

**Android - Architecture**

## **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.

* 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.

## **Android Runtime**

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

## **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, colour 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 Core Building Blocks**



An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

#### Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.



#### **View**

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

#### **Intent**

Intent is used to invoke components. It is mainly used to:

* Start the service
* Launch an activity
* Display a web page
* Display a list of contacts
* Broadcast a message
* Dial a phone call etc.

For example, you may write the following code to view the webpage.

*Intent intent=new Intent(Intent.ACTION\_VIEW);*

*intent.setData(Uri.parse("http://www.javatpoint.com"));*

*startActivity*(intent);



#### **Service**

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.



#### **Content Provider**

Content Providers are used to share data between the applications.



#### **Fragment**

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

#### **AndroidManifest.xml**

It contains information about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

#### **Android Virtual Device (AVD)**

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.



## **Techopedia Explains Dalvik Debug Monitor Service (DDMS)**

The Dalvik Debug Monitor Service allows developers to spot bugs in applications running on either an emulator or an actual Android device.   
  
For example, by using the DDMS’ LogCat feature, developers can view log messages regarding the state of the application and the device. LogCat can pinpoint the exact line number on which an error occurred.   
  
Another DDMS feature, known as the Emulator Control, allows developers to simulate phone states and activities. For example, it can simulate different types of networks such as GPRS, EDGE, and UTMS, which can have different network characteristics such as speed and latency.  
  
Aside from the features mentioned earlier, the Dalvik Debug Monitor Service also provides LogCat, process, and radio state information as well as incoming call, SMS, and location data spoofing.   
  
This debugging tool can be integrated into the Eclipse IDE by adding the ADT (Android Development Tools) plug-in. Otherwise, it can be accessed from the command line and will automatically connect to any running emulator.

**What is service? Explain the different types of services.**

* **Services run invisibly** —doing Internet lookups, processing data, updating your Content Providers, firing Intents, and triggering Notifications.
* **Services are designed to be longer-lived** —specifically, to perform ongoing and potentially time-consuming operations.
* Services are started, stopped, and controlled from other application components, including Activities, Broadcast Receivers, and other Services.
* Running Services have a higher priority than inactive or invisible (stopped) Activities, making them less likely to be terminated by the run time’s resource management.
  + 1. **Creating and Controlling Services**
    - The following sections describe how to create a new Service, and how to start and stop it using Intents with the **startService** and **stopService** methods, respectively.
    - Later you’ll learn how to bind a Service to an Activity to provide a richer interface.

Creating Services To define a Service, create a new class that extends Service. You will need to override the **onCreate** and **onBind** methods,

* + 1. 
    2. 

After you’ve constructed a new Service, you must register it in the application manifest.

*<service android:enabled=”true”*

*android:name=”.MyService”*

*android:permission=”com.paad.MY\_SERVICE\_PERMISSION”/>*