Android Fundamentals

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

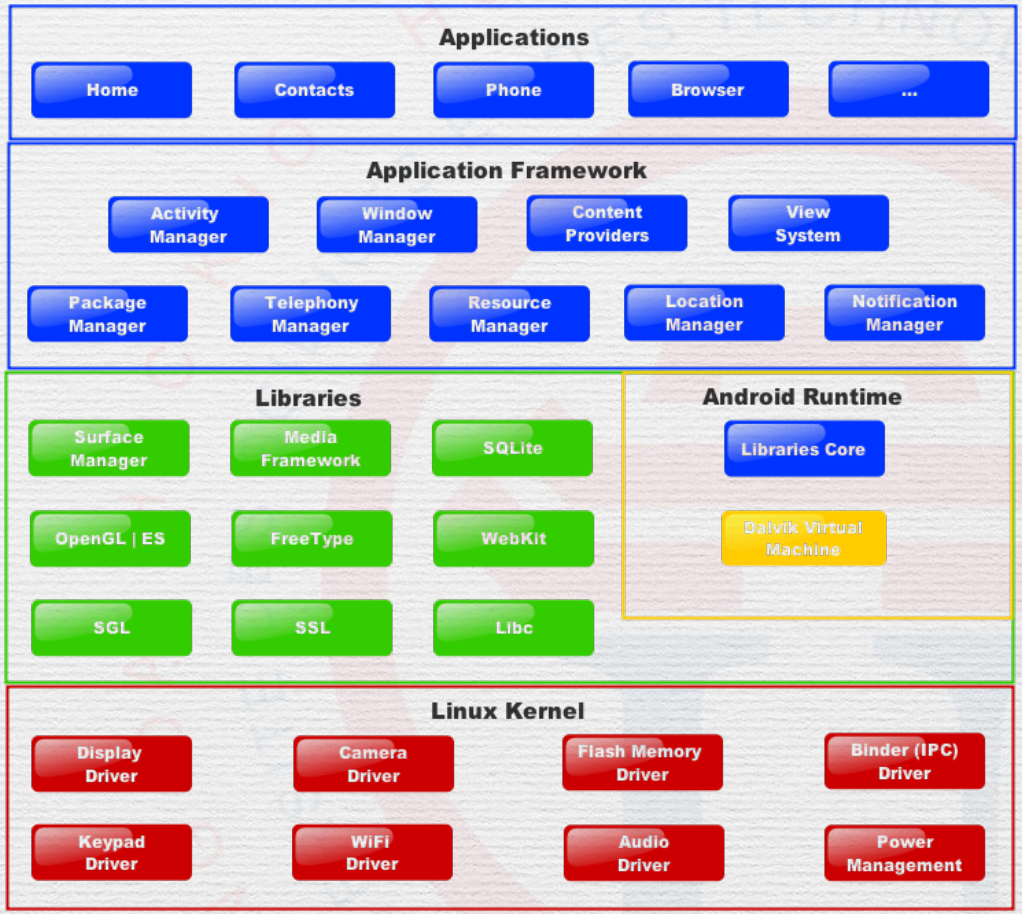
Content

- Architecture
- Compilation
- Controllers: Context, Application, Activity, Fragment
- View

Architecture

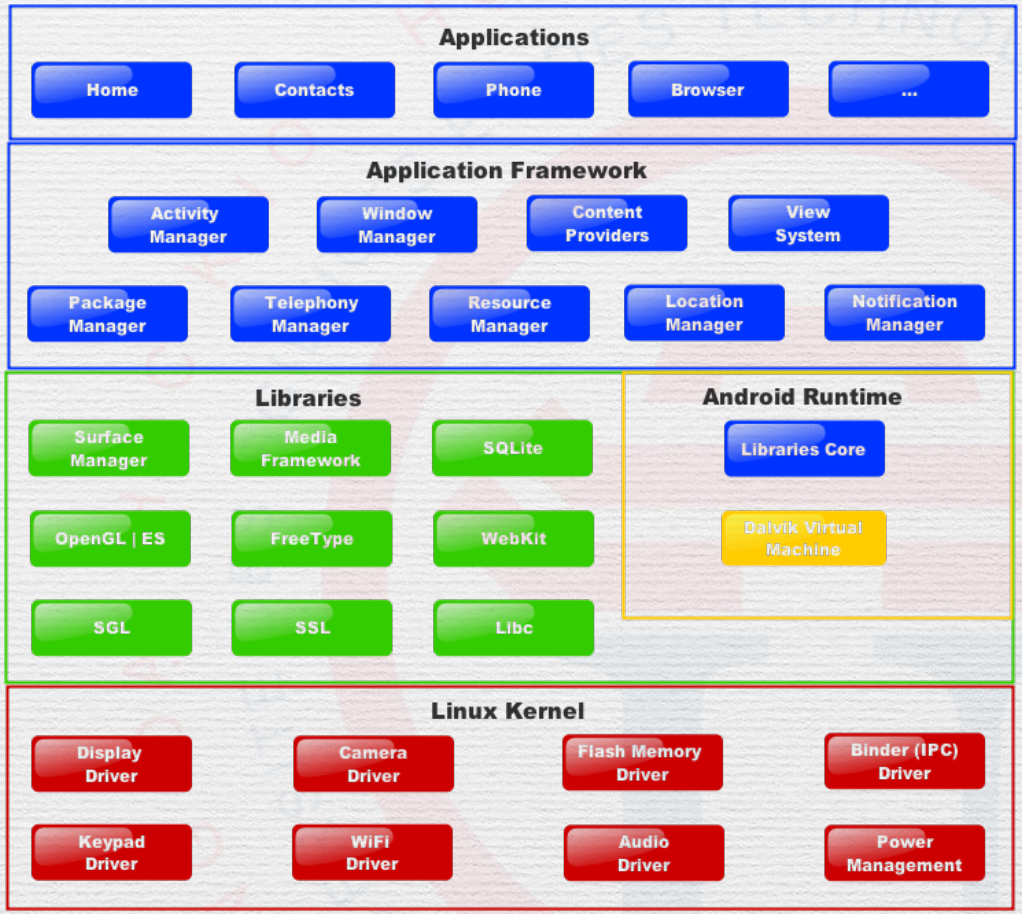
Overview

- Applications
- Application Framework
- Libraries
- Linux Kernel



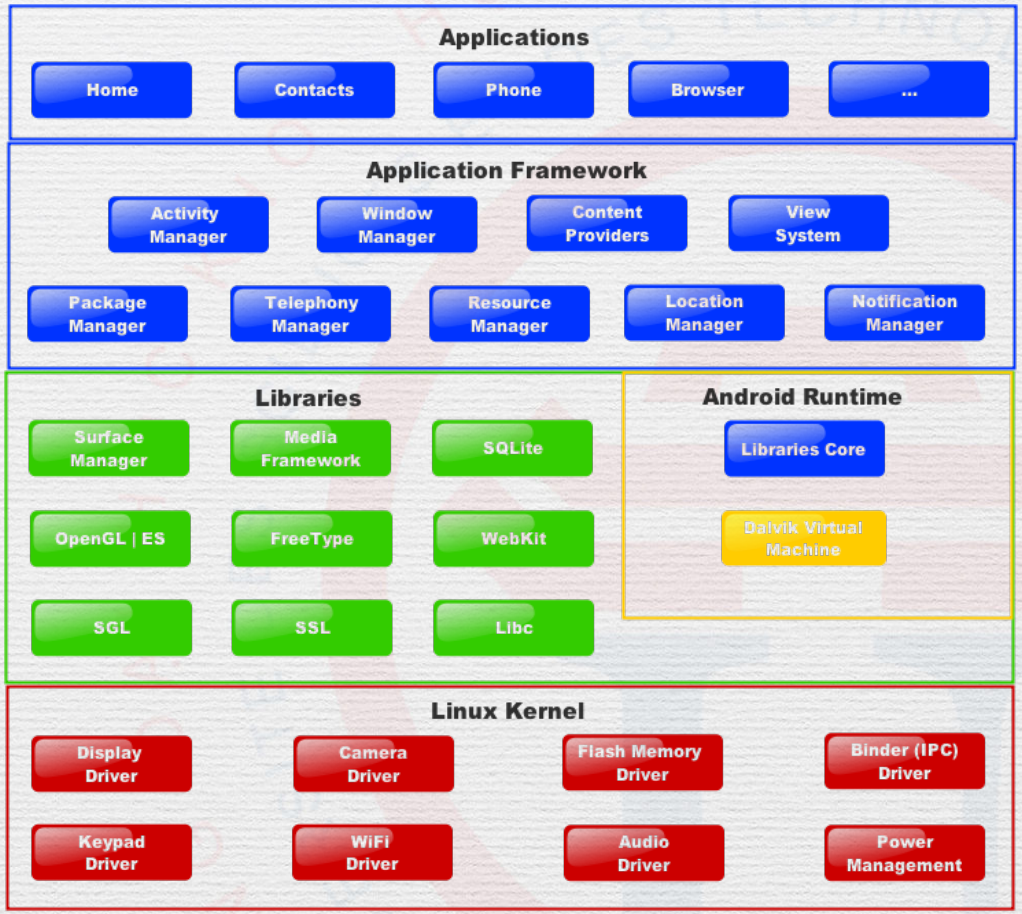
Linux Kernel

- Well shaped
- Secured
- Active development



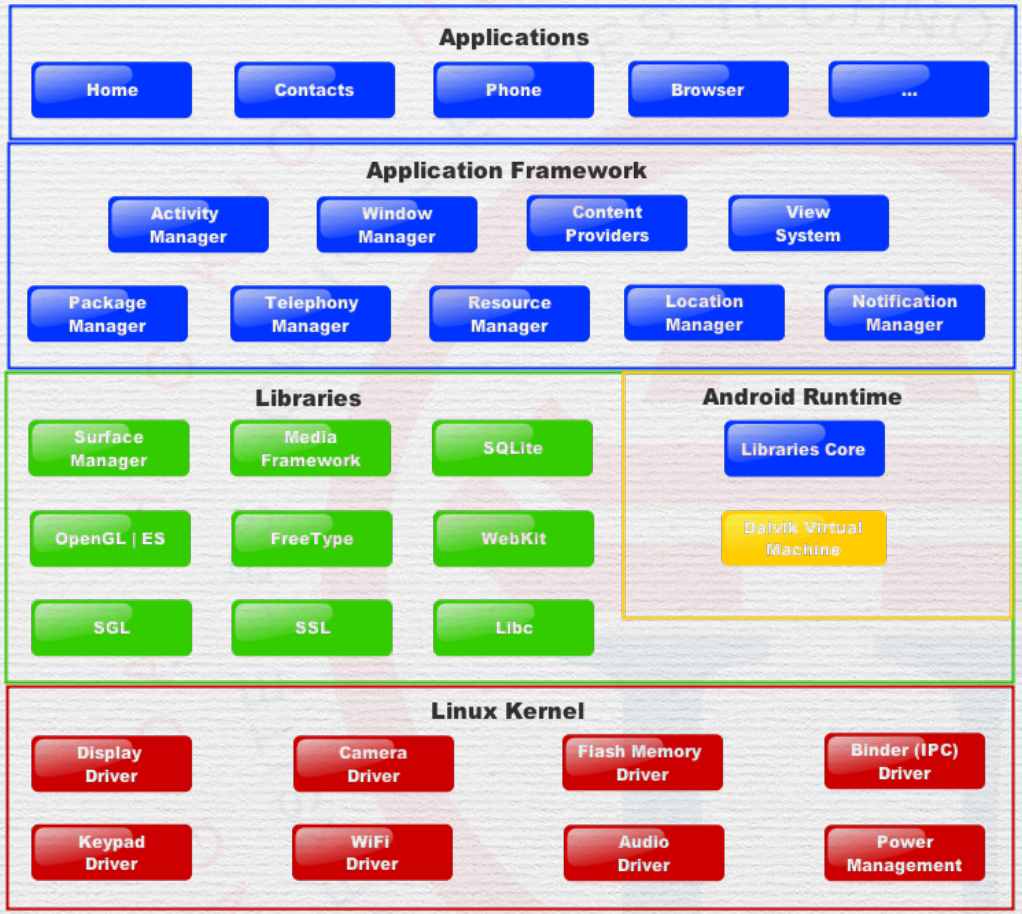
Libraries

- Mostly in C/C++
- Low level
- Render text
- Play media
- Local databases
- ...



