

Android Notes

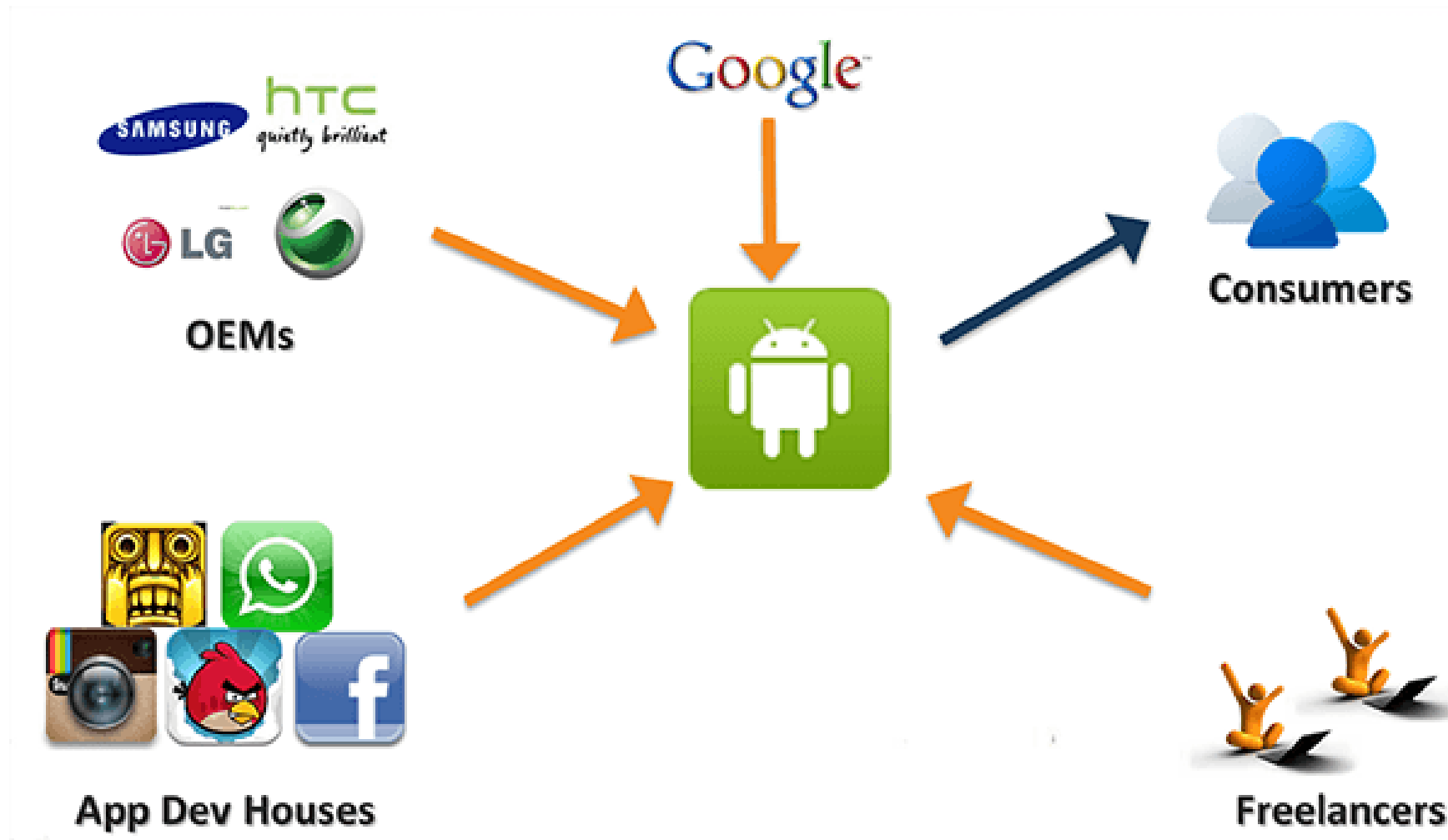
Unit I

Introduction to Android

What is Android?

- **Android**
- Android is a Linux-based operating system designed primarily for touchscreen mobile devices; such as smartphones and tablet computers.
- The first Android-powered phone was sold in October 2008.

Android Ecosystem



History of Android

- **Android Inc.**
- Android, Inc. was founded in Palo Alto, California in October 2003 by Andy Rubin, Rich Miner, Nick Sears and Chris White.



- The first android phone was HTC G1. Andy Rubin and his team were very excited about the first prototype of phone by Motorola and when it was brought in front of them this was their reaction: “It looked like a weapon. It was so sharp and jagged and full of hard lines. It looked like you could cut yourself on the edges”. Clearly, they were expecting something else.
- **Enter Google!**
- Google acquired Android Inc. on August 17, 2005, making it a wholly owned subsidiary. As the development advanced, Google marketed the platform to handset makers, promising them a smarter, more flexible and customizable system.

Android Applications

- Android applications are usually developed in the Java language using the Android Software Development Kit.
- Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play, SlideME, Opera Mobile Store, Mobango, F-droid** and the **Amazon Appstore**.

