ANDROID: THE BASICS

COMP 41690

DAVID COYLE > D.COYLE@UCD.IE

GUIDING PRINCIPLES

Android™ delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.

Open

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open.

For example, an application can call upon any of the phone's core functionality such as making calls, sending text messages, or using the camera, allowing developers to create richer and more cohesive experiences for users.

GUIDING PRINCIPLES

Android™ delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.

All applications are created equal

Android does not differentiate between the phone's core applications and third-party applications.

They can all be built to have equal access to a phone's capabilities providing users with a broad spectrum of applications and services.

GUIDING PRINCIPLES

Android™ delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.

Breaking down application boundaries

Android breaks down the barriers to building new and innovative applications.

For example, a developer can combine information from the web with data on an individual's mobile phone — such as the user's contacts, calendar, or geographic location — to provide a more relevant user experience.

GUIDING PRINCIPLES

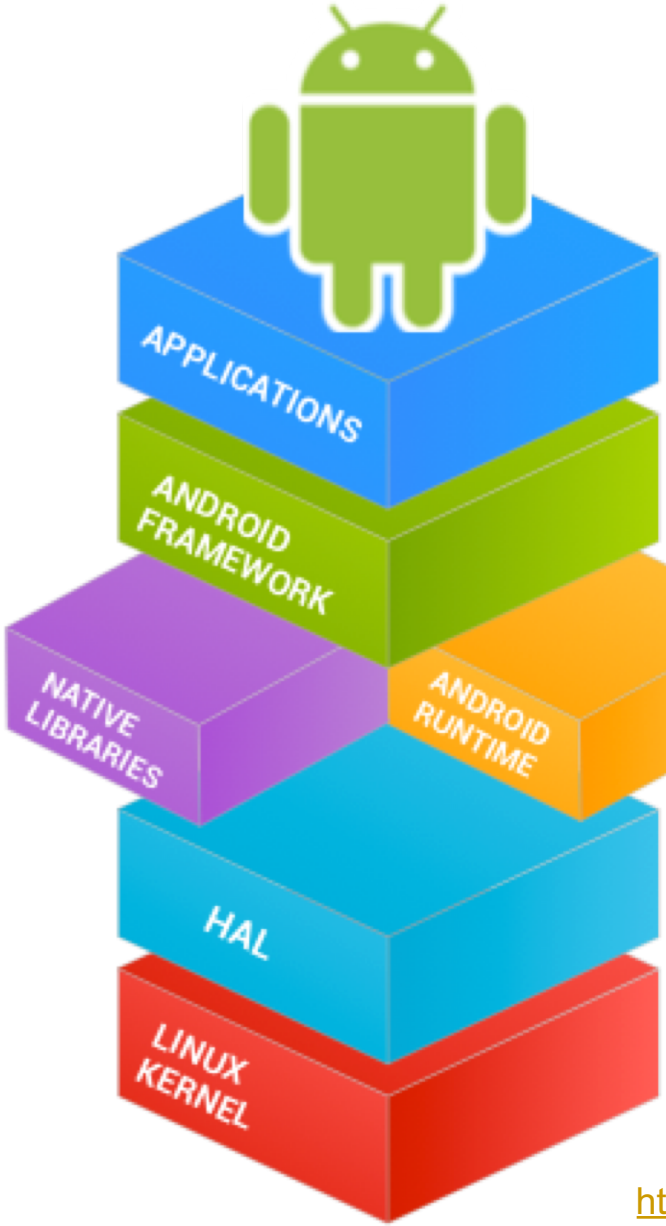
Android™ delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications.

Fast & easy application development

Android provides access to a wide range of useful libraries and tools that can be used to build rich applications. For example, Android enables developers to obtain the location of the device, and allows devices to communicate with one another enabling rich peer–to–peer social applications.

In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications.