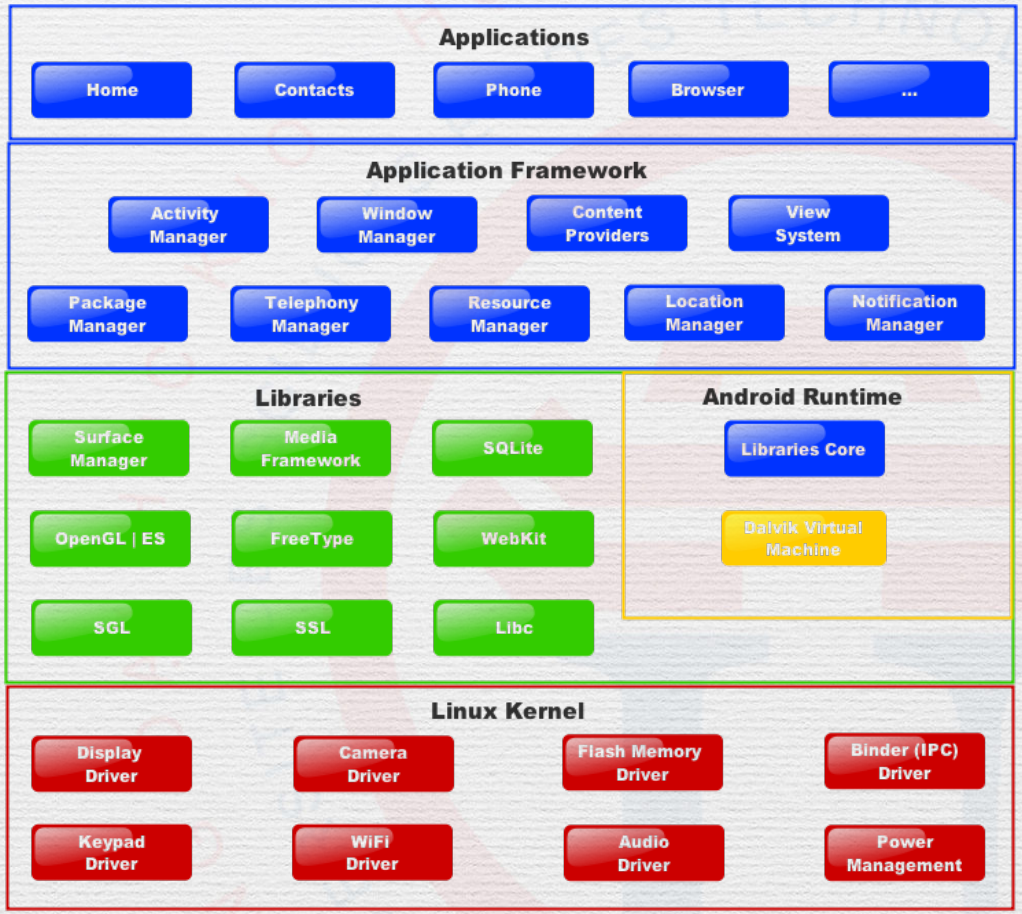
Application Framework

- Java
- Higher level
- User Interface
- Location Service
- Notification
- ...



Applications

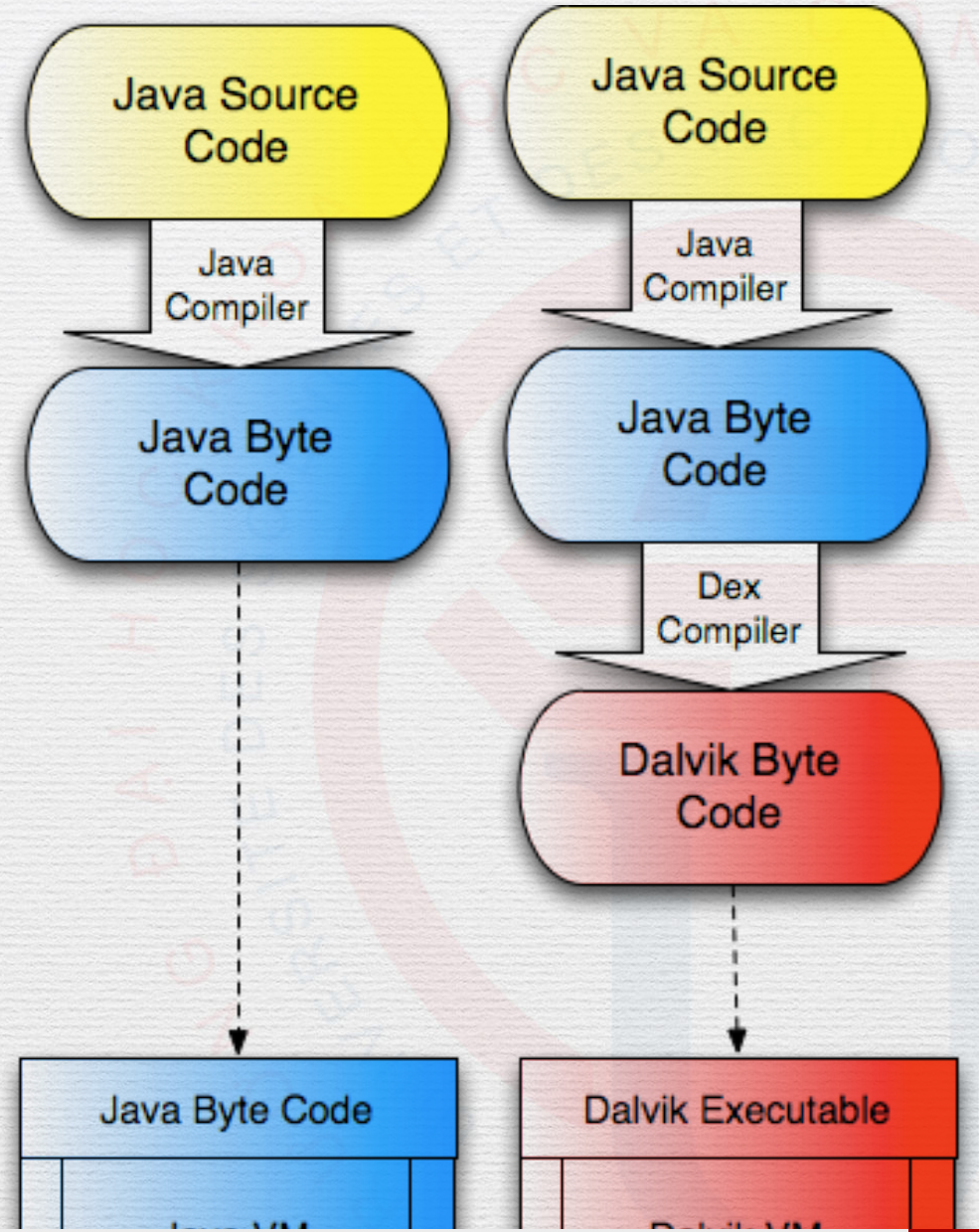
- Java
- Our focus
- Where you will make your app



Compilation

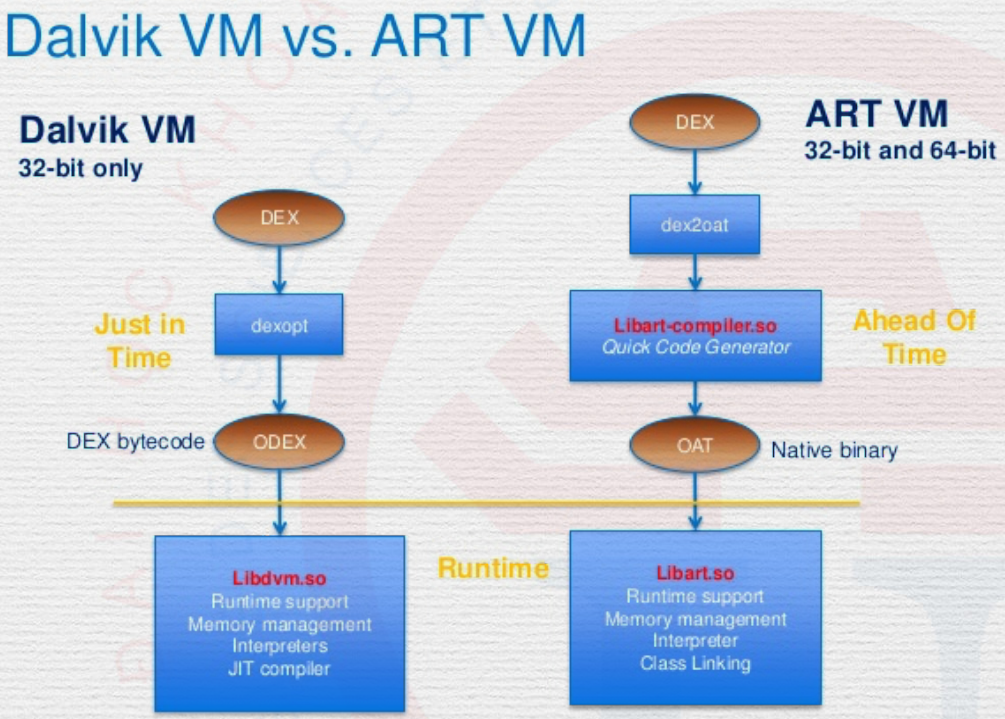
From Source to Device

- Java
- Dex
- Virtual Machine
  - Separated Process
  - Separated User
  - Isolation
- Why Java?