Categories of Android applications

There are many android applications in the market. The top categories are –



Music



News



Multimedia



Sports



Lifestyle



Food & Drink



Travel



Weather



Books



Business



Reference



Navigation



Social Media



Utilities



Finance

What is Android versions?

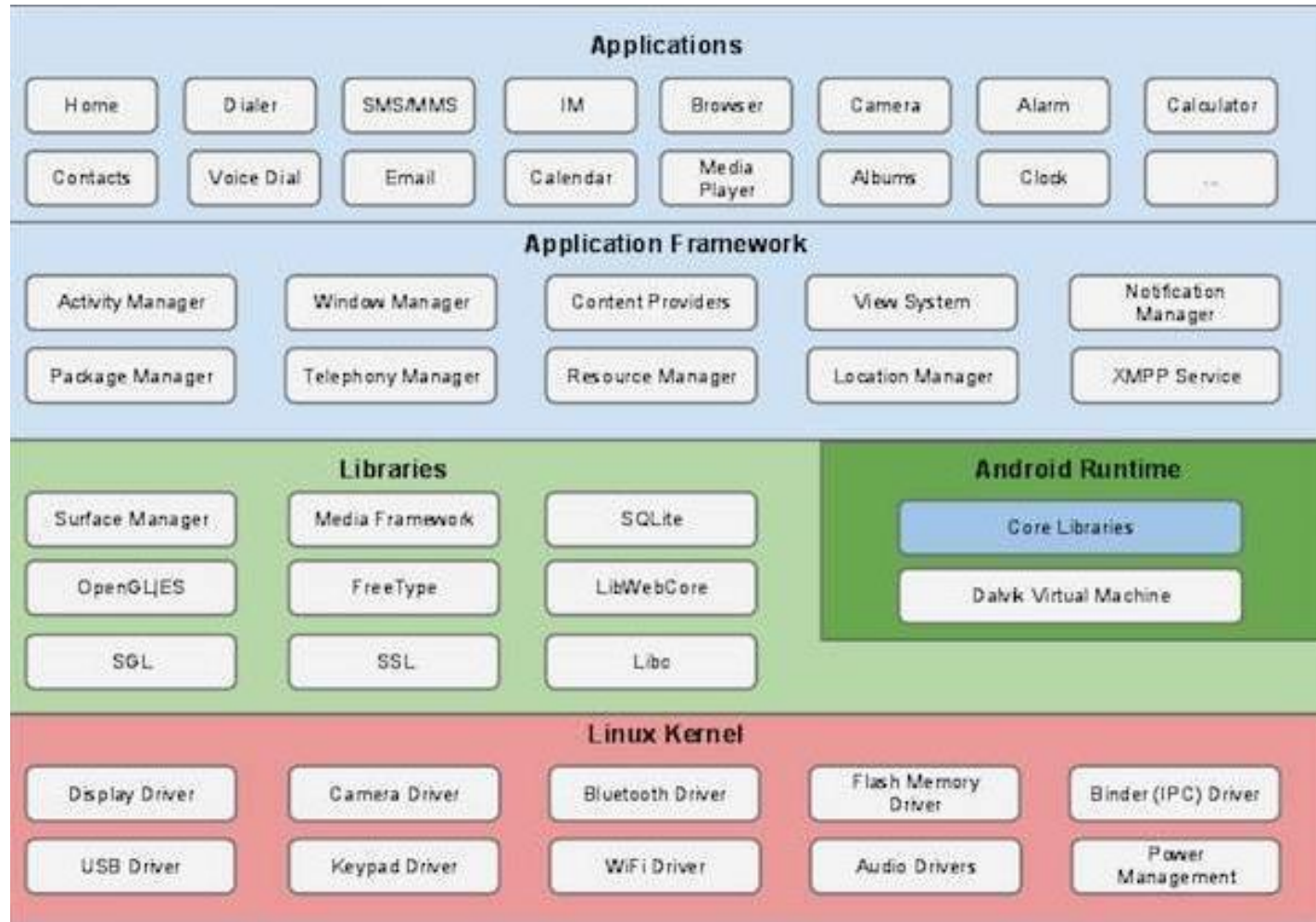
- Android is released in series of versions. Starting from 1.0 version (where 2.0, 3.0, Are latest releases).
- Google name these versions with some food items like ice cream, jelly bean, sandwich etc. which is one of the specialty of android versions.

Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010

Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016

Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

Android Architecture



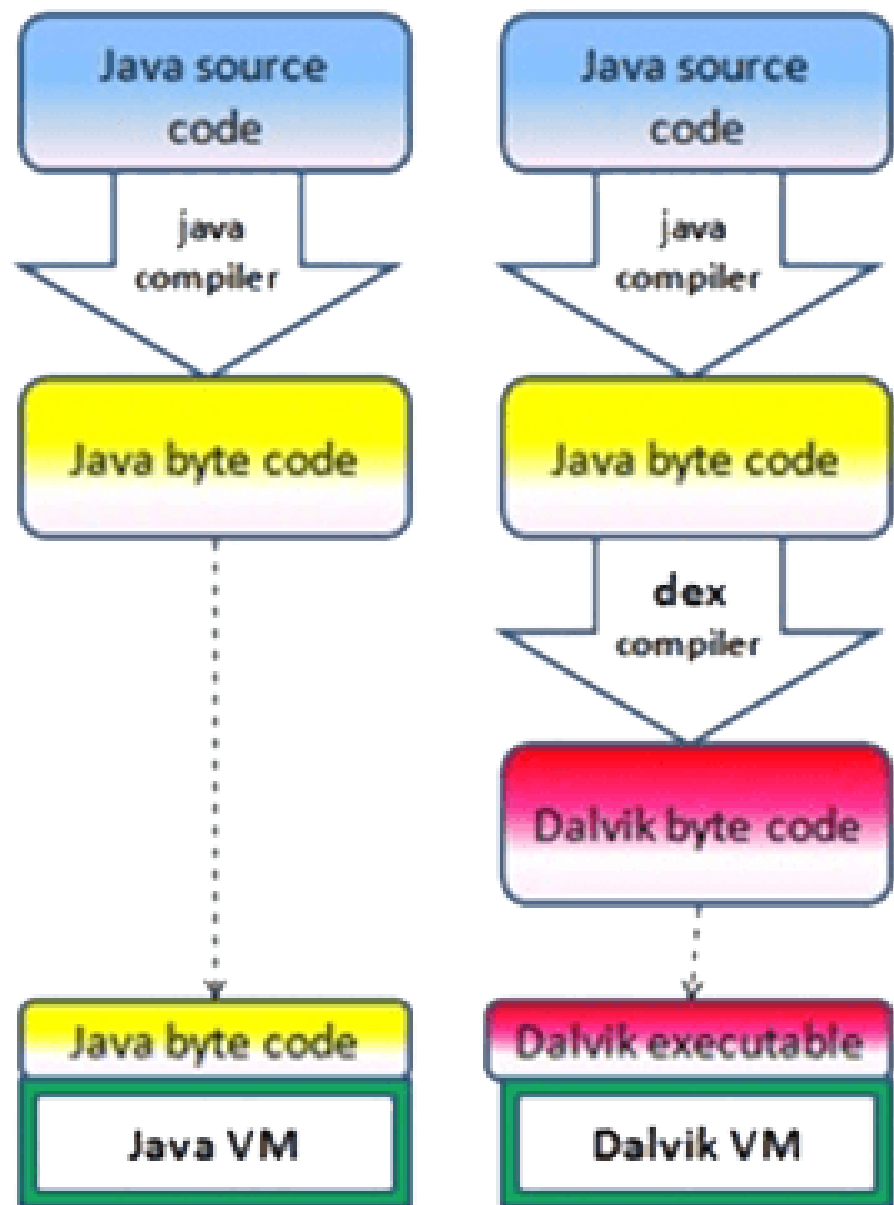
- Linux kernel
- At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.
- Libraries
- On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

- Android Libraries
- This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows –
- **android.app** – Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** – Facilitates content access, publishing and messaging between applications and application components.
- **android.database** – Used to access data published by content providers and includes SQLite database management classes.

- **android.opengl** – A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** – Used to render and manipulate text on a device display.
- **android.view** – The fundamental building blocks of application user interfaces.
- **android.widget** – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

What is Android Run-time?

Android Runtime consists of Dalvik Virtual machine and Core Java libraries.



- Application Framework
- The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.
- The Android framework includes the following key services –
- **Activity Manager** – Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** – Allows applications to publish and share data with other applications.
- **Resource Manager** – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** – Allows applications to display alerts and notifications to the user.
- **View System** – An extensible set of views used to create application user interfaces.

- Applications
- You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

Android Building blocks



Application Building Blocks



Activity

- UI Component Typically Corresponding to one screen.

IntentReceiver

- Responds to notifications or status changes. Can wake up your process.

Service

- Faceless task that runs in the background.

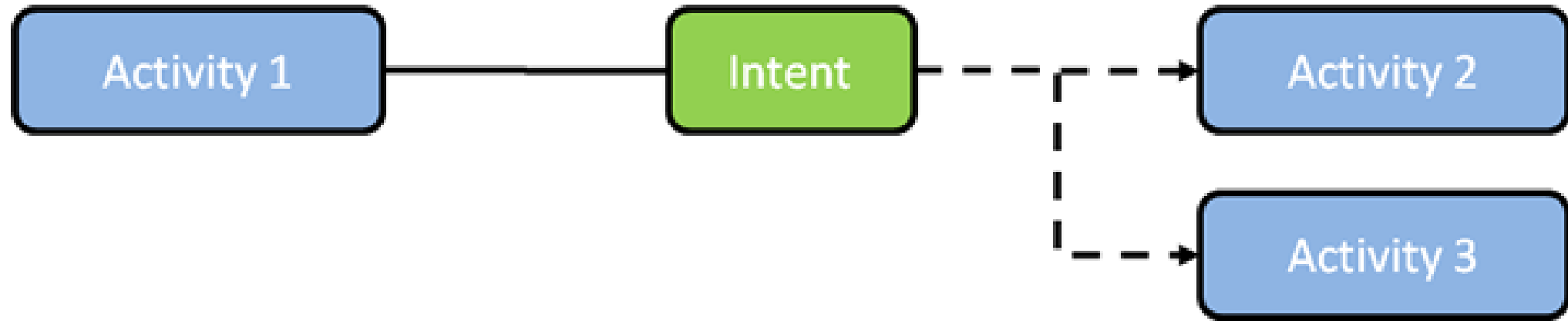
ContentProvider

- Enable applications to share data.

