Θέμα: Android development

Γιάννης Παππάς
Μονάδα Αριστείας ΕΛ/ΛΑΚ | 11/5/2015

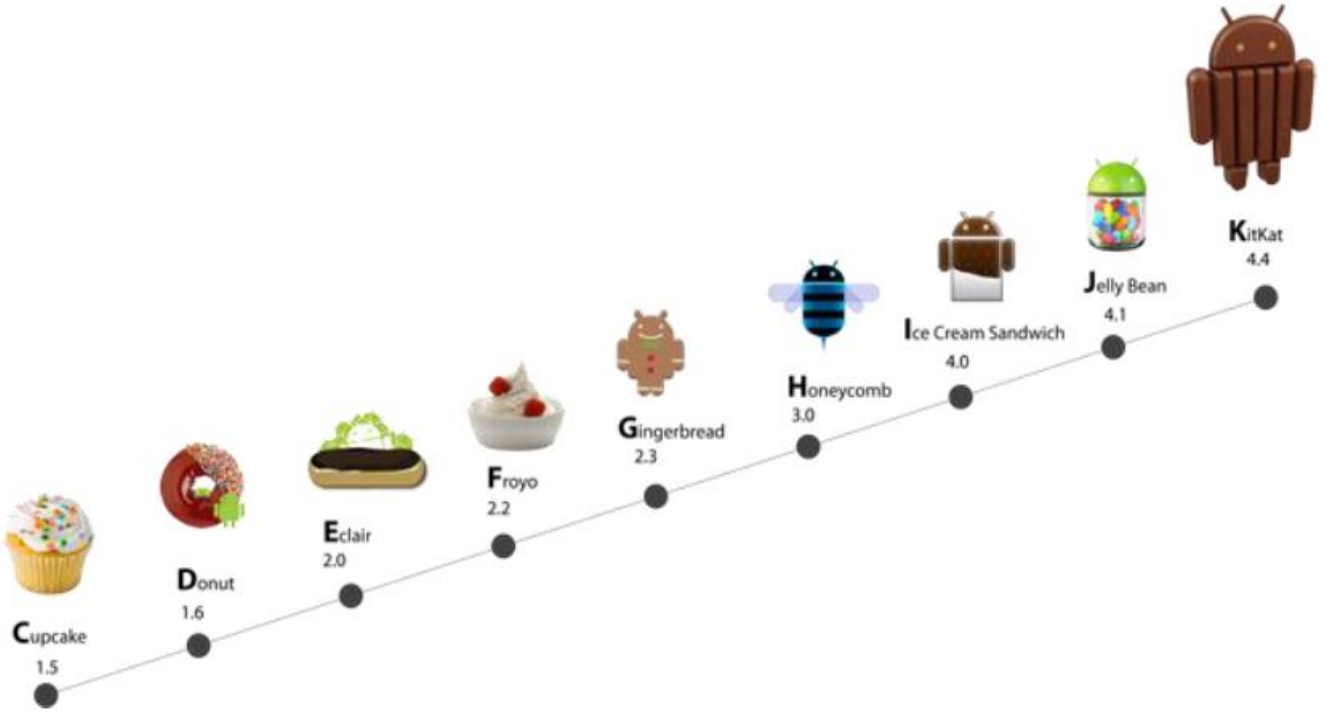
Σχεδιάγραμμα της παρουσίασης

- Android OS
- Development Tools
- Development Overview
- A Simple Activity with Layout
- Some Pitfalls to Avoid
- Useful Apps and Libraries

Στόχος της παρουσίασης

- Provide a high level overview on how both Android apps operate and how they are developed.
  - Required tools and their purpose.
  - Design and development considerations to keep in mind.
- Provide a starting point for experimenting and research with Android development.
  - Aspects of Android development that aren't likely to be encountered early on will not be covered in detail.
  - Technical specifics will also not be covered in depth either but followup research is encouraged.