ANDROID STACK



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

CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM

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

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

https://source.android.com/source/index.html

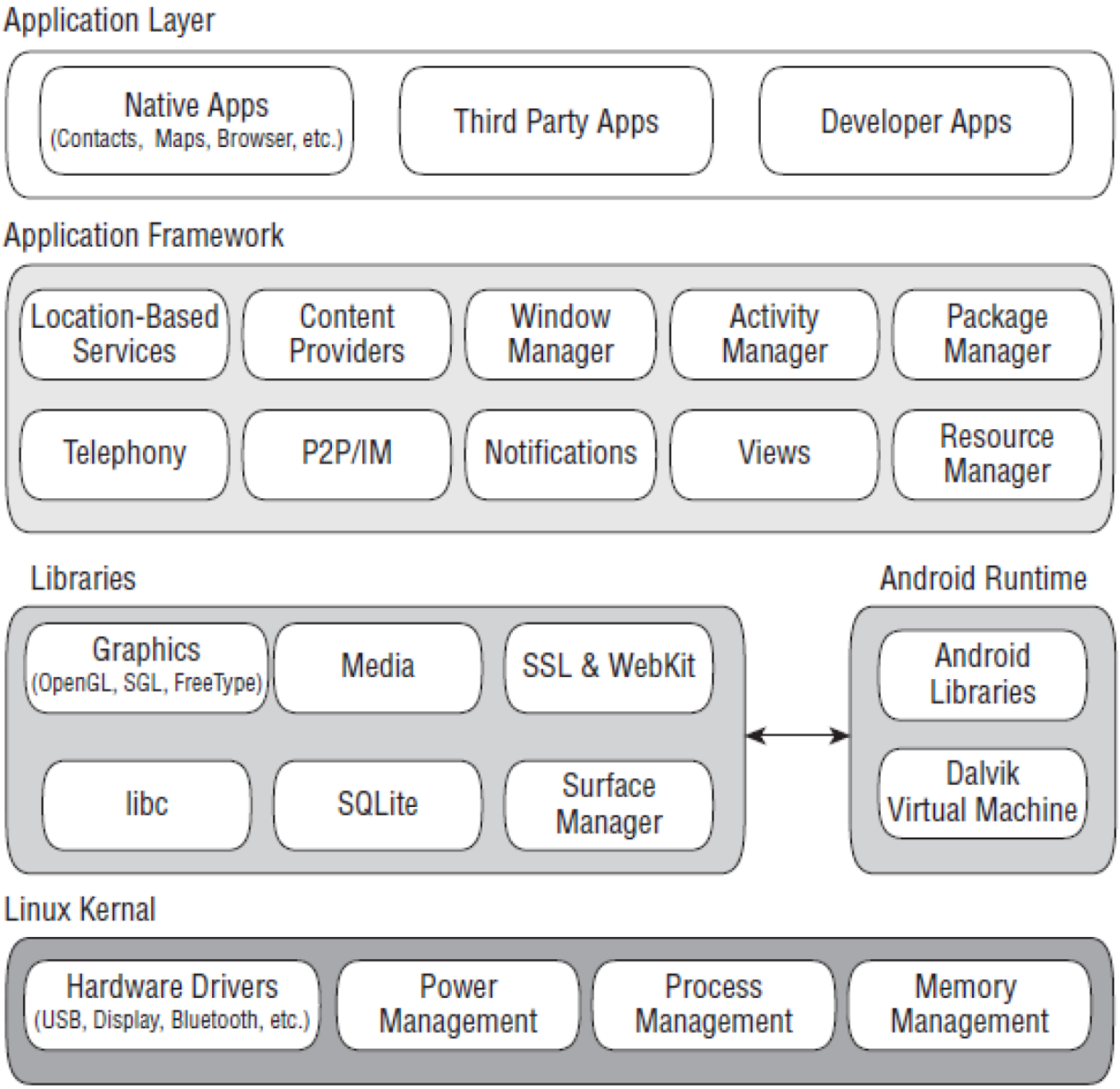


Figure 1-1

LINUX KERNEL

It is the heart of android architecture that exists at the root of android architecture.

Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

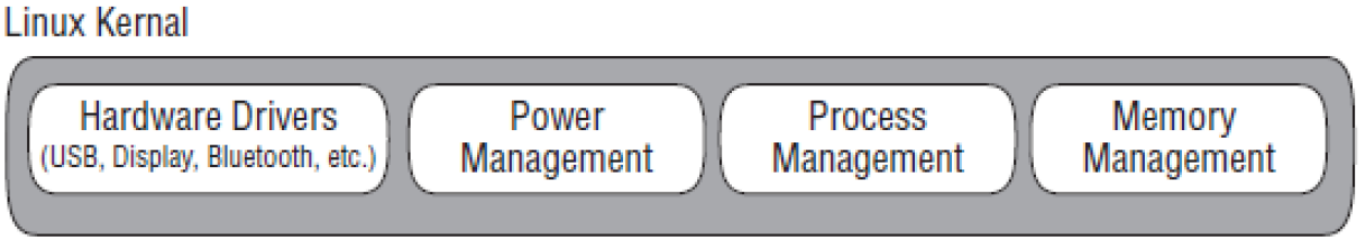


Figure 1-1

NATIVE LIBARIES

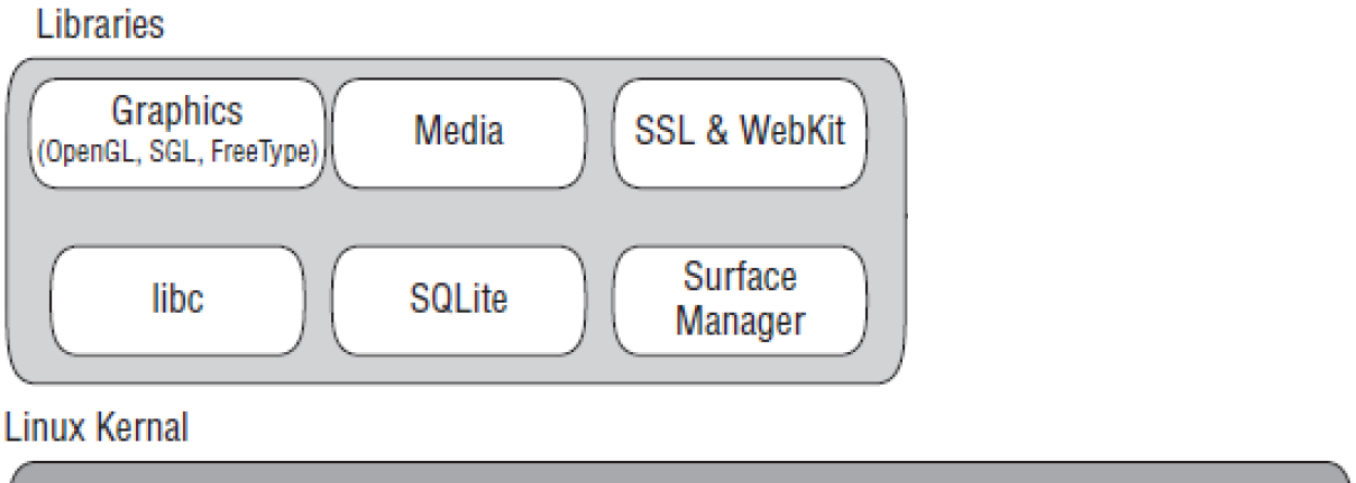
On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support

SQLite is for database,

FreeType for font support,

Media for playing and recording audio and video formats.

