

# Welcome





# **UNDERSTANDING THE ANDROID PLATFORM**

# Today's Overview

1

- What is Android

2

- Android System Architecture Overview

3

- Android Application Architecture Overview

4

- Function of Intent & IntentFilter

# What is Android?

- Android is a **software environment for mobile devices (not a hardware platform)** that includes an operating system, middleware and key applications
- Android is **not a particular device**, or even class of devices. It is a **platform that can be used and adapted to power different hardware configurations**

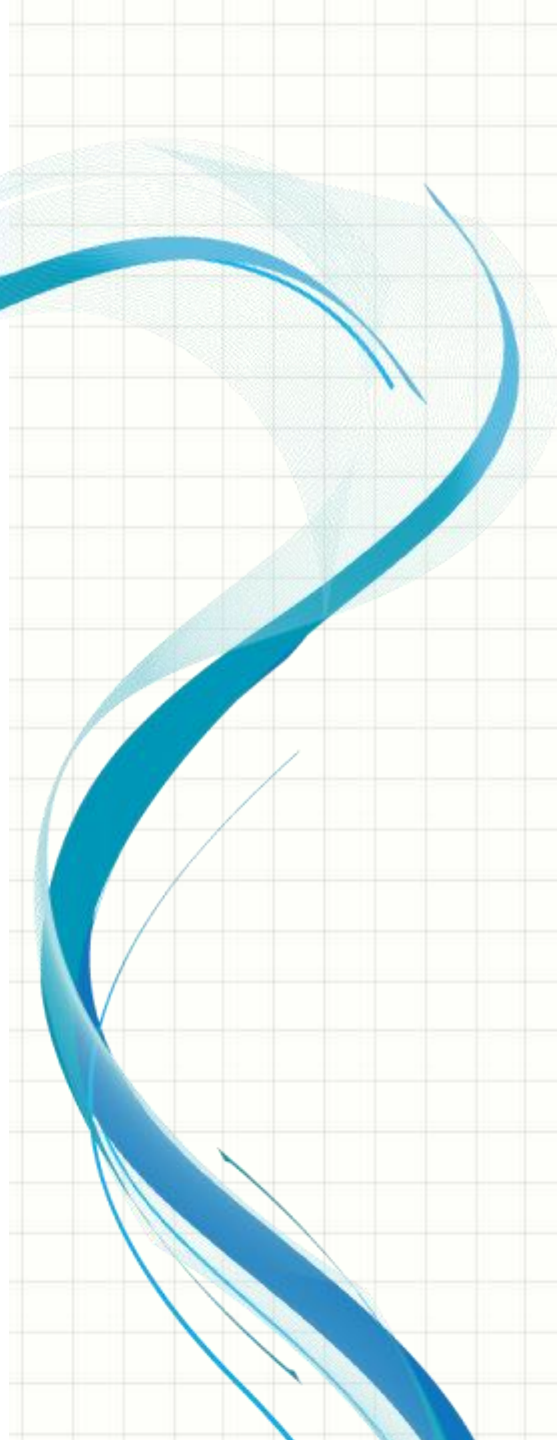
# Android Devices

- **Mobile phones** are the main class of Android powered devices, but it is currently used on **electronic book readers, netbooks, tablets, glass and set-top boxes**

android





A decorative graphic on the left side of the slide, consisting of a thick blue wavy line that curves upwards and then downwards. It is surrounded by lighter blue, semi-transparent wavy lines and small arrows pointing in the direction of the flow.

# System Architecture Overview

# Android System Architecture

- an Android system comprises **4 basic layers**
  - Linux Kernel
  - Libraries & Android Runtime
  - Application Framework
  - Applications



# Android System Architecture

## Application Layer

Native Apps  
(Contacts, Maps, Browser, etc)

Third-Party Apps

Developer Apps

## Application Framework

Location-Based  
Services

Content  
Providers

Window  
Manager

Activity  
Manager

Package  
Manager

Telephony

P2P/XMPP

Notifications

Views

Resource  
Manager

## Libraries

Graphics  
(OpenGL, SGL, FreeType)

Media

SSL &  
Webkit

libc

SQLite

Surface  
Manager

## Android Run Time

Android Libraries

Dalvik Virtual  
Machine

## Linux Kernel

Hardware Drivers  
(USB, Display, Bluetooth, etc)

Power  
Management

Process  
Management

Memory  
Management



# Android System Architecture

## Linux Kernel

- Android relies on **Linux for core system services** such as **security**, **memory management**, process management, network stack, and **driver model**. The **kernel** also acts as an abstraction layer between the hardware and the rest of the software stack