Android versions



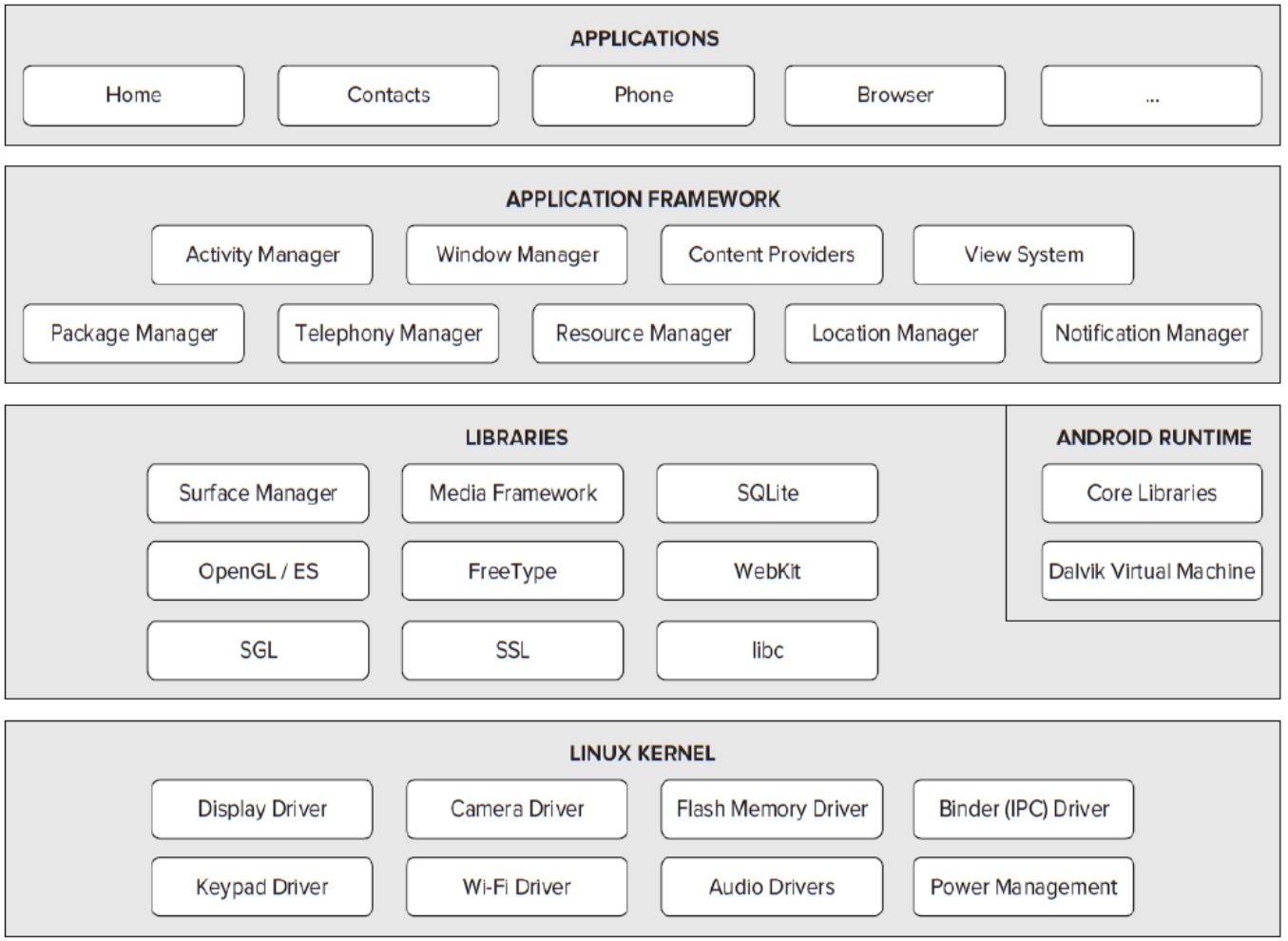
Features of Android

- **Storage** — Uses SQLite, a lightweight relational database, for data storage.
- **Connectivity** — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), Wi-Fi, LTE, and WiMAX.
- **Messaging** — Supports both SMS and MMS.
- **Web brow**ser — Based on the open source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
- **Hardware suppor**t — Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
- **Multi-touch** — Supports multi-touch screens
- **Multi-tasking** — Supports multi-tasking applications
- **Flash support** — Android 2.3 supports Flash 10.1.
- **Tethering** — Supports sharing of Internet connections as a wired/wireless hotspot.