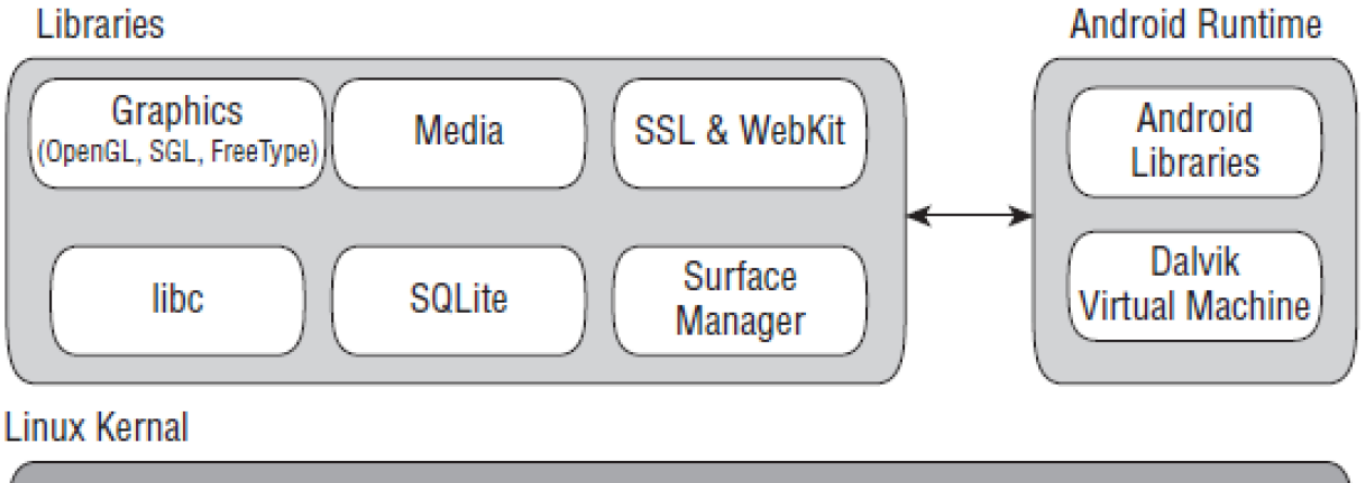


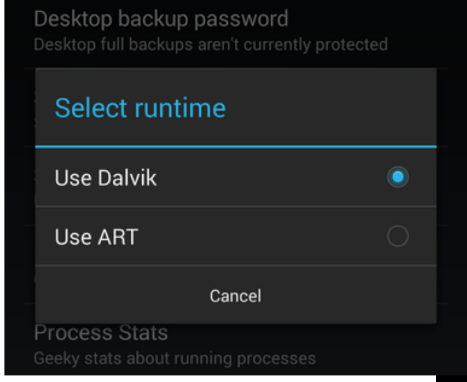
ANDROID RUNTIME

Android runtime has core libraries and VM (Virtual Machine) which is responsible to run android application.

Android uses a custom virtual machine, Dalvik or ART, to run Java-based applications. It is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

Dalvik uses a custom bytecode format which is different from Java bytecode. Therefore you cannot run Java class files on Android directly; they need to be converted into the Dalvik bytecode format.



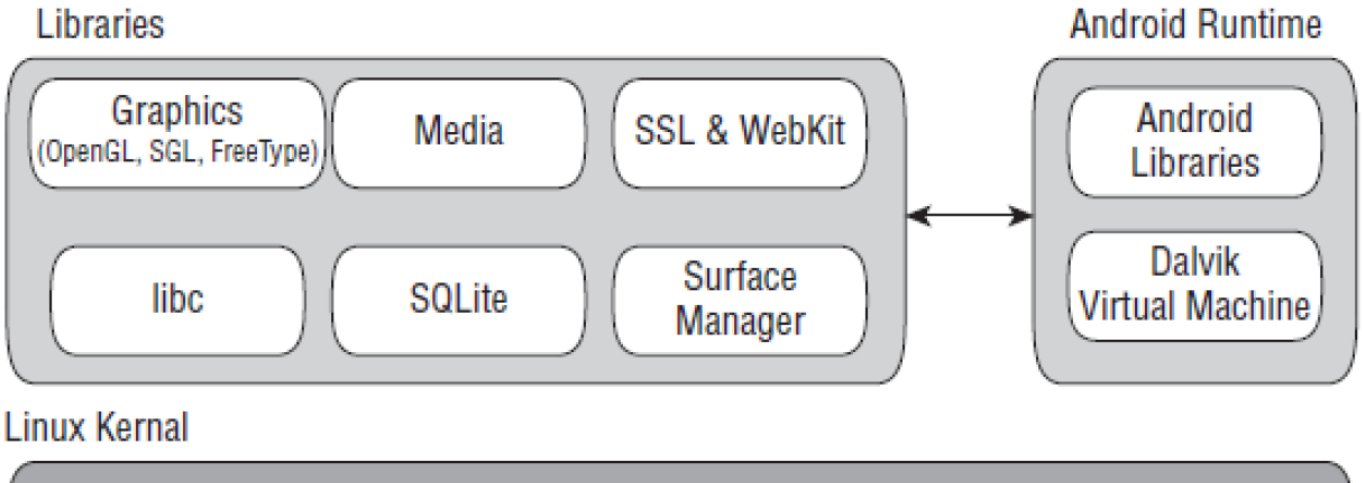


ANDROID RUNTIME

"ART is a new Android runtime being introduced experimentally in the 4.4 release KitKat. This is a preview of work in progress in KitKat. It is available for the purpose of obtaining early developer and partner feedback." - Android developers

Android RunTime (ART) uses Ahead Of Time compilation, and optional runtime for Android 4.4.