# Android System Architecture

## Application Layer

Native Apps  
(Contacts, Maps, Browser, etc)

Third-Party Apps

Developer Apps

## Application Framework

Location-Based  
Services

Content  
Providers

Window  
Manager

Activity  
Manager

Package  
Manager

Telephony

P2P/XMPP

Notifications

Views

Resource  
Manager

## Libraries

Graphics  
(OpenGL, SGL, FreeType)

Media

SSL &  
Webkit

libc

SQLite

Surface  
Manager

## Android Run Time

Android Libraries

Dalvik Virtual  
Machine

## Linux Kernel

Hardware Drivers  
(USB, Display, Bluetooth, etc)

Power  
Management

Process  
Management

Memory  
Management

# Android System Architecture

## Libraries

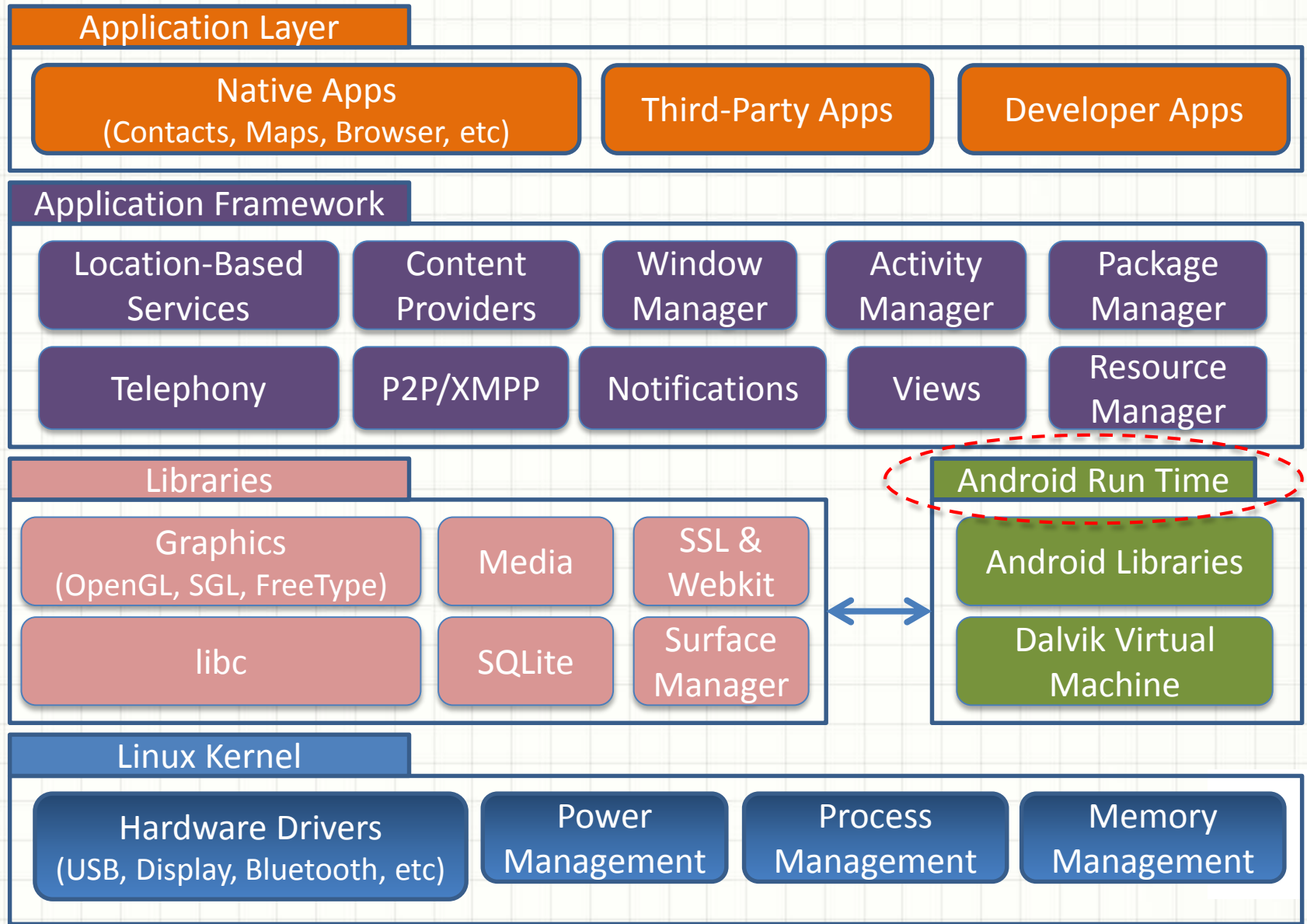
- Running on top of the kernel, Android includes
  - C/C++ core libraries  
such as libs and SSL
  - Media library  
for playback of audio and video media
  - Surface manager  
to provide display management