Architecture of Android

- Linux kernel — This is the kernel on which Android is based. This layer contains all the low level device drivers for the various hardware components of an Android device.
- Libraries — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.
- Android runtime — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the **Dalvik virtual machine**, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into Dalvik executables).
- Application framework — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.
- Applications — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

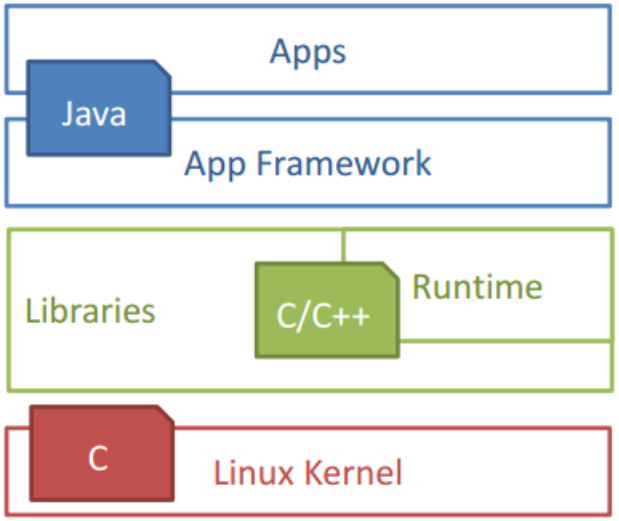
Architecture of Android



Architecture of Android

- Important blocks:
  - Activity Manager: Manages the activity life cycle of applications
  - Content Providers: Manage the data sharing between applications
  - Telephony Manager: Manages all voice calls. We use telephony manager if we want to access voice calls in our application.
  - Location Manager: Location management, using GPS or cell tower
  - Resource Manager: Manage the various types of resources we use in our Application

Architecture of Android



- Design goals
  - Open Source
  - High flexibility
  - High data accessibility
  - Rapid development (XML, Java)
- Used Languages
  - App: Java
  - Framework: Java
  - Libraries: C/C++
  - OS & Drivers: C

The Android Developer Community

- Stack Overflow (www.stackoverflow.com) — Stack Overflow is a collaboratively edited question and answer site for developers.
- Google Android Training (http://developer.android.com/training/index .html) — Google has launched the Android Training site that contains a number of useful classes grouped by topics.
- Android Discuss (http://groups.google.com/group/android-discuss) — Android Discuss is a discussion group hosted by Google using the Google Groups service.

Android OS

OS Internals

- Android is Linux-based although apps are usually developed against APIs that abstract anything Linux-specific.
  - The OS mainly provides a platform to run instances of the DVM (Dalvik Virtual Machine).
    - Each Android app runs as its own user, in its own process, in its own DVM instance.
    - In Android 4.4, Google introduced ART (Android Run Time) which will potentially replace Dalvik in the long term.
- For a typical app developer, being Linux-based is mostly an implementation detail.

Developer's Perspective

- Android apps are developed in Java using the Android API.
  - It's possible to develop natively in C/C++ with some caveats but it's generally not recommended.
- Dalvik's class libraries will be mostly familiar to Java SE developers but there are some differences.
- Android's class library is based on Apache Harmony 6 which is mostly compatible with Java SE 6.

Development Tools

Which IDE?

- Eclipse + ADT (Android Developer Tools) was the primary IDE for Android development.
- Android Studio is a new IDE that is being actively developed and is available for download now.
  - Still an early access preview but it addresses many stability and usability issues of Eclipse.
  - v0.4.6 is available for download from Google and newer (albeit potentially less stable) versions are available from the Canary channel.
  - New versions are released rapidly if you prefer to be on the bleeding edge or have issues with your current version.

Android SDK Manager

- Used to install or update the Android API SDKs, libraries and the Android build tools used within the IDE itself.
- You'll use it mainly to install files needed to develop and test for a given Android API.
  - Not all of them are bundled in Android Studio!

AVDM & Emulator

- Android Virtual Device Manager
  - Used to manage a list of your virtual Android devices for use with the Android Emulator.
- Define screen dimensions, Android API version, memory size and other hardware specifics.
- Test your Android app without requiring a matching phone with matching API version.

Layout Editor

- WYSIWYG editor used for editing views and their properties for a given layout.
- Layouts, like most other Android resources, are serialised to XML files.
  - They can be edited directly if you prefer; it's up to your personal preference.
  - Both methods provide an up-to-date visual preview.

Layout Editor

- Allows you to preview layouts with a given screen size, API version and theme.
  - Observe what happens when your layout is previewed in landscape, on a smaller screen, with a different aspect ratio etc.
- Try to do as much as possible in the layout editor and keep programmatic layout code to a minimum.