More details on ART and Dalvik:
https://source.android.com/devices/tech/dalvik/
https://www.youtube.com/watch?v=USgXkI-NRPo
https://infinum.co/the-capsized-eight/articles/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat

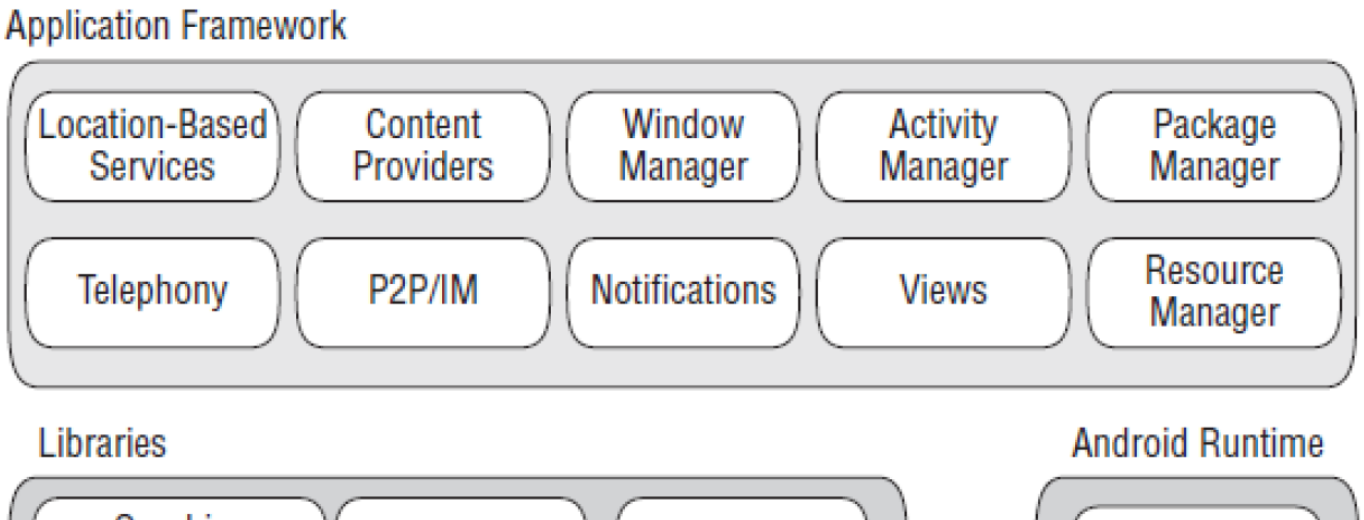


ANDROID FRAMEWORK

On the top of libraries and Android Runtime, there is Android Framework.

Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers.

It provides the classes and interfaces for Android Application development.