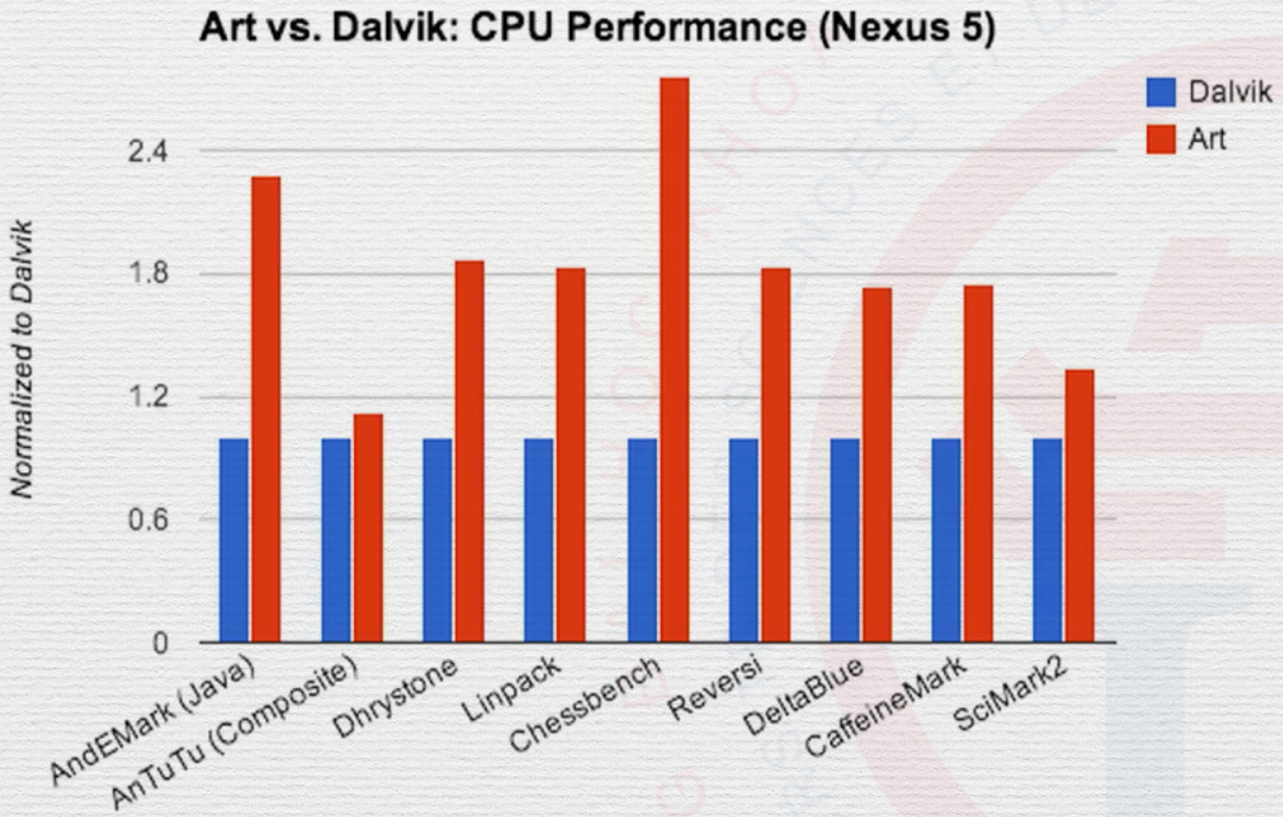


Android Virtual Machines

- Dalvik
- ART
  - Android RunTime



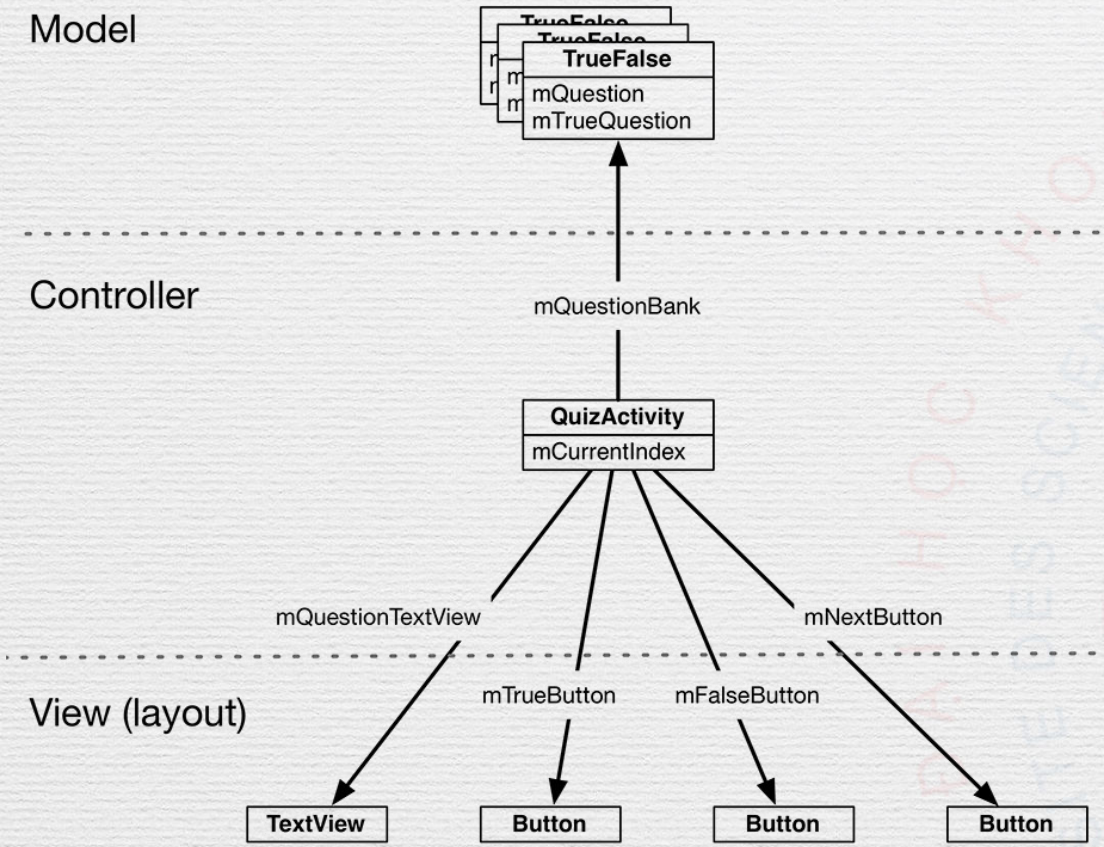
Android Virtual Machines

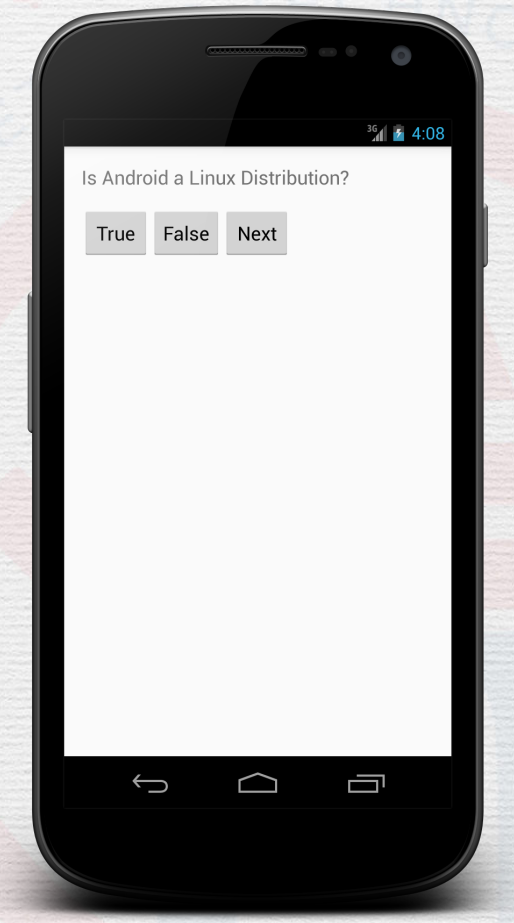


Source: AnandTech

Context & Application

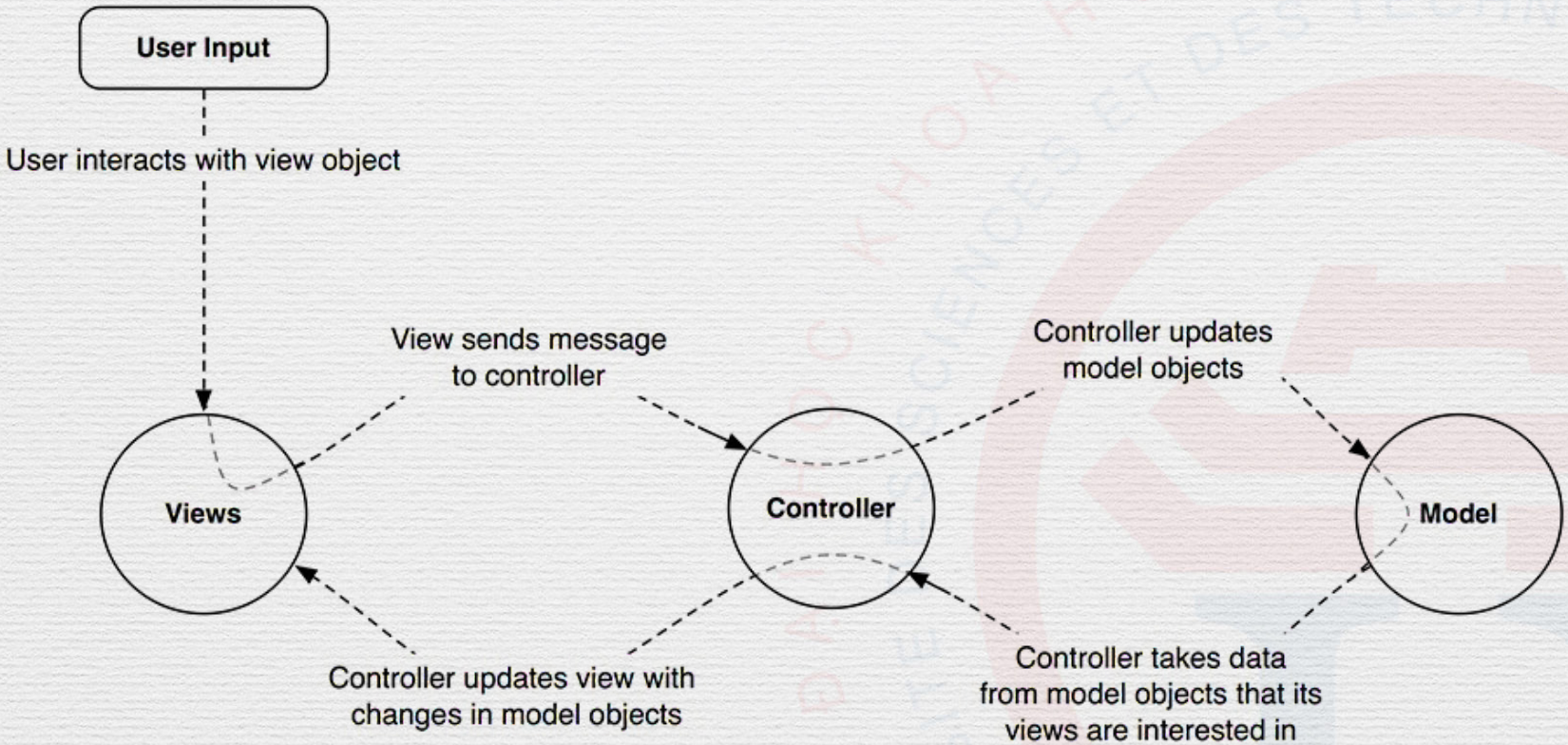
Simple « MVC » Model





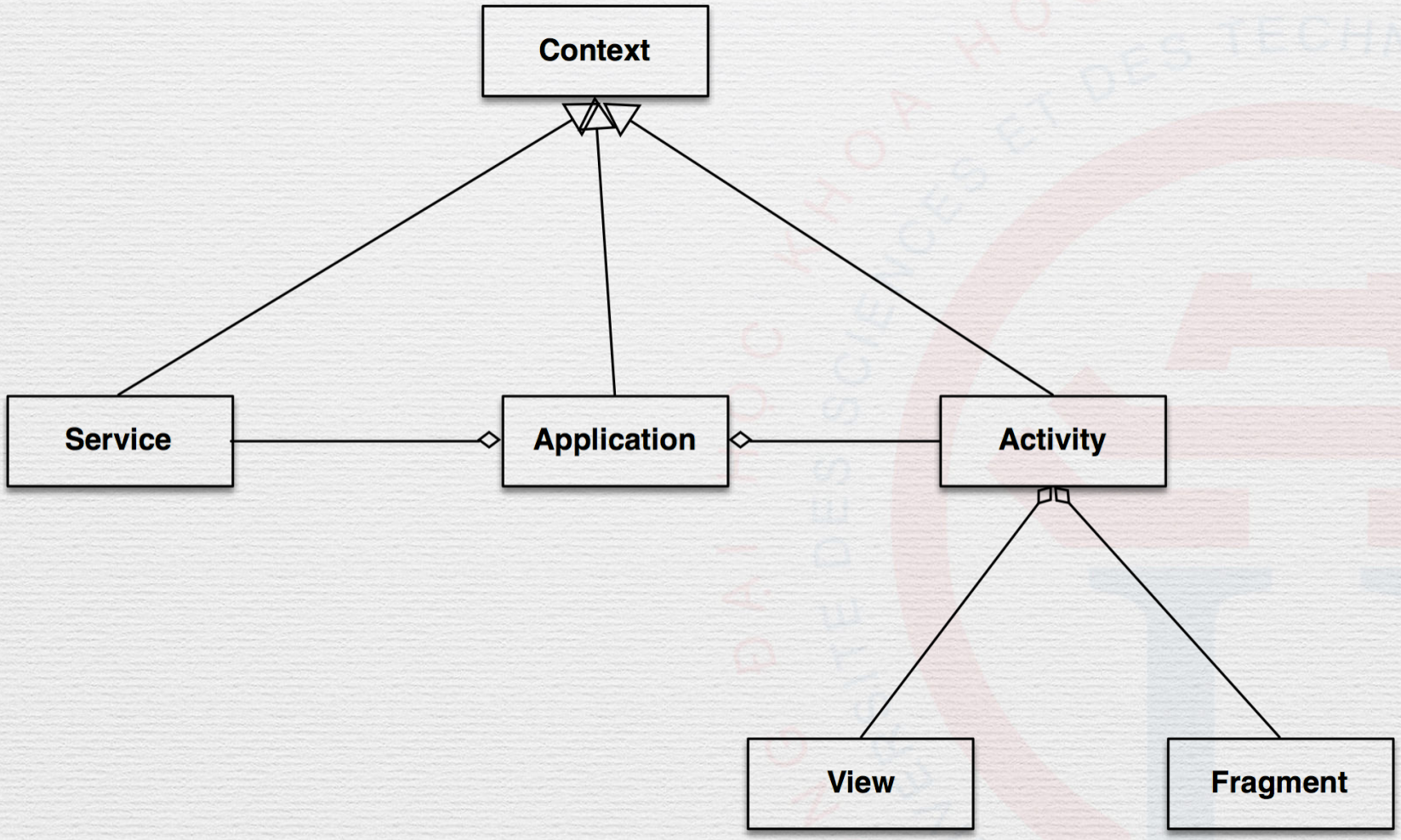
Android Programming: The Big Nerd Ranch Guide, 2nd Edition

Simple « MVC » Model

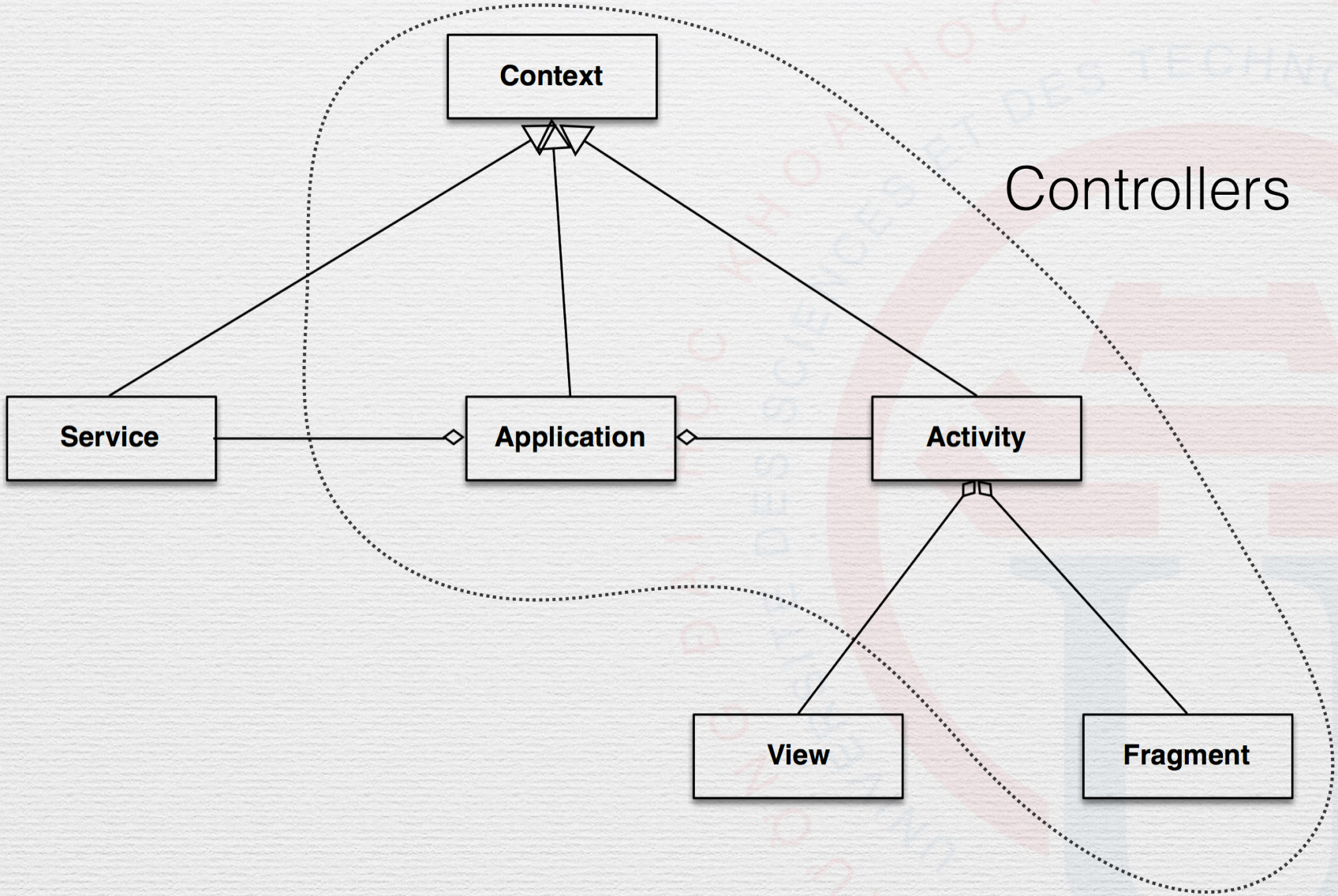


Android Programming: The Big Nerd Ranch Guide, 2nd Edition

Android Application's Components



Android Application's Components



Controllers

- **Context**
- **Application**
- Activity
- Fragment

Context

- Central command center
- Access application-specific data
  - Settings
  - Private files
  - Resources
  - Assets
- System services

Application

- A context
- Can be subclassed
  - Global data
  - Early initialization of libraries

Application

- Global data

```
public abstract class MyApplication extends Application {
    private static MyApplication instance;
    public static MyApplication getInstance() {
        return instance;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        instance = this;