# Android System Architecture

## Libraries

- Running on top of the kernel, Android includes
  - Graphics libraries
    - include SGL and OpenGL for 2D and 3D graphics
  - SQLite
    - for native database support
  - SSL and WebKit
    - for integrated browser and Internet security

# Android System Architecture





# Android System Architecture

## Android Run Time

- The Android run time is the **engine** that powers your applications and, along with the libraries, forms the basis for the application framework

# Android System Architecture

## Android Run Time

Including

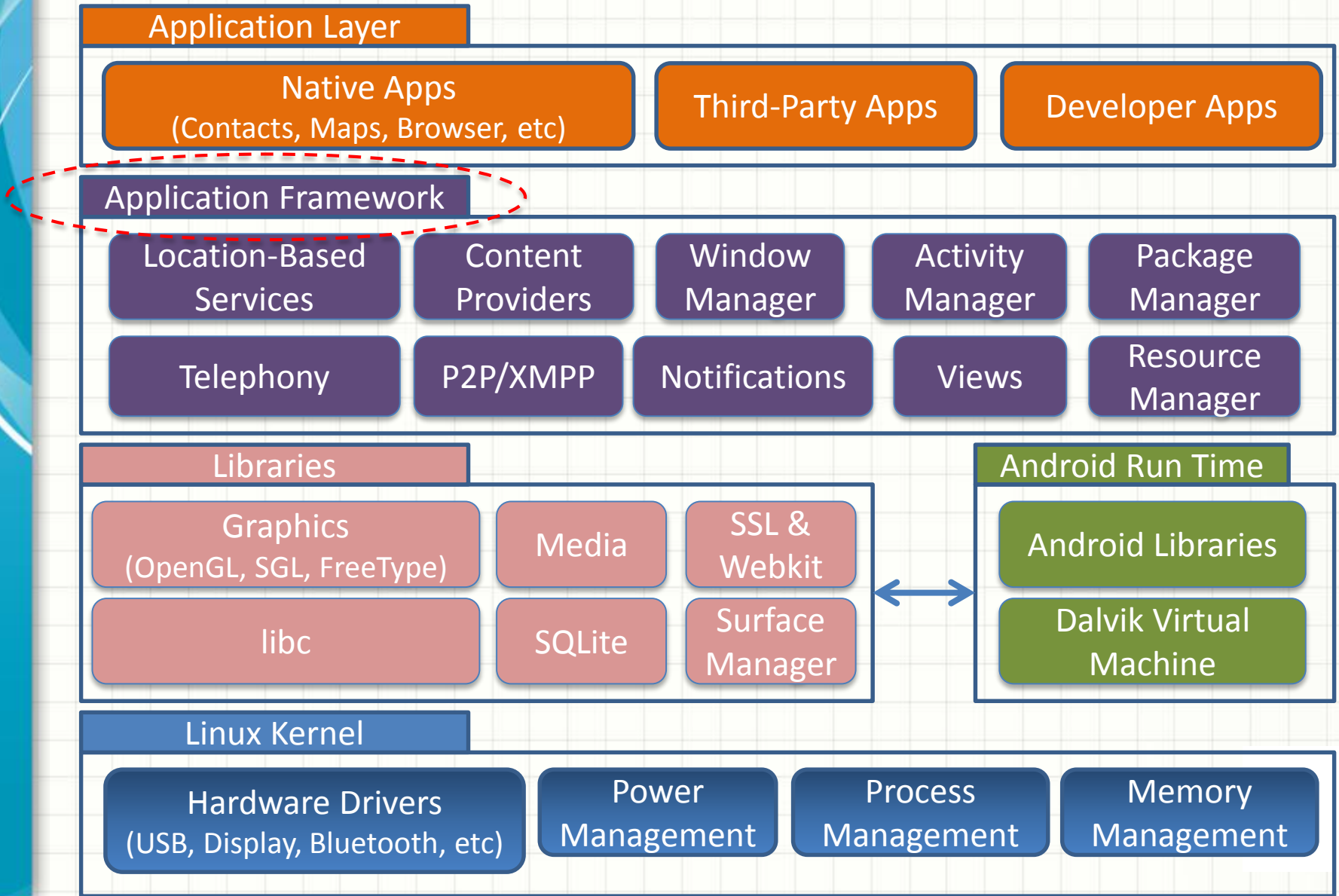
- **Core libraries**

The core Android libraries provide most of the functionality available in **core Java libraries** as well as the **Android-specific libraries**