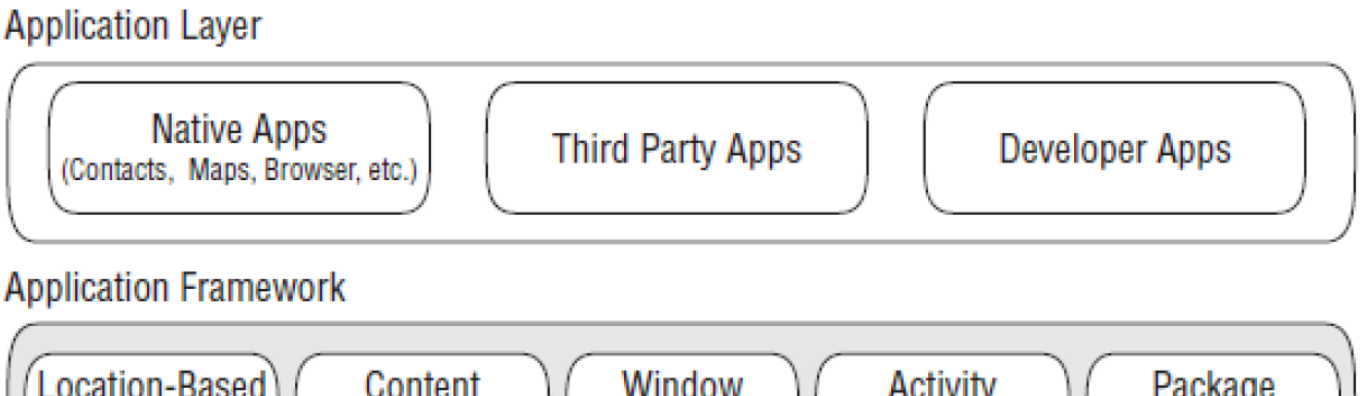
APPLICATIONS

On the top of Android Framework, there are applications.

Android apps are written in the Java programming language.

The Android SDK tools compile your code - along with any data and resource files - into an APK: an Android package, which is an archive file with an .apk suffix.

One APK file contains all the contents of an Android app and is the file that Android-powered devices use to install the app.



APPLICATION SECURITY

The Android system implements the *principle of least privilege*.

Once installed on a device, each Android app lives in its own security sandbox:

- The Android operating system is a multi-user Linux system in which each app is a different user. Security is managed by the kernel.
- By default, the system assigns each app a unique Linux user ID. The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them.
- Each process has its own virtual machine (Dalvik VM), so an app's code runs in isolation from other apps.
- By default, every app runs in its own Linux process. Android starts the process when any of the app's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other apps.

The processes are isolated and crash of one app does not bring down the whole system!

APPLICATION SECURITY

There are ways for an app to share data with other apps and for an app to access system services:

- It's possible to arrange for two apps to share the same Linux user ID, in which case they are able to access each other's files.
- To conserve system resources, apps with the same user ID can also arrange to run in the same Linux process and share the same VM (the apps must also be signed with the same certificate).

- An app can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more.
- All such app permissions must be granted by the user at install time.

ANDROID SOFTWARE STACK : A FINAL WORD

Android 2.x had more than 12 mln lines of code.

This included (in mln)

- 3 XML
- 2.8 C
- 2.1 Java
- 1.75 C++

New versions have similar amounts and have even optimised the code to be smaller.

BASIC BUILDING BLOCKS

Activities

Content Providers

Services

Broadcase receivers

Intents

https://developer.android.com/guide/components/fundamentals.html

ACTIVITIES

An activity represents a single screen with a user interface. They are the building blocks of the user interface.

For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.

Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities (if the email app allows it). For example, a camera app can start the activity in the email app that composes new mail, in order for the user to share a picture.

Breaking down application boundaries

CONTENT PROVIDERS

A content provider manages a shared set of app data. They provide a level of abstraction for any data stored on the device that is accessible by multiple applications.

You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access.

The Android development model encourages you to make your own data available to other applications. Building a content provider lets you do that, ***while maintaining complete control over how your data is accessed***.

For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query part of the content provider to read and write information about a particular person.

SERVICES

A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface.

For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity.

Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

BROADCASE RECEIVERS

A broadcast receiver is a component that responds to system-wide broadcast announcements.

Many broadcasts originate from the system - for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.

Apps can also initiate broadcasts - for example, to let other apps know that some data has been downloaded to the device and is available for them to use.

Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work.

INTENTS

Intents are system messages, running around the inside of the device, notifying applications of various events, from hardware state changes, to incoming data, to application events.

Not only can you respond to intents, but you can create your own to launch other activities or to let you know when specific situations arise.