        instance.initializeInstance();
    }

    private void initializeInstance() {
        // perform your initialization here
    }
}
```

Application

- Early Initialization

```
import org.acra.*;
import org.acra.annotation.*;

@ReportsCrashes(
    formUri = "http://www.backendofyourchoice.com/reportpath"
)
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // The following line triggers the initialization of ACRA
        ACRA.init(this);
    }
}
```

Application

- Android memory management
  - Garbage Collector
  - Upper limit for each Application
  - «Kill» activities when low on memory
  - Out-of-memory Exception

Application

- AndroidManifest.xml
  - Metadata about the app
  - Target SDK
  - «Entry point» of the app
  - Permissions, activities, services, receivers...
- Declare permission:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

Practical Work 1: Hello World!

- Launch Android Studio
- Create a new application
- Name it "USTH Weather"
- Package: vn.edu.usth.weather
- Run it
- 15 mins
  - It may take more time with Gradle dependencies

Activity

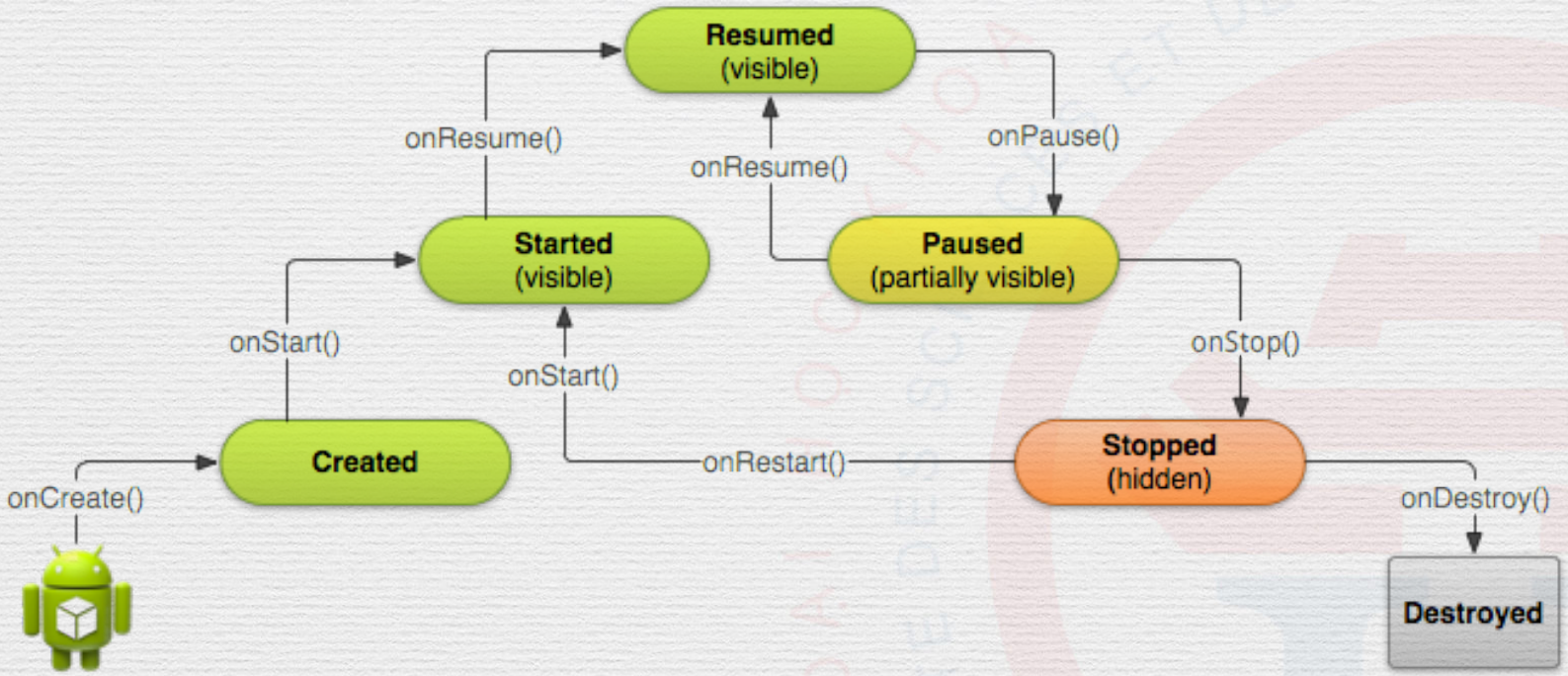
Controllers

- Context
- Application
- **Activity**
- Fragment

What is it?