What is Android Activity?



What is Android Intent?



Services



What is Android Content Provider?



Wall Activity

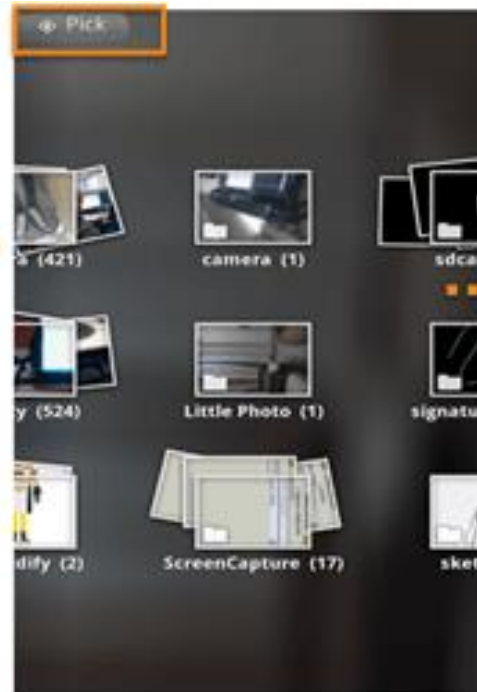


Photo Gallery
Content Provider

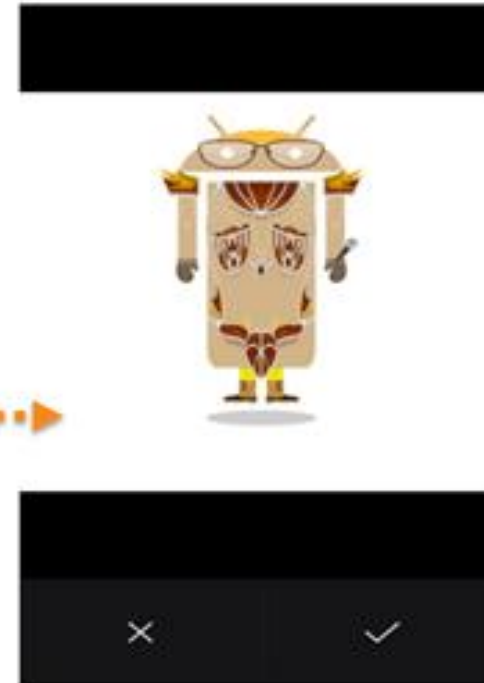
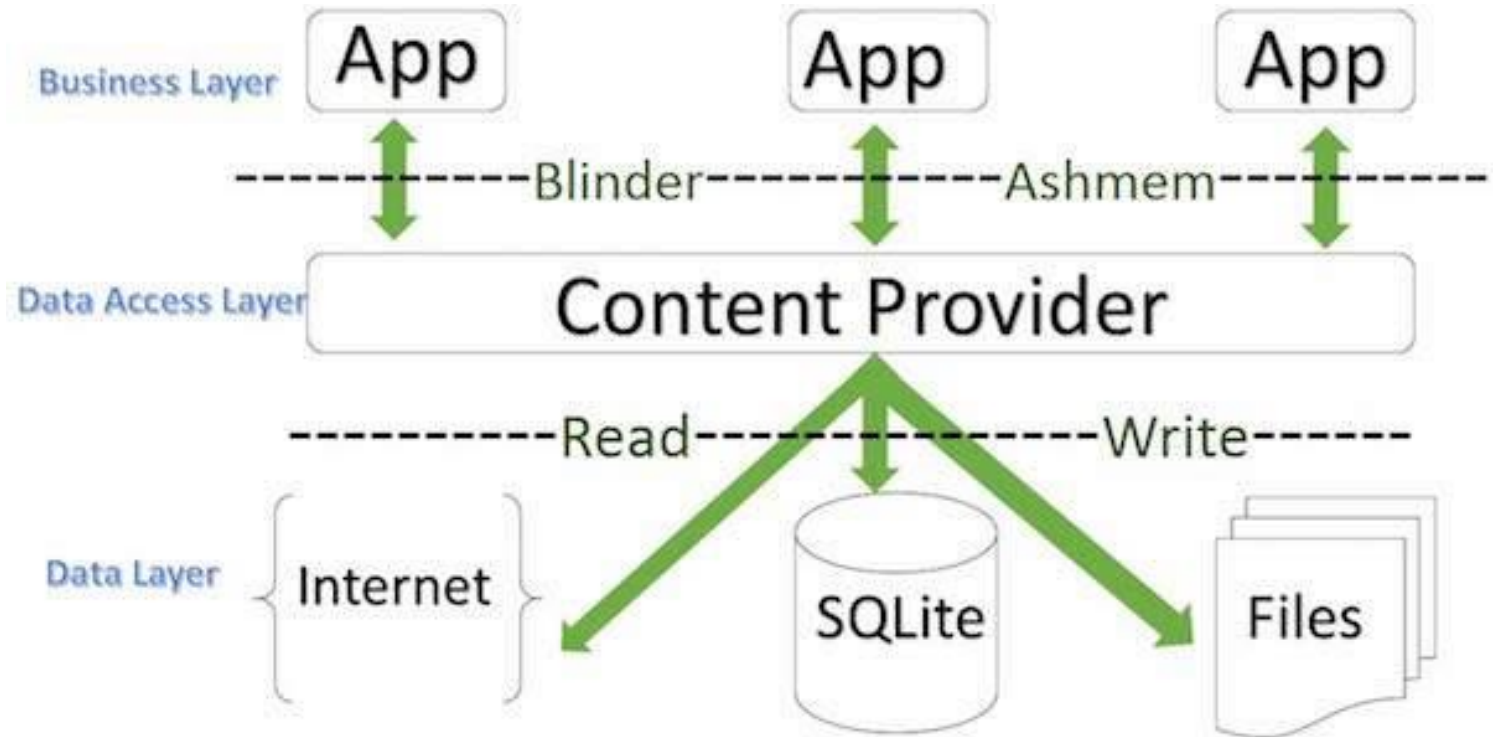
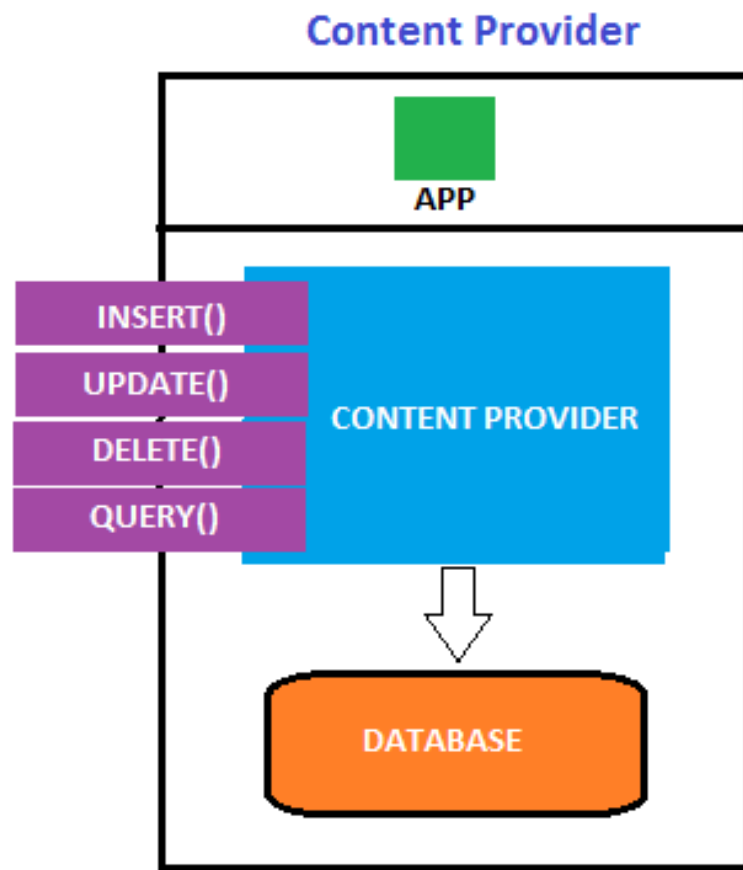
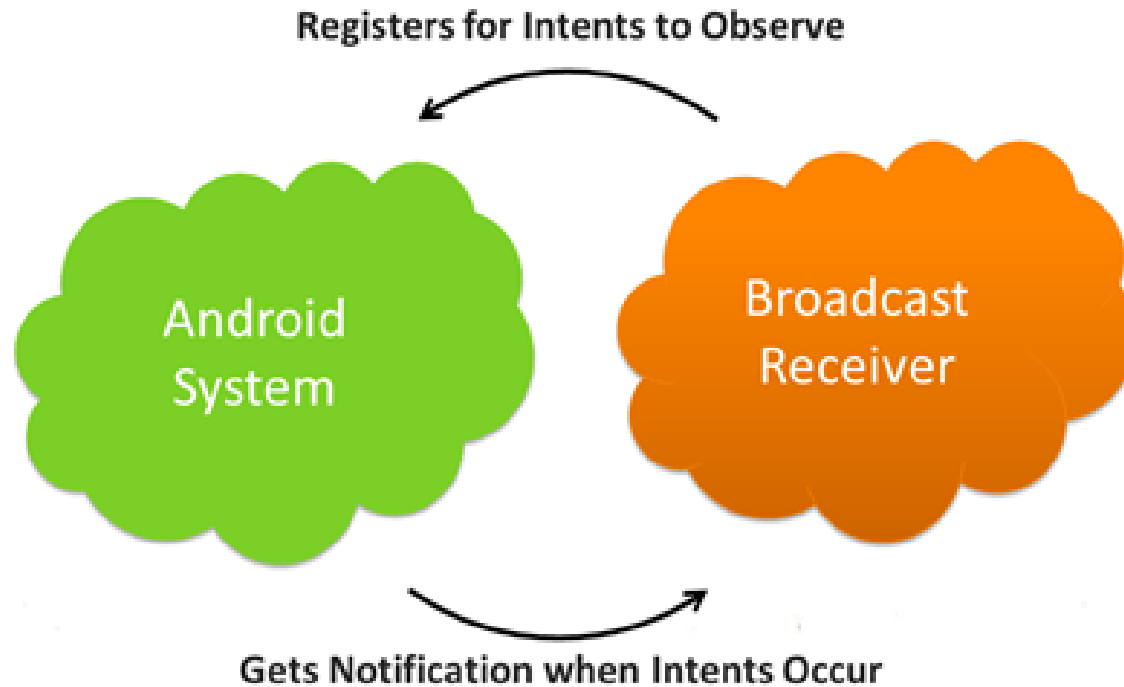