BASICS IN THE NUTSHELL

Activity: basic building block of an application

Intents: communication mechanism

Service: background process with no user interface

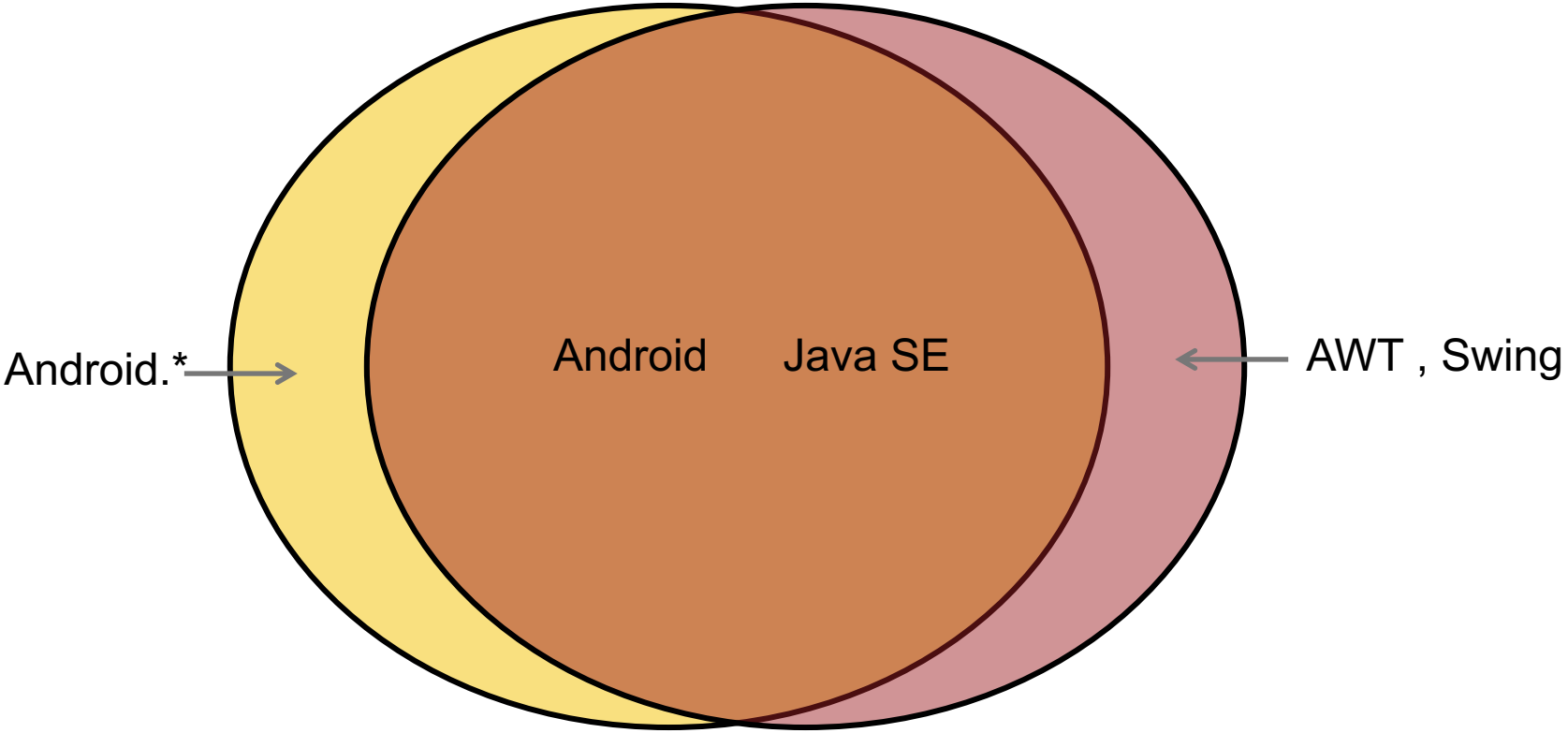
Content provider: basic superclass framework for handling and storing data

ANDROID VS JAVA

Android is a mobile operating system

Java is a programming language for various platforms

Android programming is done with Java compatible libraries



ANDROID.*

android.util – containers, formatters, parsers

android.os – message passing, IPC, debugging

android.graphics – canvas, colour, primitives, ...

android.text – displaying & parsing text

android.database – handling cursors for db

android.content – data access and publishing

android.view – core user interface

android.widget – lists, buttons, layouts...