- **Important!**
- Fundamental building block
- Has a unique task or purpose
- At least one per Application
- «Handles» display of single screen

Activity Lifecycle



Source: Android Developers

Activity Lifecycle

- `onCreate()`: initialization
  - Load view layout
  - Init view components

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Set the user interface layout for this Activity
    // The layout file is defined in the project res/layout/main_activity.xml fi
    setContentView(R.layout.main_activity);

    // Initialize member TextView so we can manipulate it later
    mTextView = (TextView) findViewById(R.id.text_message);
    mTextView.setText("Hello World!");
}
```

Activity Lifecycle

- `onPause()`
  - Stop animation or heavy tasks
  - Save unsaved changes
  - Release resources (e.g. camera)

```
@Override
public void onPause() {
    // Always call the superclass method first
    super.onPause();

    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release();
        mCamera = null;
    }
}
```

Activity Lifecycle

- onResume()
  - Called when activity comes to foreground
  - Acquire resources (e.g. camera)

```
@Override
public void onResume() {
    // Always call the superclass method first
    super.onResume();

    // Get the Camera instance as the activity achieves full user focus
    if (mCamera == null) {
        initializeCamera(); // Local method to handle camera init
    }
}
```

Activity Lifecycle: Screen orientation

- `onSaveInstanceState()`
- `onDestroy()`
- Create a new activity instance
- `onCreate()`
- `onRestoreInstanceState()`

Activity Lifecycle

- Close current activity: call `finish()`
- `onDestroy()` will be called if no memory leak

Intent

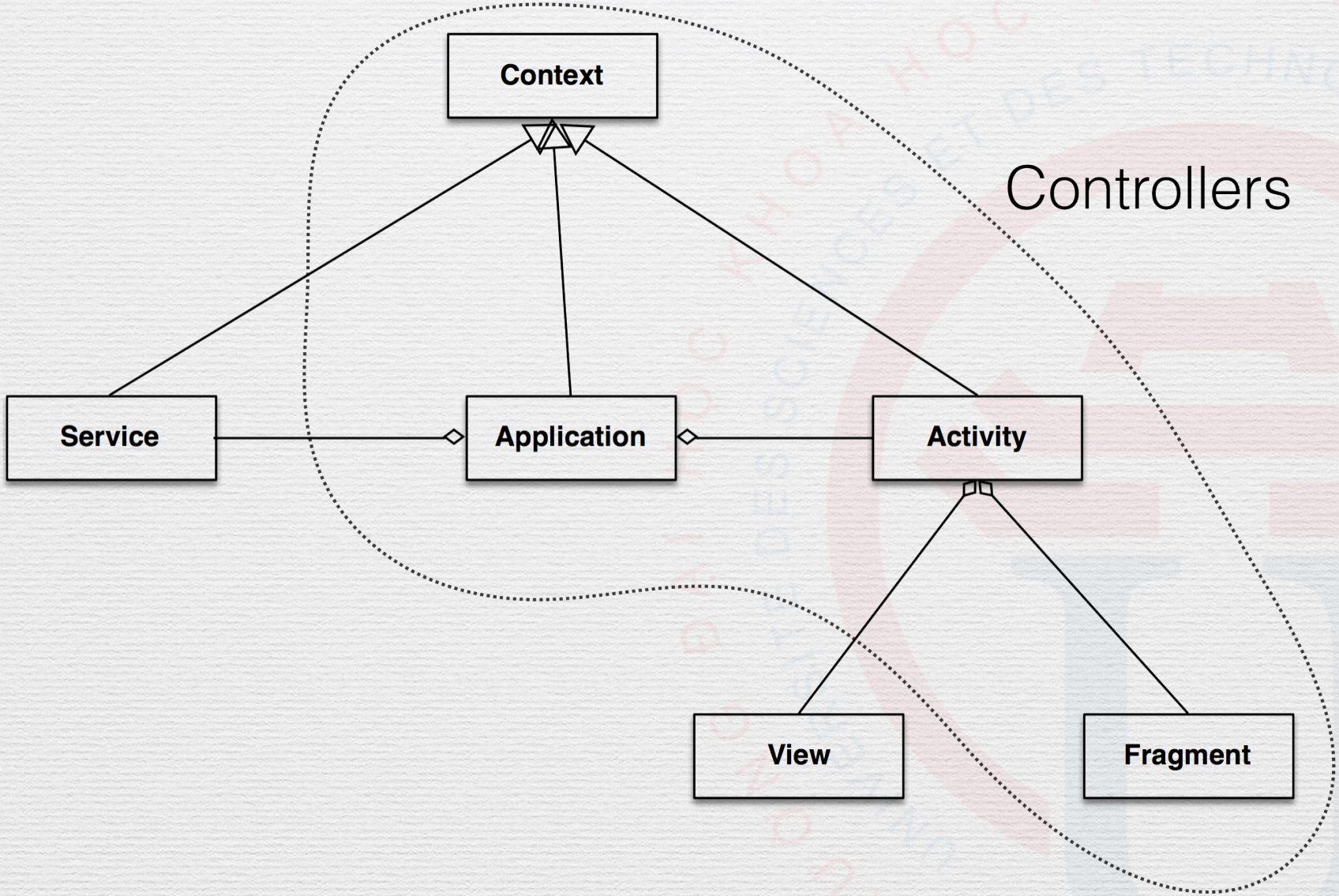
- Intent
  - Asynchronous messaging mechanism
  - Message to pass to other activities/services
  - Contains data
- Use intent to create Activity and pass parameters to it