Photo - Content



What are Broadcast Receivers?



INTRODUCING THE DEVELOPMENT FRAMEWORK

- Android applications normally are written using Java as the programming language but executed by means of a custom VM called *Dalvik*, rather than a traditional Java VM.
- Each Android application runs in a separate process within its own Dalvik instance, relinquishing all responsibility for memory and process management to the Android run time, which stops and kills processes as necessary to manage resources.
- Dalvik and the Android run time sit on top of a Linux kernel that handles low-level hardware interaction, including drivers and memory management, while a set of APIs provides access to all the underlying services, features, and hardware.

What Comes in the Box

- The Android SDK includes everything you need to start developing, testing, and debugging Android applications:
- **The Android APIs** — The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries that Google uses to create native Android applications.
- **Development tools** — The SDK includes several development tools that let you compile and debug your applications so that you can turn Android source code into executable applications.

- **The Android Virtual Device Manager and emulator** — The Android emulator is a fully interactive mobile device emulator featuring several alternative skins. The emulator runs within an Android Virtual Device (AVD) that simulates a device hardware configuration. Using the emulator you can see how your applications will look and behave on a real Android device. All Android applications run within the Dalvik VM, so the software emulator is an excellent development environment — in fact, because it's hardware-neutral, it provides a better independent test environment than any single hardware implementation.
- **Full documentation** — The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them. In addition to the code documentation, Android's reference documentation and developer guide explains how to get started, gives detailed explanations of the fundamentals behind Android development, highlights best practices, and provides deep-dives into framework topics.

- **Sample code** — The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available with Android, as well as simple programs that highlight how to use individual API features.
- **Online support** — Android has rapidly generated a vibrant developer community. The Google Groups (<http://developer.android.com/resources/community-groups> .html#ApplicationDeveloperLists) are active forums of Android developers with regular input from the Android engineering and developer relations teams at Google. Stack Overflow (www.stackoverflow.com/questions/tagged/android) is also a hugely popular destination for Android questions and a great place to find answers to beginner questions.

Android Application Architecture

- Android's architecture encourages component reuse, enabling you to publish and share Activities, Services, and data with other applications, with access managed by the security restrictions you define.
- The same mechanism that enables you to produce a replacement contact manager or phone dialer can let you expose your application's components in order to let other developers build on them by creating new UI front ends or functionality extensions.

- The following application services are the architectural cornerstones of all Android applications, providing the framework you'll be using for your own software:
- **Activity Manager and Fragment Manager** — Control the lifecycle of your Activities and Fragments, respectively, including management of the Activity stack.
- **Views** — Used to construct the user interfaces for your Activities and Fragments,
- **Notification Manager** — Provides a consistent and nonintrusive mechanism for signaling your users.
- **Content Providers** — Lets your applications share data.
- **Resource Manager** — Enables non-code resources, such as strings and graphics, to be externalized.
- **Intents** — Provides a mechanism for transferring data between applications and their components.

Types of Android Applications

- ❑ **Foreground Activity** An application that's only useful when it's in the foreground and is effectively suspended when it's not visible. Games and map are common examples.
- ❑ **Background Service** An application with limited interaction that, apart from when being configured, spends most of its lifetime hidden. Examples of this include call screening applications or SMS auto-responders.
- ❑ **Intermittent Activity** Expects some interactivity but does most of its work in the background. Often these applications will be set up and then run silently, notifying users when appropriate. A common example would be a media player.

- ***Foreground Activities***
- When creating foreground applications, you need to consider the Activity life cycle carefully so that the Activity switches seamlessly between the foreground and the background.
- Applications have no control over their life cycles, and a backgrounded application, with no Services,
- is a prime candidate for cleanup by Android's resource management.
- This means that you need to save the state of the application when the Activity becomes invisible, and present the exact same state when it returns to the foreground.
- It's also particularly important for foreground Activities to present a slick and intuitive user experience.