ANDROID.* CONT'D

com.google.android.maps – map controls

android.app – activity and service API

android.provider – standard content provider

android.telephony – telephony API

android.webkit – web-based content work

Also: OpenGL, FreeType, SGL, libc, SQLite, SSL

ANDROID.* CONT'D

android.location

android.media

android.opengl

android.hardware

android.bluetooth

android.net.wifi

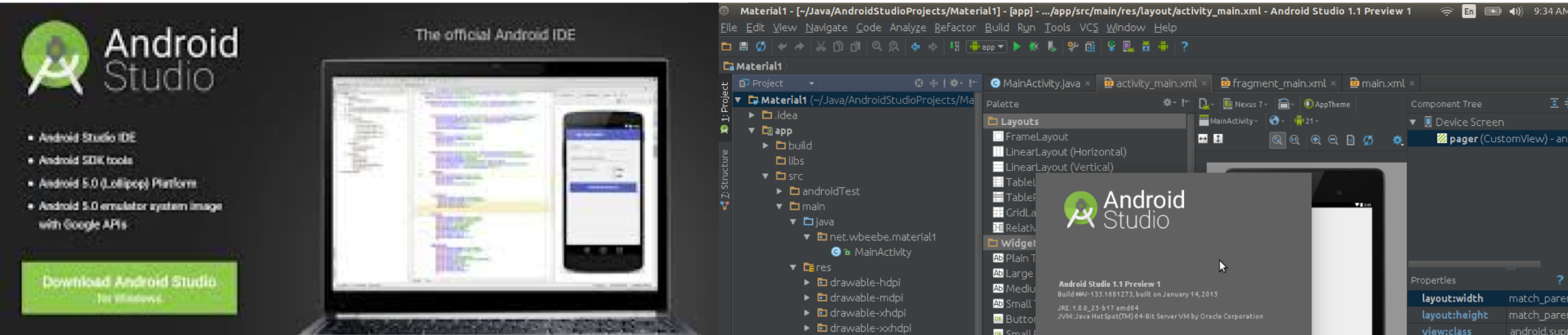
DEVELOPING APPS

Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications

Android debug bridge (adb), which is a tool that allows you to connect to a virtual or real Android device

Google provides two integrated development environments (IDEs) to develop new applications.

- Android Developer Tools (ADT) are based on the Eclipse IDE
- Android Studio based on the IntelliJ IDE



https://developer.android.com/ndk/index.h

Android NDK

The Android NDK is a toolset that lets you implement parts of your app using native-code languages such as C and C++. For certain types of apps, this can help you reuse existing code libraries written in those languages.

> Get Started

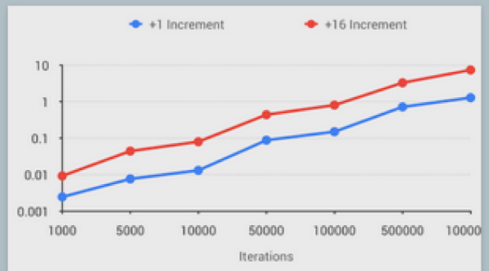
```
public class MyActivity extends Activity {
  /**
   * Native method implemented in C/C++
   */
  public native void computeFoo();
}
```

Latest

BLOG

Get your hands on Android Studio 1.3

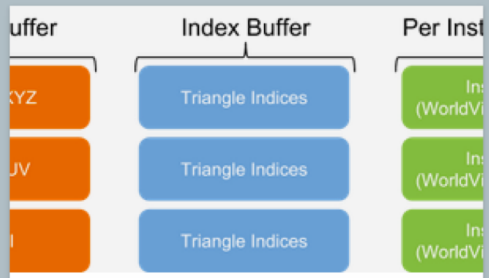
Android Studio 1.3 is our biggest feature release for the year so far, which includes ...



BLOG

Game Performance: Data-Oriented Programming

To improve game performance, we'd like to highlight a programming paradigm that wi ...



BLOG

Game Performance: Geometry Instancing

Rendering a lot of meshes is desired to create a beautiful scene like a forest, a ...

PRACTICAL

Download and install Android Studio and add recommended SDK packages

https://developer.android.com/sdk/index.html

Read the introduction to Android Studio

https://developer.android.com/tools/studio/index.html

Complete the tutorial “Building Your First App”

https://developer.android.com/training/basics/firstapp/index.html

QUESTIONS?

Contact:

d.coyle@ucd.ie

Please ask in the Discussion Forum.

Next week:

Activities and Fragments