```
Intent intent = new Intent(this, DisplayMessageActivity.class);
intent.putExtra("location", locationEditText.getText());
startActivity(intent);
```

Practical Work 2

- Create a new **empty** activity
  - `WeatherActivity`
- Remove the default MainActivity (don't forget manifest...)
- Override `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onDestroy()`
- Use `Log.i()` to output function traces
- Try running `WeatherActivity`, play with back/home/recent buttons and analyze your log

Remind: Android Application's Components

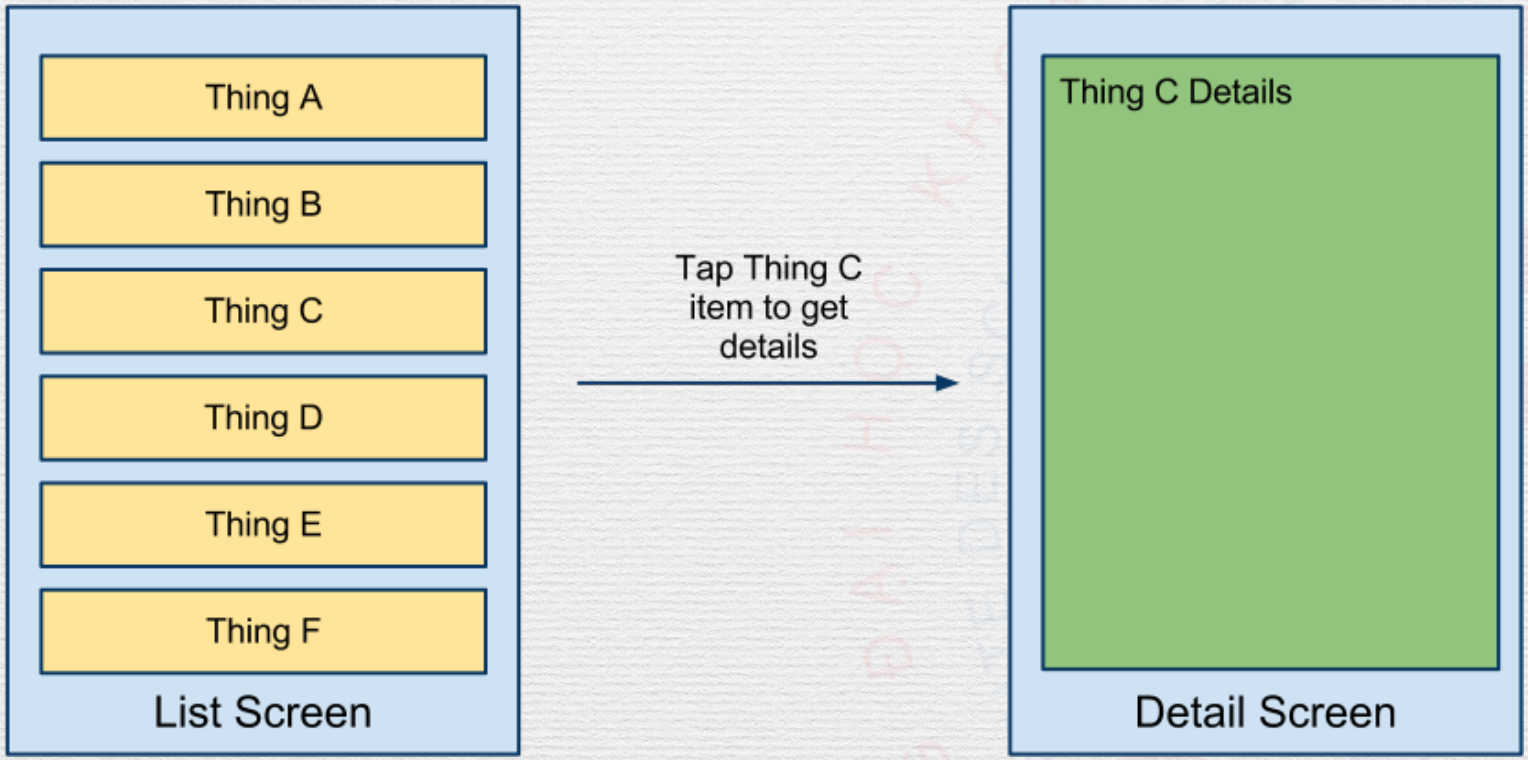


Fragment

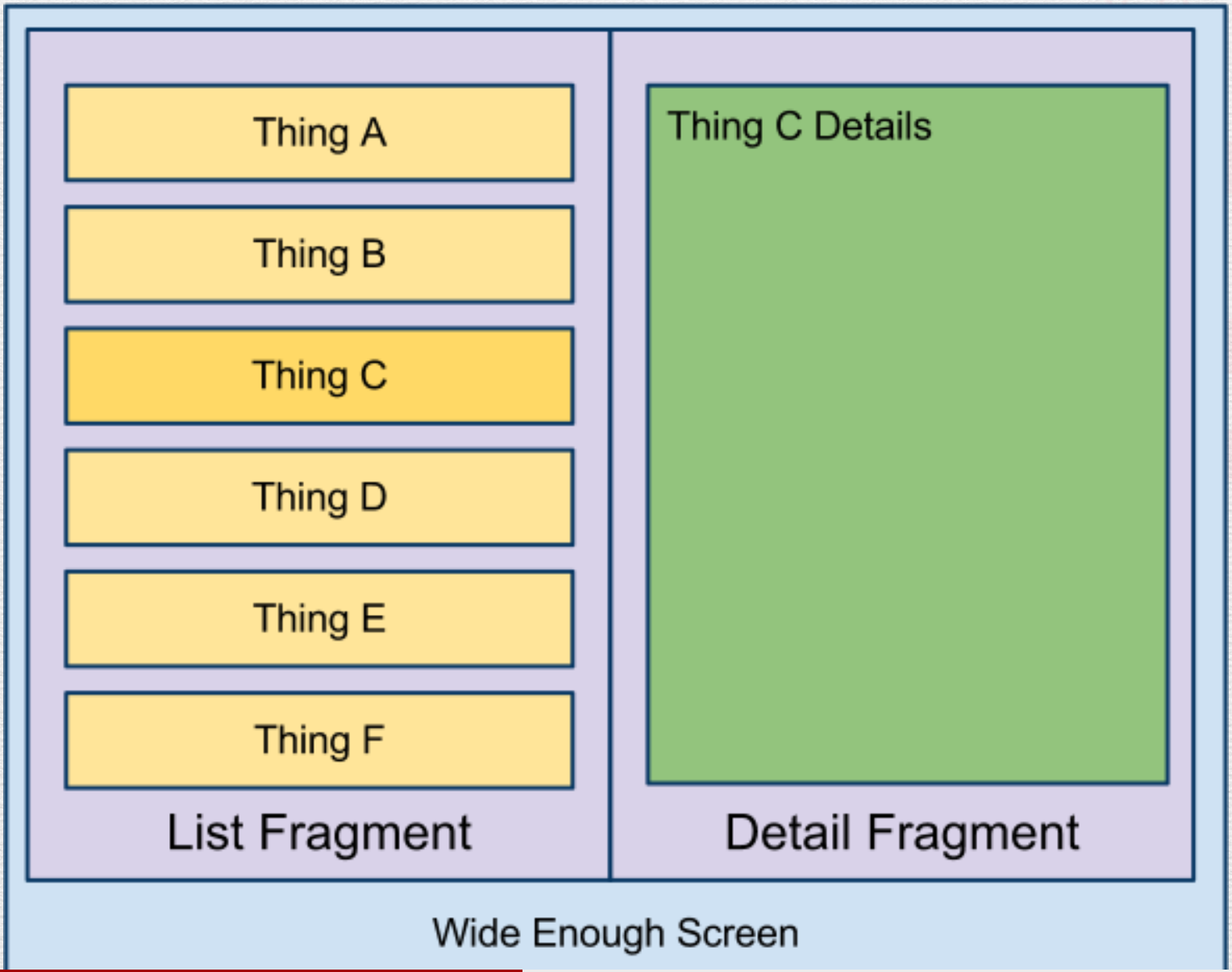
Controllers

- Context
- Application
- Activity
- **Fragment**

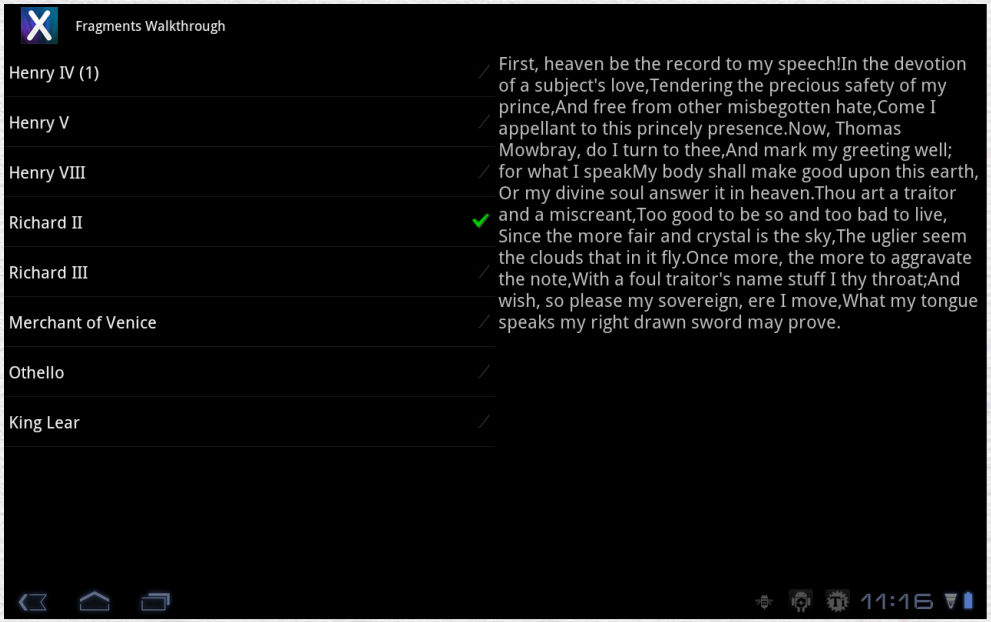
User Experience Example



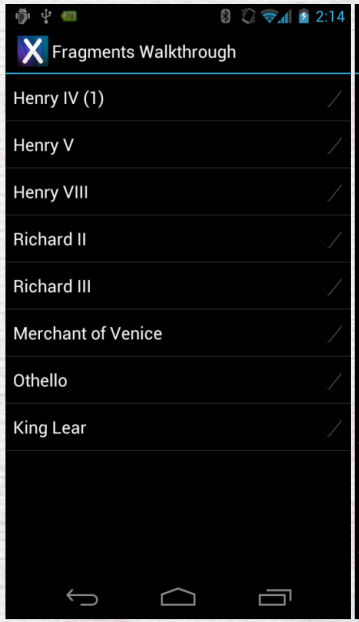
User Experience Example

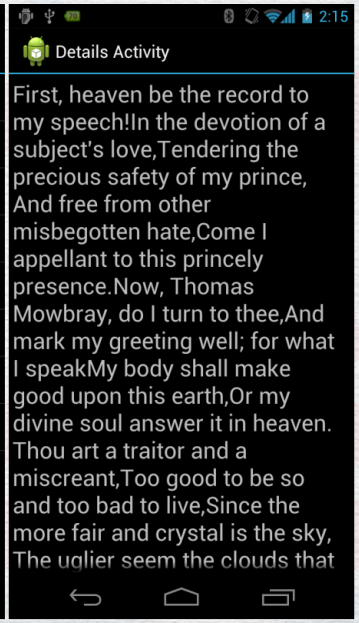


User Experience Example



Tablet





Phone

Source: developer.xamarin.com

Fragment

- Why fragment?
  - Explosion in the variety of devices
  - Screen size differs
  - Screen resolution differs
  - Screen density differs
  - Screen orientation differs

Fragment

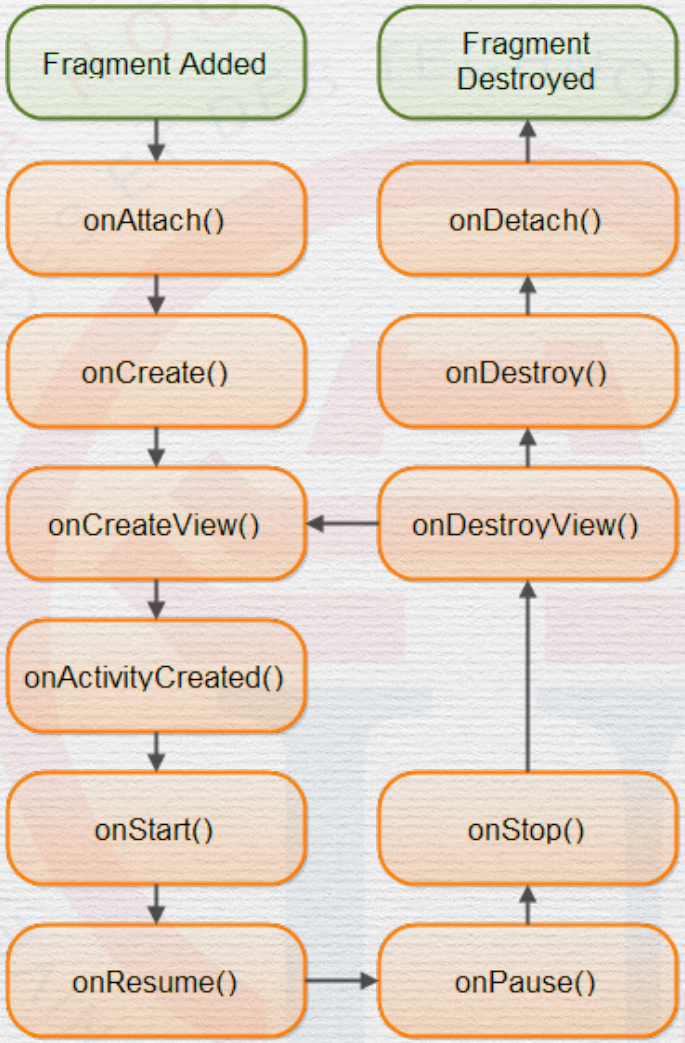
- Fragment...
  - represents a behavior or a portion of user interface
  - is building block of the Fundamental building blocks
  - is optional
  - is officially supported from Honeycomb [API 11]

Fragment

- Can be used from devices as low as Donut [API 4]
  - Android Support Library
- «Native» from Honeycomb onward

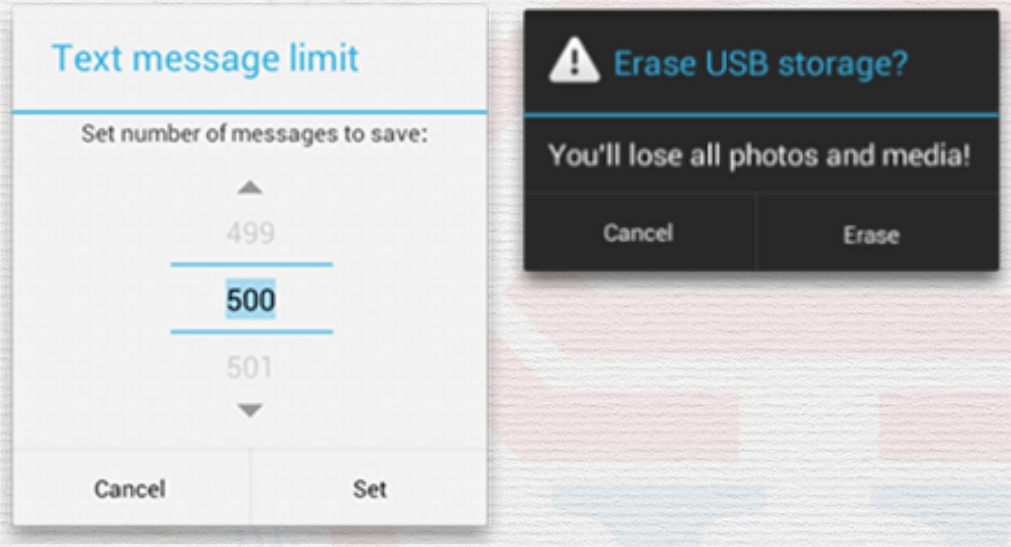
Fragment

- Similar lifecycle as Activity
  - `onCreate()`: initialization
  - `onCreateView()`: view init
  - `onPause()`: user leaves



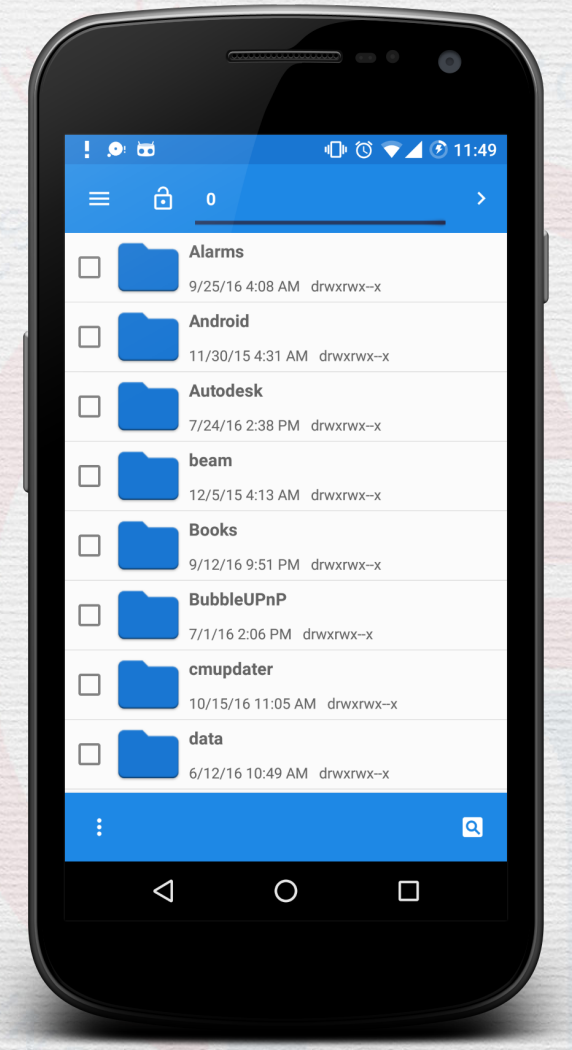
Popular Fragment Classes

- DialogFragment



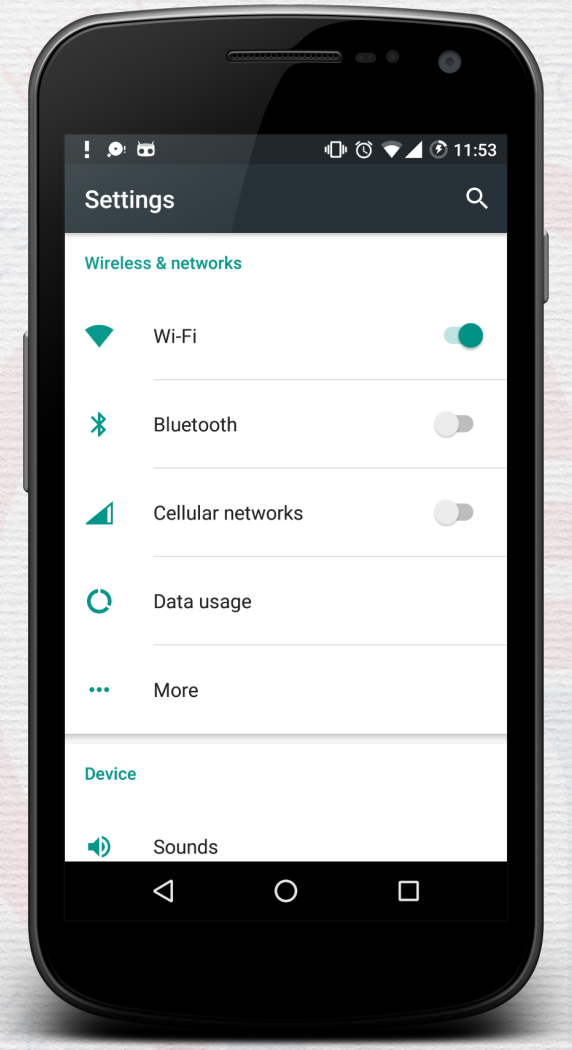
Popular Fragment Classes

- ListFragment



Popular Fragment Classes

- PreferenceFragment



Fragment

- Activity with fragments
  - is simplified
  - coordinates fragments
  - [optionally, but mostly] uses FragmentManager (or SupportFragmentManager)

Fragment

- Put inside a layout XML
- Dynamically created using codes