Development Overview

The Activity Class

- An Activity usually manages a single screen's behaviour.
- Activity instances initialise themselves in the onCreate() method.
  - Main initialisation task is to call setContentView() to initialise the UI with a given layout.
- Contains methods to handle user interaction with the layout.

The Activity Class

- Various lifecycle callback methods are called when the activity's state changes (onStart(), onPause() etc.).
  - Only one activity is active at a time; navigating to a new activity suspends the current one.
  - You may have a design that calls for modular "sub-activities" that can be added to a single parent activity; investigate Fragments to learn more.

Layout Fundamentals

- Layout creation mainly involves organising Views into the appropriate ViewGroups.
  - Individual Android UI widgets are often implemented as a single View.
- A collection of these widgets (Views) can be stored within a container layout (a ViewGroup) to structure them.

Device Variance

- Visual resources must account for major variations in both screen size and screen density.
- DP (Density-independent Pixels) are used for layout measurement to account for the density disparity.
- Android can automatically select resources based on the running device metrics.
  - Example: Higher resolution bitmap resources on devices with greater DPI.
  - The closest matching resource is picked in the absence of an exact match.

Device Variance

- Resource folders have qualifiers in their names based on their intended device class.
  - Folder selection is dependent on the running device.
  - Example: */drawable-MDPI*, */drawable-HDPI*…
  - Note the *ic-launcher.png* icon file in newly created Android project.
- You can design for a single device for simplicity.
  - Since Android picks the resource with the closest match, it's possible to supply only a single set of resources that will be used on *all* devices.

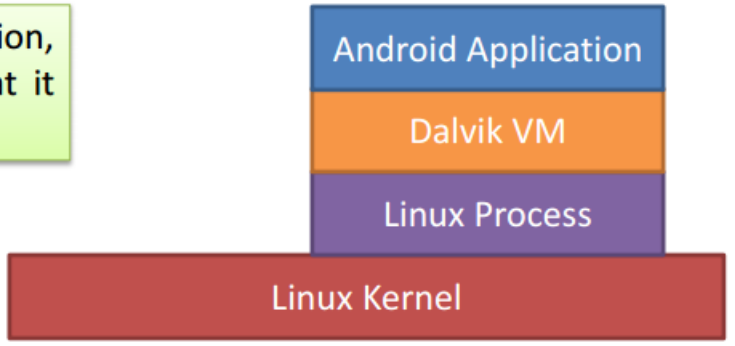
AndroidManifest.xml

- Exposes some key characteristics about your app to the system.
  - Minimum API version required for your app to function.
  - Declaration of permissions required by your app such as internet access, ability to write files to storage etc.
  - Declaration of activities in your app including which one will be the initial activity of your app.
  - App name, icon to display, versioning info...

