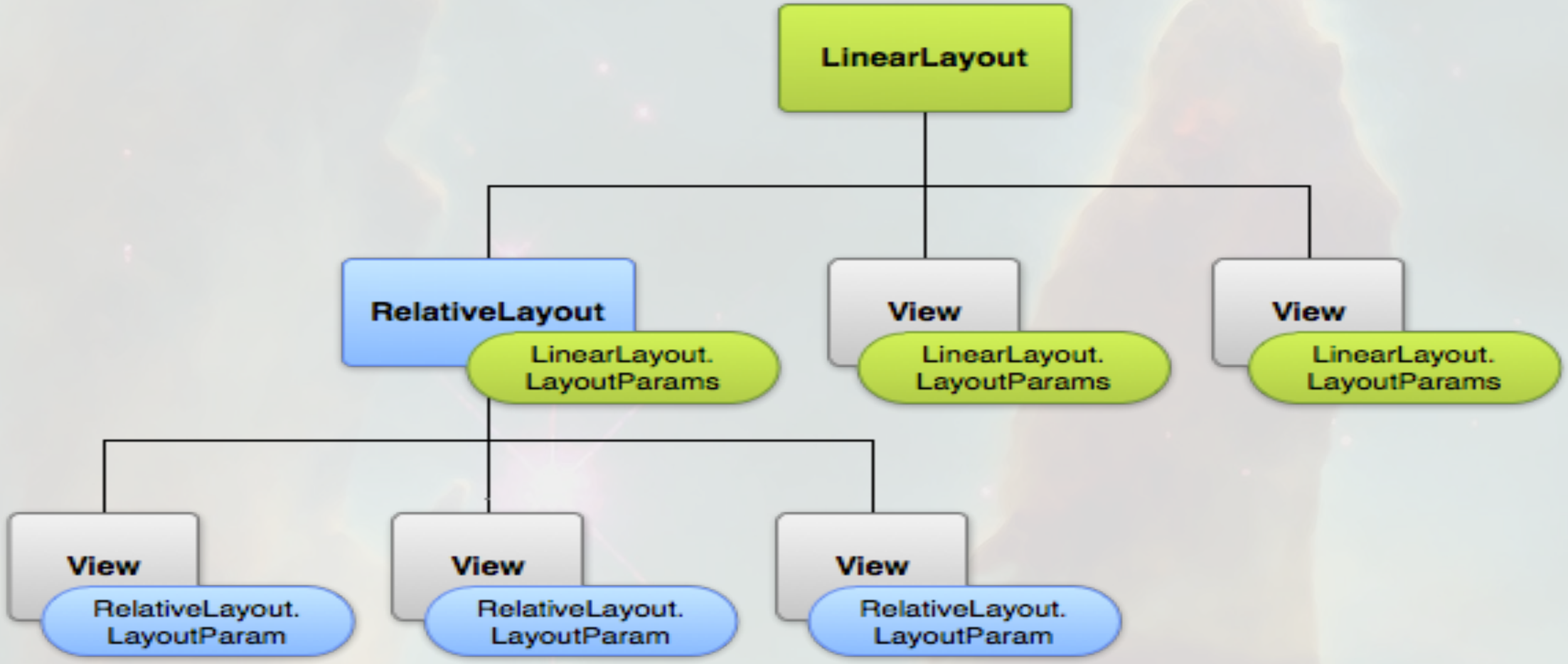
- Relative Layout
  - **Elements are ordered according to their relative position with other elements**

- Webview
  - **Display a webpage**

Views Hierarchy

A screen can be seen as a tree view

- ViewGroup
  - views containers (internal nodes of the tree)
- Views
  - TextView, Buttons, etc. (leaf nodes of the tree)



Display a GUI

In the controller

- modify java/.../MainActivity.java
- Setup in the onCreate method
- Specify which GUI XML file we want to use

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // More stuff here ...
}
```

Display a GUI

In the controller

- modify java/.../MainActivity.java
- Setup in the onCreate method
- Specify which GUI XML file we want to use

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // More stuff here ...
}
```

The "mysterious" Ressource class

Automatically generated during the compilation

- Binding between the view and the controller

Associate an ID to all elements

- Layout
- Views
- Images
- ...

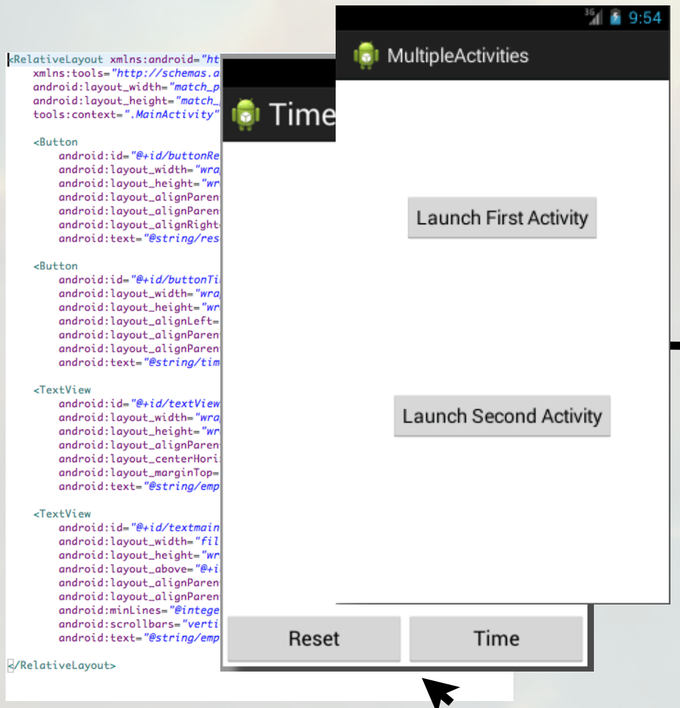
How to get the ID of an element

```
FindViewById (R.id.my_fancy_object)

FindViewById (R.layout.my_fancy_object)
```

The relation between the elements

36



Automatic Generation

Class R

Access

Creation / Instantiation

Source

Device orientation

We can build a view per orientation

- One XML for the landscape view
- One XML for the portrait view
- The two XML must have the same name

Device orientation

We can build a view per orientation

- One XML for the landscape view
- One XML for the portrait view
- The two XML must have the same name

Only for static application

Summary

Overview of an Android Application

- The applicative stack
- Sandboxing

How to define a GUI

- Through XML files
- View hierarchy
- Landscape / Portrait pre-defined views

How the R class is working

- Automatic Generation of ID for each elements