- ***Background Services***

- These applications run silently in the background with very little user input.
- They often listen for messages or actions caused by the hardware, system, or other applications, rather than rely on user interaction.
- It's possible to create completely invisible services, but in practice, it's better form to provide at least some sort of user control.
- At a minimum, you should let users confirm that the service is running and let them configure, pause, or terminate it as needed.

- ***Intermittent Activities***
- Often you'll want to create an application that reacts to user input but is still useful when it's not the active foreground Activity.
- These applications are generally a union of a visible controller Activity with an invisible background Service.
- These applications need to be aware of their state when interacting with the user.
- This might mean updating the Activity UI when it's visible and sending notifications to keep the user updated when it's in the background,

Android Development Tools

- The Android SDK includes several tools and utilities to help you create, test, and debug your projects.
- the ADT plug-in conveniently incorporates most of these tools into the Eclipse IDE,
 - ☐ **The Android Virtual Device and SDK Managers** — Used to create and manage AVDs and to download SDK packages, respectively. The AVD hosts an Emulator running a particular build of Android, letting you specify the supported SDK version, screen resolution, amount of SD card storage available, and available hardware capabilities (such as touchscreens and GPS).
 - ☐ **The Android Emulator** An implementation of the Android virtual machine designed to run on your development computer. You can use the emulator to test and debug your android applications.
 - ☐ **Dalvik Debug Monitoring Service (DDMS)** Use the DDMS perspective to monitor and control the Dalvik virtual machines on which you're debugging your applications.
 - ☐ **Android Asset Packaging Tool (AAPT)** Constructs the distributable Android package files (.apk).

- ☐ **Android Debug Bridge (ADB)** The *ADB is a client-server application that provides a link to a running emulator*. It lets you copy files, install compiled application packages (.apk), and run shell commands.
- The following additional tools are also available:
 - ☐ **SQLite3** A database tool that you can use to access the SQLite database files created and used by Android
 - ☐ **Traceview** Graphical analysis tool for viewing the trace logs from your Android application
 - ☐ **MkSDCard** Creates an SDCard disk image that can be used by the emulator to simulate an external storage card.
 - ☐ **dx** Converts Java .class bytecode into Android .dex bytecode.
 - ☐ **activityCreator** Script that builds Ant build files that you can then use to compile your Android applications without the ADT plug-in.
 - ☐ **Hierarchy Viewer** — Provides both a visual representation of a layout's View hierarchy to debug and optimize your UI, and a magnified display to get your layouts pixel-perfect.
 - ☐ **Lint** — A tool that analyzes your application and its resources to suggest improvements and optimizations.



The Android Emulator

- The emulator is the perfect tool for testing and debugging your applications, particularly if you don't have a real device (or don't want to risk it) for experimentation.
- The Emulator is an implementation of the Dalvik VM, making it as valid a platform for running Android applications as any Android phone. Because it's decoupled from any particular hardware, it's an excellent baseline to use for testing your applications.

- ***Dalvik Debug Monitor Service (DDMS)***
- The emulator lets you see how your application will look, behave, and interact,
- but to really see what's happening under the surface, you need the DDMS.
- The Dalvik Debug Monitoring Service is a powerful debugging tool that lets you interrogate active processes, view the stack and heap, watch and pause active threads, and explore the filesystem of any active emulator.
- The DDMS perspective in Eclipse also provides simplified access to screen captures of the emulator and the logs generated by LogCat.

- ***The Android Debug Bridge (ADB)***
- *The Android debug bridge (ADB) is a client-service application that lets you connect with an Android Emulator or device.*
- It's made up of three components: a daemon running on the emulator, a service that runs on your development hardware, and client applications (like the DDMS) that communicate with the daemon through the service.
- As a communications conduit between your development hardware and the Android device/emulator, the ADB lets you install applications, push and pull files, and run shell commands on the target device.
- Using the device shell, you can change logging settings, and query or modify SQLite databases available on the device.

What Makes an Android Application?

- There are six components that provide the building blocks for your applications:
-  **Activities**
 - Your application's presentation layer.
 - Every screen in your application will be an extension of the Activityclass.
 - Activities use Views to form graphical user interfaces that display information and respond to user actions.
 - In terms of desktop development, an Activity is equivalent to a Form.
-  **Services**
 - The invisible workers of your application.
 - Service components run invisibly, updating your data sources and visible Activities and triggering Notifications.
 - They're used to perform regular processing that needs to continue even when your application's Activities aren't active or visible.

- **❑ Content Providers**

- A shareable data store.
- Content Providers are used to manage and share application databases.
- Content Providers are the preferred way of sharing data across application boundaries.
- This means that you can configure your own Content Providers to permit access from other applications and use Content Providers exposed by others to access their stored data.
- Android devices include several native Content Providers that expose useful databases like contact information.