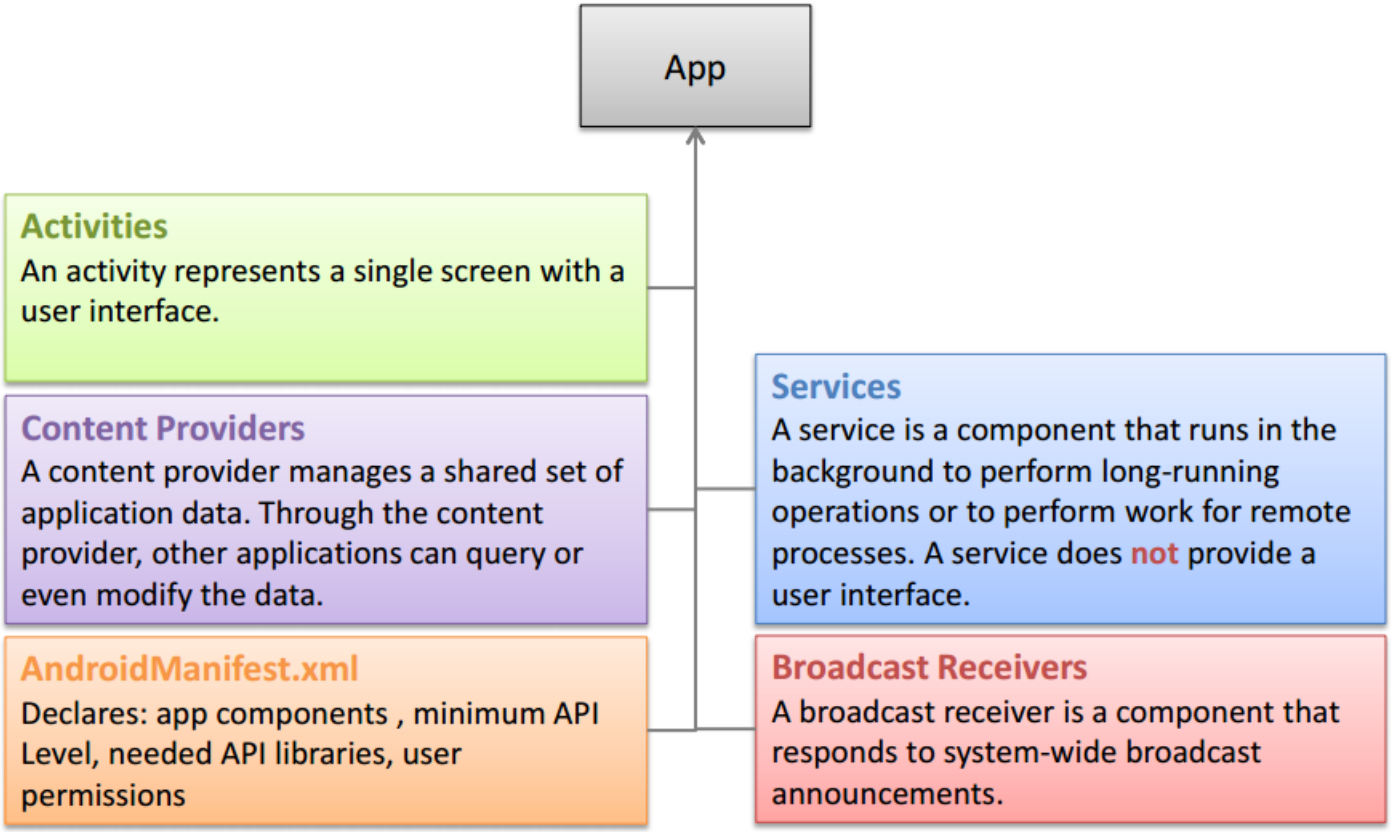
Application Fundamentals

- Development Language: Java
- Android SDK tools compile the code into an Android package, an archive file with an .apk suffix
- Security sandbox
  - Each application has a unique Linux user ID
  - Each process has its own virtual machine (VM)
  - Every application runs in its own Linux process

Principle of least privilege: Each application, has access only to the components that it requires to do its work and no more.



Application components



Some Pitfalls to Avoid!

Android Design vs. iOS Design

- If you're mostly familiar with iOS UI design, don't assume that
- Android apps should be designed the same way.
  - Forcing an iOS-style UI into an Android app is very painful and you end up fighting the framework to do so; it's not worth it.
  - Google provides detailed design guidelines to use as a reference.

API revisions

- Ensure the API version you're targeting contains the functionality your app depends on.
  - Check the docs for when a given class or method was introduced.
- The Android API has changed substantially in a relatively short time.
  - Large amount of *seemingly* useful functionality has been deprecated.
  - Refer to the documentation and up-to-date discussions when considering using an unfamiliar class.

Stuck in the Layout Editor?

- If so, stop and consider if an alternate ViewGroup and View hierarchy may work out.
- Each ViewGroup has major differences in how its child Views are structured.
  - Can be initially daunting at first once you start nesting them.
  - Experimenting with a “clean slate” in the Layout Editor can be extremely helpful when starting out.

Useful Apps & Libraries

Open Source

- Retrofit is a simple HTTP REST client with minimal development overhead.
- – Other libraries from square.github.io are also worth checking out (butterknife, picasso…).
- jfeinstein10/SlidingMenu on GitHub provides an easy to use implementation of the typical sliding menu UI component.

An Alternative Emulator

- The bundled Android Emulator can be quite slow even when configured for speed.
- Genymotion is substantially faster and free for personal use.
- Also includes convenient interfaces for updating the sensor state of the emulated device sensors (GPS, battery life…).
- Integrates into developer workflow just as well as the default emulator.

Ερωτήσεις;
Google Search

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Ταμείο Περιφερειακής Ανάπτυξης
ψηφιακήελλάδα
Όλα είναι δυνατά
Επιχειρησιακό Πρόγραμμα "Ψηφιακή Σύγκλιση"
grnet
ΕΣΠΑ 2007-2013
Ποιότητα ζωής για όλους
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
Επιστημονικό & Τεχνολογικό Πάρκο Ηπείρου
MedLab