- **Dalvik Virtual Machine**

Dalvik is a register-based virtual machine that's been optimized to ensure that a device can run multiple instances efficiently. **It relies on the Linux kernel for threading and low-level memory management**

# Android System Architecture



# Android System Architecture

## Application Framework

- A set of high-level building blocks for creating apps
- It also provides a generic abstraction for hardware access and manages the user interface and application resources

# Android System Architecture

## Application Framework

The **preinstalled framework on Android devices** consists of the following components:

- **Activity Manager**

**manages the lifecycle of applications** and maintains a shared activity stack for navigating within and among apps

- **Views**

**manages user interface elements (e.g. lists, text boxes, buttons, etc) and user interface-oriented event generation**



# Android System Architecture

## Application Framework

- **Notification Manager**  
enables all applications to display custom alerts in the status bar
- **Content Providers**  
enable applications to access data from other applications (such as Contacts), or to share their own data
- **Resource Manager**  
providing access to non-code resources such as localized strings, graphics, and layout files

# Android System Architecture

## Application Framework

- **Location Manager**  
enables an Android device to be aware of its physical location
- **Package Manager**  
lets an app learn about other app packages that are currently installed on the device
- **Telephony Manager**  
handles making and receiving phone calls

# Android System Architecture

## Application Framework

- **Window Manager**  
organizes the screen's real estate into windows, allocates drawing surfaces, and perform other window-related jobs

# Android System Architecture

## Application Layer

Native Apps  
(Contacts, Maps, Browser, etc)

Third-Party Apps

Developer Apps

## Application Framework

Location-Based  
Services

Content  
Providers

Window  
Manager

Activity  
Manager

Package  
Manager

Telephony

P2P/XMPP

Notifications

Views

Resource  
Manager

## Libraries

Graphics  
(OpenGL, SGL, FreeType)

Media

SSL &  
Webkit

libc

SQLite

Surface  
Manager

## Android Run Time

Android Libraries

Dalvik Virtual  
Machine

## Linux Kernel

Hardware Drivers  
(USB, Display, Bluetooth, etc)