- **❑ Intents**

- A simple message-passing framework.
- Using Intents, you can broadcast messages system-wide or to a target Activity or Service, stating your intention to have an action performed.
- The system will then determine the target(s) that will perform any actions as appropriate.

- **❑ Broadcast Receivers**

- Intent broadcast consumers.
- By creating and registering a Broadcast Receiver, your application can listen for broadcast Intents that match specific filter criteria.
- Broadcast Receivers will automatically start your application to respond to an incoming Intent, making them ideal for event-driven applications.

- **❑ Notifications**

- A user notification framework.
- Notifications let you signal users without stealing focus or interrupting their current Activities.
- They're the preferred technique for getting a user's attention from within a Service or Broadcast Receiver.
- For example, when a device receives a text message or an incoming call, it alerts you by flashing lights, making sounds, displaying icons, or showing dialog messages.
- You can trigger these same events from your own applications using Notifications.

Android SDK Features

- No licensing, distribution, or development fees
- Wi-Fi hardware access
- GSM, EDGE, and 3G networks for telephony or data transfer, allowing you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks
- Comprehensive APIs for location-based services such as GPS
- Full multimedia hardware control including playback and recording using the camera and microphone
- APIs for accelerometer and compass hardware
- IPC message passing

Android SDK Features (Continue...)

- Shared data stores
- An integrated open source WebKit-based browser
- Full support for applications that integrate Map controls as part of their user interface
- Peer-to-peer (P2P) support using Google Talk
- Mobile-optimized hardware-accelerated graphics including a path-based 2D graphics library and support for 3D graphics using OpenGL ES
- Media libraries for playing and recording a variety of audio/video or still image formats
- An application framework that encourages reuse of application components and the replacement of native applications

Android software development kit (SDK) includes

- **The Android APIs**
- **Development Tools**
- **The Android Emulator**
- **Full Documentation**
- **Sample Code**
- **Online Support**

Android Virtual Device Manager

- The Android Virtual Device Manager is used to create and manage the virtual devices that will host instances of the Emulator.
- AVDs are used to simulate the software builds and hardware configurations available on different physical devices. This lets you test your application on a variety of hardware platforms without needing to buy a variety of phones.



- The additional settings include the following:
- ‰ Maximum VM heap size
- ‰ Screen pixel density
- ‰ SD card support
- ‰ Existence of D-pad, touchscreen, keyboard,
 - and trackball hardware
- ‰ Accelerometer, GPS, and proximity sensor support
- ‰ Available device memory
- ‰ Camera hardware (and resolution)
- ‰ Support for audio recording
- ‰ Existence of hardware back and home keys

SDK manager

- The Android SDK Manager can be used to see which version of the SDK you have installed and to install new SDKs when they are released.
- Each platform release is displayed, along with the platform tools and a number of additional support packages. Each platform release includes the SDK platform, documentation, tools, and examples corresponding to that release.

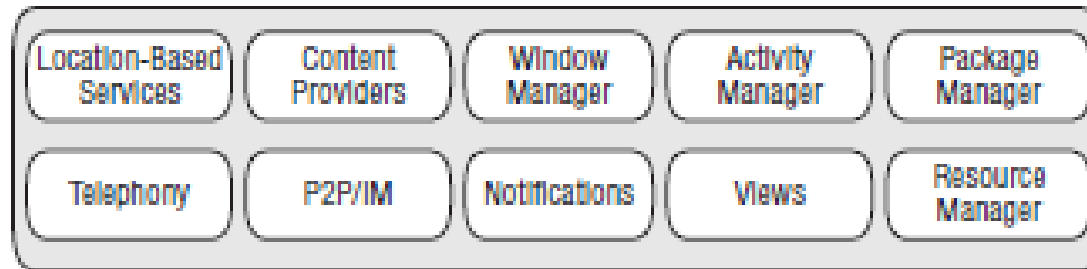
Installing and configuring Android SDK, ADT and AVD

Understanding the Android Software Stack

Application Layer



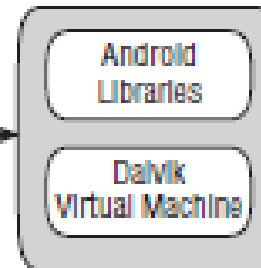
Application Framework



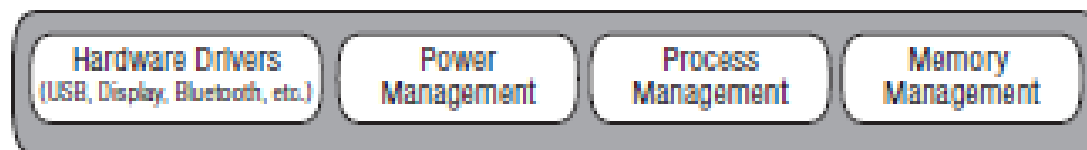
Libraries



Android Runtime



Linux Kernel



The Dalvik Virtual Machine

- The Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management.
- It's also possible to write C/C++ applications that run directly on the underlying Linux OS.
- The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory footprint.
- The .dex executables are created by transforming Java language compiled classes using the tools supplied within the SDK.

Thank you