```
// Create a new Fragment to be placed in the activity layout
DetailFragment firstFragment = new DetailFragment();

// Add the fragment to the 'container' FrameLayout
getSupportFragmentManager().beginTransaction().add(
    R.id.container, firstFragment).commit();
```

Practical Work 3

- Create one Fragment with empty layout
  - `ForecastFragment`
- Set background to #20FF0000 and #2000FF00 and #200000FF (or whatever...)
  - Override `onCreateView`, get the returned View
  - Use `view.setBackgroundColor(color)`
- Add to your previous WeatherActivity using dynamic code

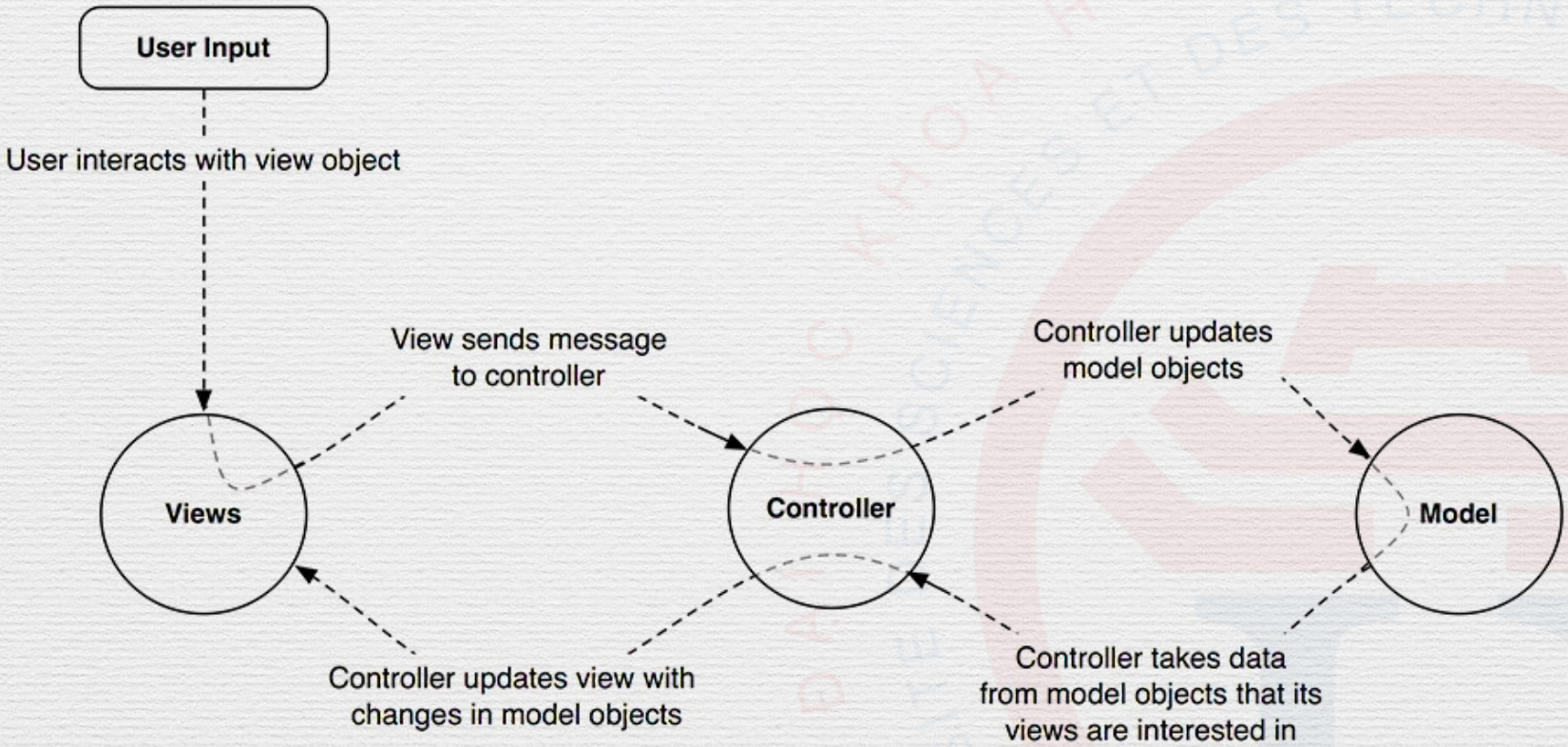
```
// Create a new Fragment to be placed in the activity
DetailFragment firstFragment = new DetailFragment();