Power  
Management

Process  
Management

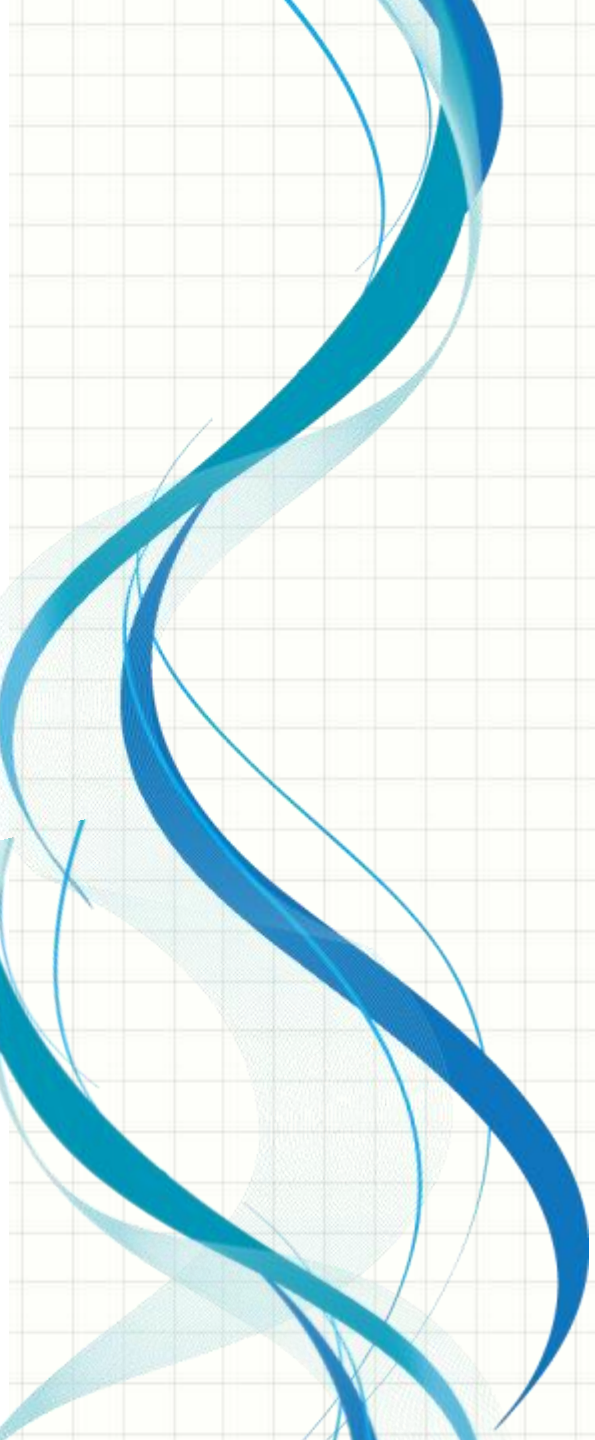
Memory  
Management

# Android System Architecture

## Application Layer

- Highest Layer in the Architecture
- All applications are written in Java programming language
- The application layer runs within the Android run time, using the classes and services made available from the application framework





# Application Architecture Overview

# Android Versions

- Android has gone through a number of updates since its first release in 2005. Since April 2009, each Android version has been developed under a codename based on a dessert item

ANDROID VERSION	CODENAME	API level
Android 1.5	Cupcake	3
Android 1.6	Donut	4
Android 2.0/2.1	Éclair	5 - 7
Android 2.2	Froyo	8

# Android Versions

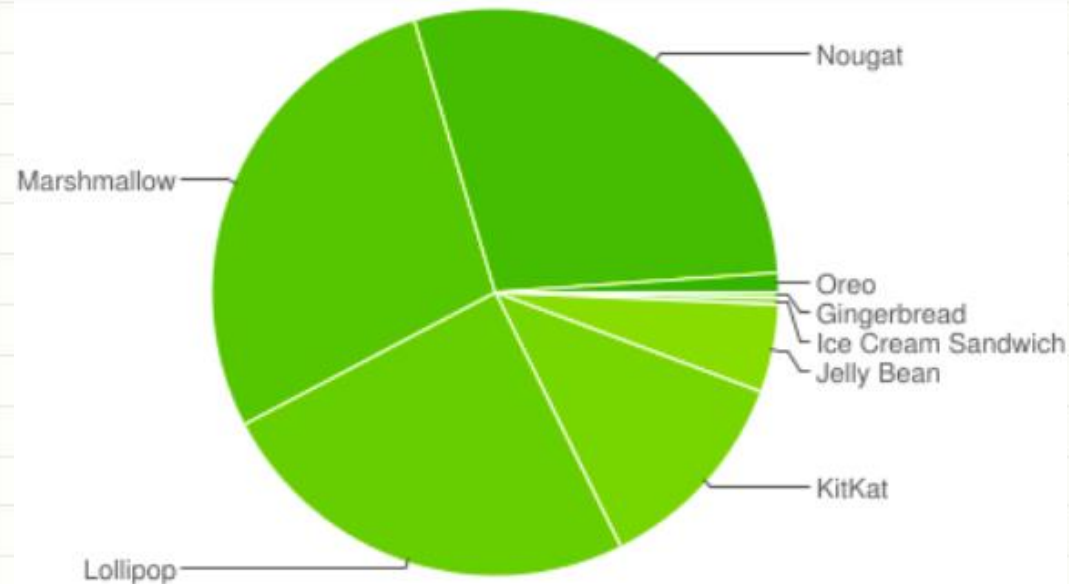
- Android has gone through a number of updates since its first release on 2005. Since April 2009, each Android version has been developed under a codename based on a dessert item

ANDROID VERSION	CODENAME	API level
Android 2.3	Gingerbread	9 - 10
Android 3.0	Honeycomb	11
Android 4.0/4.0.3	Ice cream Sandwich	14 - 15
Android 4.1/4.2/4.3	Jelly Bean	16 - 18
Android 4.4.2/4.4W	KitKat	19 - 20
Android 5.0/5.1	Lollipop	21 - 22
Android 6.0	Marshmallow	23
Android 7.0/7.1	Nougat	24 - 25
Android 8.0/8.1	Oreo	26 - 27

# Platform Version Distribution

- The relative number of devices running a given version of the Android platform
- Data collected during a *7-day period ending on Feb 5, 2018*.

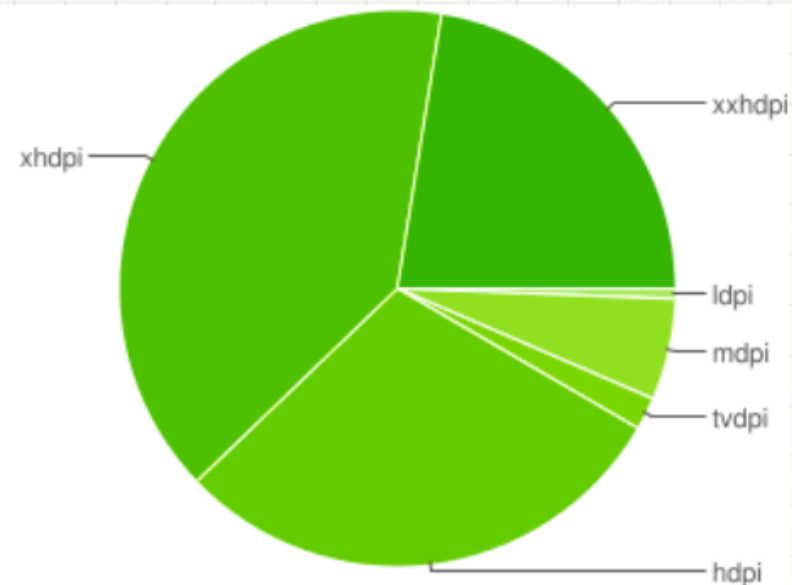
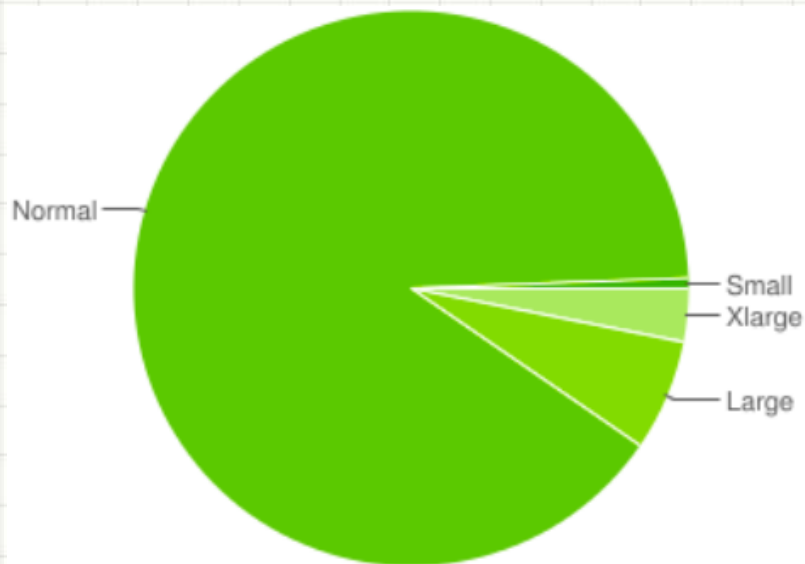
Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%



# Screen Sizes and Densities

- Data collected during a *7-day period ending on Feb 5, 2018.*

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	0.5%					0.1%	0.6%
Normal		1.1%	0.3%	28.5%	37.9%	22.0%	89.8%
Large	0.1%	2.8%	1.6%	0.4%	1.2%	0.4%	6.5%
Xlarge		2.0%		0.6%	0.5%		3.1%
Total	0.6%	5.9%	1.9%	29.5%	39.6%	22.5%	





# Android Application Architecture

- The architecture of an Android app **differs from desktop application** architecture
- App architecture is **based upon components that communicate with each other by using intents** that are described by a manifest and are stored in an app package
- **Application components are the essential building blocks of an Android application**

# Android Application Architecture

Important!

- Android apps **do not have a single entry point** (no C-style `main()` function, for example). Instead, **apps use components that are instantiated and run as needed**
- Each components exists as its own entity and plays a specific role—each one is a unique building block that helps define your application's overall behaviour.

# Android Application Architecture

- There are four different types of application components
- Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

# Android Application Architecture

Here are the four types of application components:

- Activities
- Services
- Content Providers
- Broadcast Receivers

# Android Application Components

## Activities

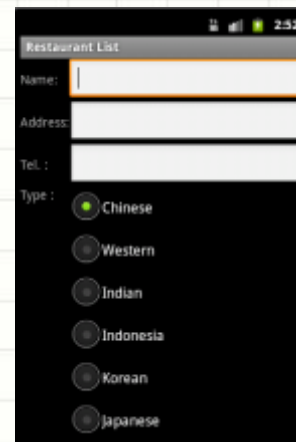
- An *activity* is implemented as a subclass of **Activity**
- An *activity* **represents a single screen with a user interface (UI)**

# Android Application Components

## Activities

- Example

a restaurant list application might have one activity that shows a list of restaurants, another activity to display the restaurant detail, and another activity for show map location





# Android Application Components

## Activities

- Each activities is independent of the others. Different application can start any one of these activities

- Example

A restaurant list app can start the map activity in the restaurant list app to show the location of the restaurant on Google map

# Android Application Architecture

Here are the **four types of application components**:

- Activities
- Services
- Content Providers
- Broadcast Receivers

# Android Application Components

Important!