// Add the fragment to the 'container' FrameLayout
getSupportFragmentManager().beginTransaction().add(
        R.id.container, firstFragment).commit();
```

View

Content

- Architecture
- Compilation
- Controllers: Context, Application, Activity, Fragment
- **View**

View

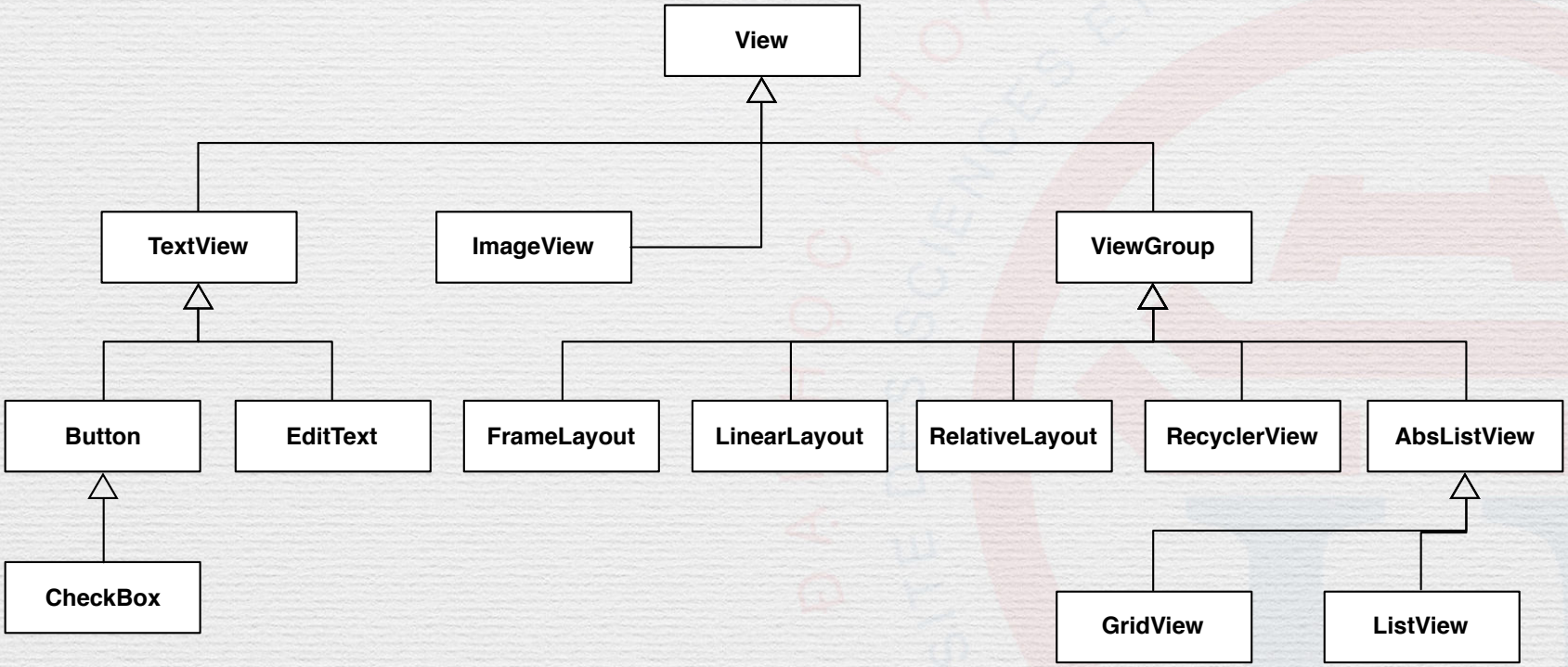


Android Programming: The Big Nerd Ranch Guide, 2nd Edition

View

- What user interacts with
- Basic building blocks of user interface
- android.view.*
- android.widget
- XML or dynamic code

Popular Views



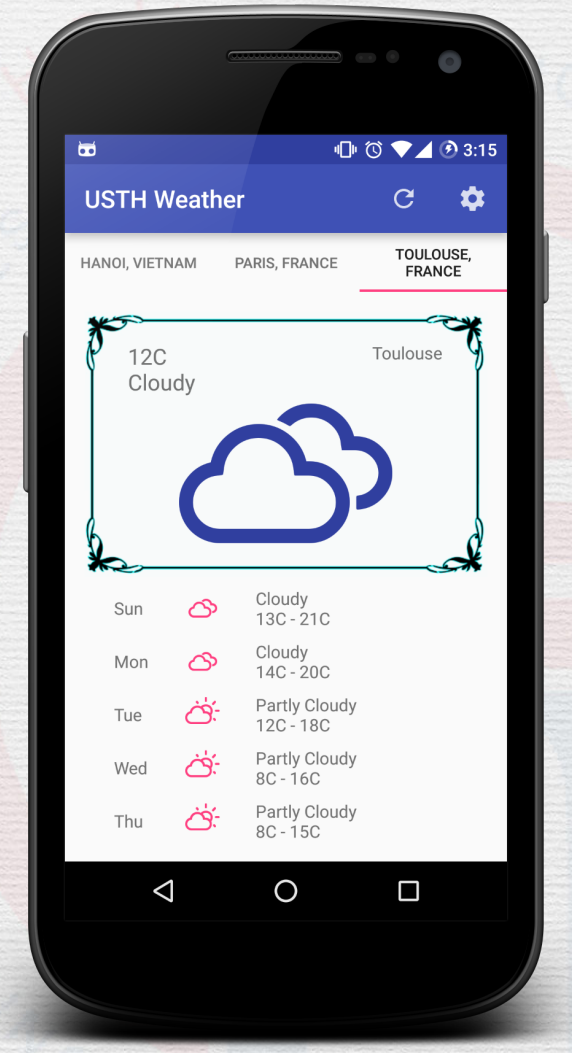
Attributes

- id: `findViewById()`
- width
- height
- padding
- margin
- visibility
- alpha
- rotation
- background
- click

Padding and Margin

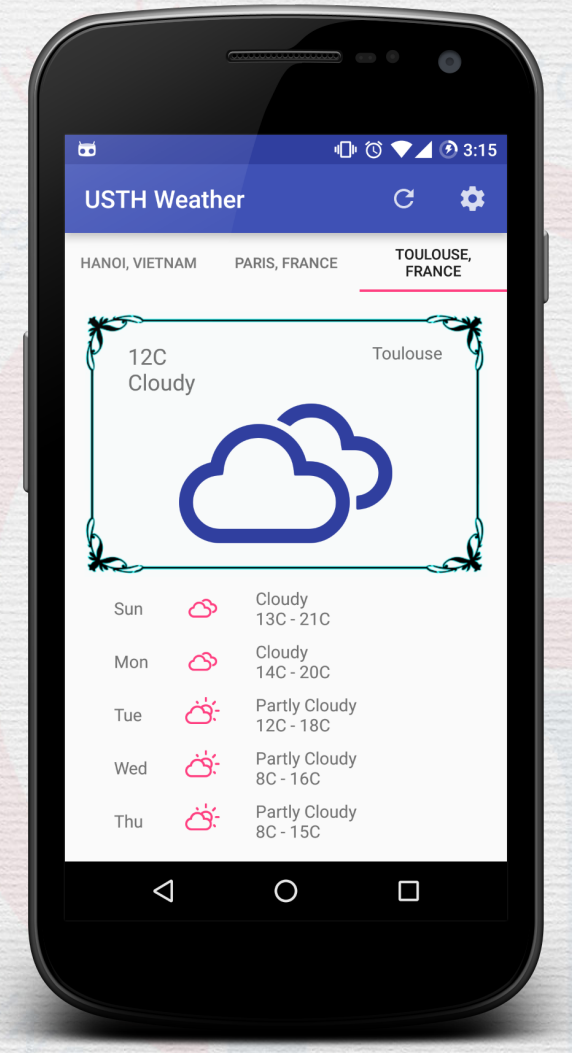
TextView

- text: setText()
- drawable
- font
- gravity
- styles



ImageView

- src: `setImageResource()`
- scaleType: fitXY, fitStart, fitEnd, centerCrop, centerInside
- tint
- crop
- viewBounds



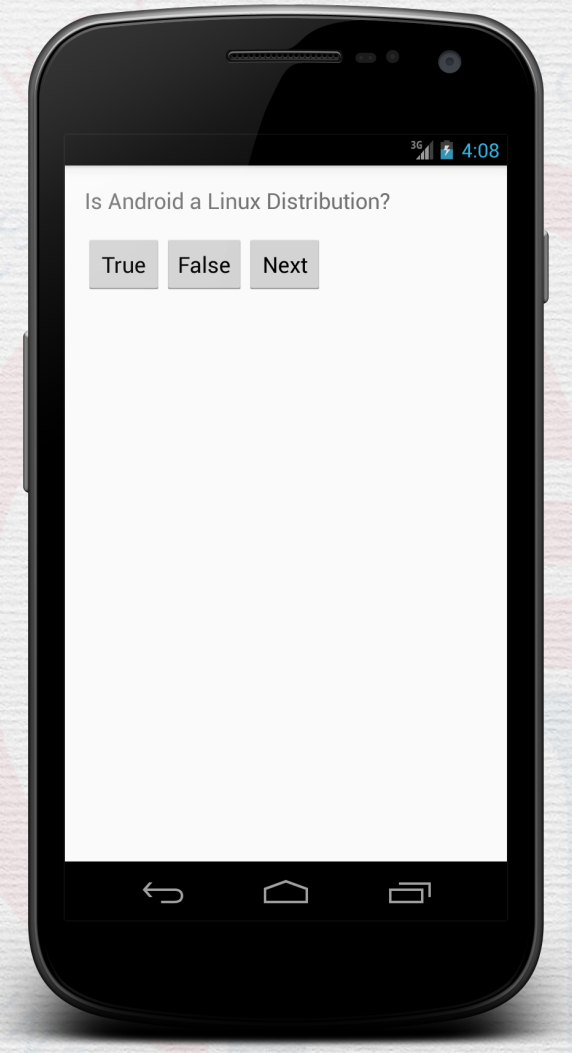
ViewGroup

- **Important.**
- Contain children (other Views)
- LayoutParams
- Important subclasses: FrameLayout, LinearLayout, RelativeLayout, AbsListView...
- Layouts will be discussed in next lecture (Resources)

Button

- Push-button
- State-list (later)
- onClick()

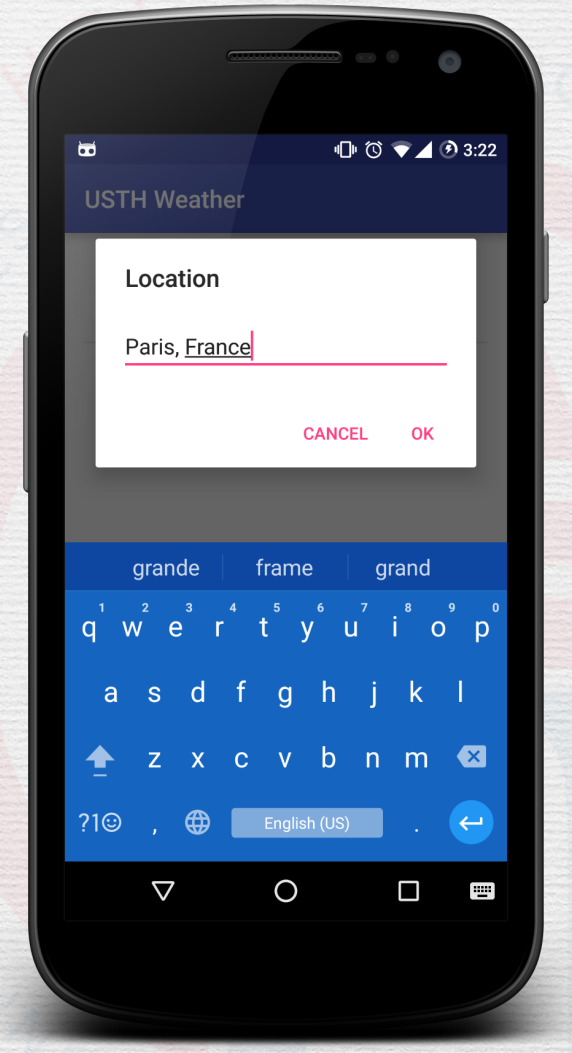
```
button.setOnClickListener(
    new View.OnClickListener() {
        public void onClick(View v) {
            // Do something in response
            // to button click
        }
    }
);
```



EditText

- Allow editing a text (TextBoxes)
- getText()
- Selection

```
button.setOnClickListener(
    new View.OnClickListener() {
        public void onClick(View v) {
            location.setText(editText.getText());
        }
    }
);
```



Practical Work 4

- Find and download a weather icon set
  - Preferable not too big, 144x144 - 256x256
  - Don't have names starting with digits
  - Names contain lowercase letters, digits and underscores
- Put them inside res/drawable-hdpi (why? later.)
- Use code to create dynamic views in your Fragment:
  - A vertical LinearLayout (use LinearLayout's `setOrientation`), containing:
    - `ForecastFragment`: TextView (Thursday) and ImageView (a weather icon)