## 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**  
No Graphical User Interface!

# Android Application Components

## Services

- Example

A service might play music in the background while the user is in a different application, or it might fetch data over the network **without blocking user interaction with an activity**

# Android Application Architecture

Here are the four types of application components:

- Activities
- Services
- Content Providers
- Broadcast Receivers

# Android Application Components

## Broadcast Receivers

- A *broadcast receiver* is implemented as a **subclass of BroadcastReceiver** and each broadcast is delivered as an Intent object
- A *broadcast receiver* is a component that **responds to system-wide broadcast announcements**



# Android Application Components

## Broadcast Receivers

- Many broadcasts **originate from the system**
- Example

A broadcast announcing that the screen has turned off, the battery is low, or a picture was captured

# Android Application Components

## Broadcast Receivers

- Apps can also **initiate broadcasts**
- Example

An app may want to let other apps know that some data has finished downloading from the network to the device and is available now for usage

# Android Application Architecture

Here are the four types of application components:

- Activities
- Services
- Content Providers
- Broadcast Receivers

# Android Application Components

## Content Providers Application Programming Interface

- A *content provider* is implemented as a subclass of **ContentProvider** and must implement a standard set of **APIs** that enable other applications to perform transactions
- A *content provider* manages a shared set of application data which is stored in the file system, an **SQLite database**, on the web, or any other persistent storage location the app can access

# Android Application Components

## Content Providers

- Through the content provider, other applications can query or even modify the data
- Example

The Android system provides a content provider that manages the user's contact information. As such, any application with the proper permissions can query part of the content provider (such as `ContactsContract.Data`) to read and write information about a particular person

# Intent & IntentFilter





# Activating Components

- Three (**activities, services, and broadcast receivers**) of the four component types **are activated by** an **asynchronous message** called an ***intent***
- **Intents** bind individual components to each other at runtime (you can think of them as the **messengers** that request an action from other components), whether the component belongs to your application or another

# Intents

- Intent is basically a message that is passed between components (such as Activities, Services, Broadcast Receivers, and Content Providers)
- It is almost equivalent to parameters passed to API calls

# Intents

- The fundamental differences between API calls and intents' way of invoking components are:

Advantage!

  - API calls are **synchronous** while **intent-based invocations are asynchronous**
  - API calls are **compile time binding** while **intent-based calls are run-time binding**

# Intents

- An **intent** is created with an **Intent object**, which defines a message to activate either a specific component or a specific *type* of component
- Example

An Activity can send an Intents to the Android system which starts another Activity

# Intents

- There are separate mechanisms for delivering intents to each type of component (such as Activities, Services or Broadcast Receivers) :
  - An Intent object is passed to Context.startActivity() or Activity.startActivityForResult() to launch an activity

Example

```
Intent intend = new Intent(.....);  
startActivity(intend);
```

# Intents

- There are separate mechanisms for delivering intents to each type of component:
  - An Intent object is passed to Context.startService() to initiate a service or deliver new instructions to an ongoing service. Similarly, an intent can be passed to Context.bindService() to establish a connection between the calling component and a target service



# Intents

- There are separate mechanisms for delivering intents to each type of component:
  - **Intent objects** passed **to** any of the **broadcast** methods (such as `Context.sendBroadcast()`, `Context.sendOrderedBroadcast()`, or `Context.sendStickyBroadcast()`) are delivered **to all interested broadcast receivers**. Many kinds of broadcasts originate in system code

Open this link in Chrome, Firefox, Safari ?

# Intents

Intents can be classified as:

- Explicit Intents
- Implicit Intents

# Explicit Intents

- Explicit Intents explicitly names the component which should be called by the Android system, by using the Java class as identifier
- It is made to work exactly like API calls

# Explicit Intents

- The following shows an explicit Intent to start the associated class
- Example

```
Intent intend = new Intent(this, HelloWorld.class);
startActivity(intend);
```
- When run, this code snippet has the following consequences:
  - A new instance of HelloWorld is created
  - The instance is pushed on top of the current task's stack, which is the one the calling activity is in.
  - The activity is started and brought to the foreground.

# Explicit Intents

- Example

you want explicitly call the activity B from activity A and **pass to it an array of integers**:

```
int intArray[] = {1,2,3,4};  
Intent in = new Intent(this, B.class);  
in.putExtra("my_array", intArray);  
startActivity(in);
```

To read the information in activity B (in onCreate() method) you should use the following code:

```
Bundle extras = getIntent().getExtras();  
int[] arrayInB = extras.getIntArray("my_array");
```

# Intents

Intents can be classified as:

- Explicit Intents
- Implicit Intents



# Implicit Intents

- Implicit Intents do not specify the Java class which should be called
- The Android system finds the best component for handling the intent. This is done by comparing the contents of Intent object with Intent Filters
- The comparison to Intent Filter is done with three elements of intent object namely action, data and category

# Implicit Intents

- Example

the following tells the Android system to view a webpage. Typically the web browser is registered to this Intent but other component could also register themself to this event

```
Intent intent = new Intent(Intent.ACTION_VIEW,  
Uri.parse("http://www.sp.edu.sg"));
```

# Intent Filters

- If an Intent is sent to the Android system, it will determine suitable applications for this Intent Example: Settings application
- If several components have been registered for this type of Intent, Android offers the users choice to open one of them Web browser
- This determination is based on IntentFilters. An IntentFilter specifies the types of Intent that an activity, service, or broadcast receiver can respond

# Intent Filters

- Intent Filters describe the capability of the component (Activity, Service or Broadcast Receivers) to handle an implicit intent
- It specifies what an activity or service can do and what types of broadcasts a receiver can handle
- Components without intent filters cannot receive implicit intents, but can only handle explicit intents

# Intent Filters

- IntentFilters are typically defined via the "AndroidManifest.xml" file.

For BroadcastReceiver it is also possible to define them in coding

- An IntentFilters is defined by its category, action and data filters. It can also contain additional metadata

# Intent Filters

- Example

The following will register an Activity for the Intent which is triggered when someone wants to open a webpage

```
<activity android:name=".BrowserActivitiy" android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http"/>
  </intent-filter>
</activity>
```



# Intent Filters

- Example

The following shows how you could define an Intent receiver for the ACTION.SEND Intent

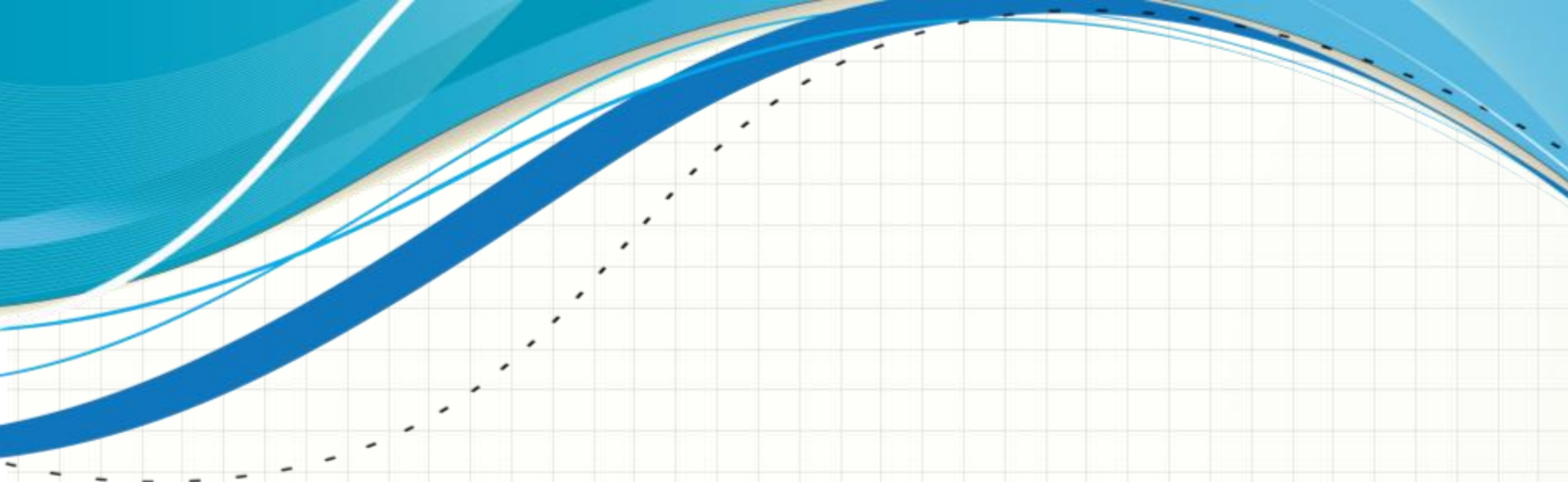
```
<activity android:name=".ActivityTest" android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
</activity>
```

# Intent Filters

- Example

The following will register an Activity for the ACTION.SEND Intent for the “text/plain” mime type

```
<activity android:name=".ActivityTest" android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
</activity>
```



**END**