Mobile Application Development(MCA-II)

Unit 1. Mobile application development architectures

- 1.1. Introduction to Mobile Application technologies
- 1.2. Android Architecture
- 1.3. IOS Architecture
- 1.4. Windows Architecture
- 1.5. Hybrid Architecture

1.1. Introduction to Mobile Application technologies:

Talking about the mobile applications, the first thing that comes to mind are the apps like Whatsapp, Instagram, swiggy, etc that we use in our everyday life. Ever thought about how these apps are made? Which technology is used? Let's discuss what technologies or frameworks can be used to develop a mobile application. Mobile apps are majorly developed for 3 Operating System. :

1. Android
2. IOS
3. Windows

There are 3 different ways to develop Mobile apps: –

1. Native App development
2. Progressive web Application
3. Cross-Platform Application

- **Native App development: –**

These types of apps normally run in the native devices, that is, it runs only in the OS that it is specifically designed for it. These apps cannot be used on different

devices using a different OS. The apps that are developed for android are normally coded using Java or Kotlin languages. The IDE normally used for android app development is Android Studio which provides all features and the apps that are developed for IOS are generally coded in Swift language or Objective-C. The IDE suggested for IOS App Development is XCode.

Example:

Here's an example of a native app:

A retail company wants to improve the in-store shopping experience for its customers. They develop a native app that allows customers to:

- Browse the store's inventory and product information
- Create a shopping list
- Scan barcodes to view product information and reviews
- Locate items in the store using an interactive map
- Pay for items directly through the app, without having to wait in line at the register
- The app is only available to the company's customers and can only be used in their physical stores. The app is designed to integrate with the company's existing systems, such as inventory management and point-of-sale systems.

This app is developed by the retail company for their own use, to improve the in-store customer experience, increase sales and gain insights from the customer's behavior.

In this example, the retail company is the 1st party, and the app is a native app, because it is developed for the specific platform (iOS or Android) and can take full advantage of the device's capabilities and features.

Advantages of Native App development:

1. The performances of these apps are very high these apps very fast compared to any other apps.
2. We have easy access to all the features and APIs.
3. The community is widespread so all your doubts and errors can be discussed and solved easily.
4. Updates are available on the same day.

Disadvantages of Native App development:

1. The development speed is too slow as we have to code it again for different OS.
2. And this category doesn't support open source.

2. Progressive web Application: –

Progressive web apps are essentially a website which runs locally on your device. The technologies used are Microsoft Blazor, React, Angular JS, Native Script, Iconic. These technologies normally used for web development propose. The apps' UI is developed the same way as they are developed while developing the website. This category has many ups and downs let's start with the advantages of Progressive web apps.

Example:

Here's an example of a Progressive Web App:

A news website wants to provide its users with a better mobile experience. They develop a Progressive Web App that:

- Allows users to access the website offline by storing content on the user's device
- Sends push notifications to users to alert them of breaking news
- Can be installed on the user's home screen like a native app
- Provides a fast and smooth browsing experience
- Has a responsive design that adapts to different screen sizes
- Users can access the PWA by visiting the website on their mobile browser. They are prompted to install the PWA on their home screen, which allows them to access the website offline and receive push notifications.

In this example, the news website is the 1st party and the app is a Progressive web app, because it can be accessed through a web browser and can be installed on the user's device like a native app. It also allows users to access the content offline and have a fast and smooth experience.

Advantages of Progressive web Application:

1. The main advantage of this process is that its development speed is fast the same code base is used for IOS, Android, web applications.
2. The web development team can be repurposed to develop the mobile application.
3. No installation required.

Disadvantages of Progressive web Application:

1. The major disadvantage is that PWA don't have access to all the feature and so the user experience is not that good IOS does not support all the features of PWA
2. The UI for development is bespoke i.e. the buttons, edit texts need to be programmed which was not necessary for the 1st party native Apps.
3. The community is not that wide spread.
4. No extra room for business model i.e. it is still a challenge to develop a revenue model or advertising opportunities for PWAs. At the moment, there are fewer options than among native apps to subscribe to.

3. Cross-Platform Application: –

These are frameworks that allow developing total native applications which have access to all the native features of IOS and Android but with the same code base. These apps run on both Android and IOS. So normally the development speeds of these apps are very fast and the maintenance cost is low. The performance speed is comparatively low to native apps but faster than PWA.
Xamarin is Microsoft cross-platform solution that uses the programming languages like .NET, C#, F#. The IDE preferred is Visual Studio. The UI/UX is totally native giving access to all features. This technology is having a wide community. And whenever an update is released by Android and IOS the same updates are released by Microsoft through Visual Studio.

React Native is Facebook's cross-platform solution which uses the language JavaScript And the preferred IDE is WebStrome & Visual Studio Code. Same like Xamarin React Native has totally native UI/UX and gives access to all features. And the updates are released the same day by Facebook as Android and IOS.

Flutter is Google's cross-platform solution which uses the language, Dart. The IDE preferred is Android Studio, IntelliJ IDE, and Visual Studio Code. The UI/UX is bespoke and Flutters has to come up with their new libraries whenever Android and IOS comes up with an update to mimic those update. The community is fast growing.

Example:

Here's an example of a cross-platform application:

A project management company wants to create a project management tool that can be used by teams on different platforms. They develop a cross-platform application that:

- Can be used on Windows, Mac, iOS, and Android devices
- Allows users to create and assign tasks, set deadlines, and track progress
- Integrates with popular tools such as Google Calendar and Trello
- Has a user-friendly interface that works seamlessly across all platforms
- The application can be downloaded from the company's website or from different app stores such as App Store, Google Play Store, Microsoft Store, and Mac App Store, depending on the platform.

This example illustrates how the company developed a project management tool that can be used by teams on different platforms, Windows, Mac, iOS and Android, which is a cross-platform application. It allows teams to collaborate and manage their projects seamlessly, regardless of the platform they use.

Advantages of Cross-Platform Application:

1. The apps' development speed is very high as they use the same code base for both Android and IOS.
2. The apps' maintenance cost is low as the errors and updates as to be countered only once.

Disadvantages of Cross-Platform Application:

1. Slow Code Performance With Limited Tool Availability.
2. Limited User Experience i.e. these apps does not have access to Native only features.

1.2. Android Architecture

Android architecture is a software stack of components to support mobile device needs. Android software stack contains a Linux Kernel, a collection of c/c++ libraries which are exposed through an application framework services, runtime, and application.

Following are main components of android architecture those are:

1. **Applications:**

Applications is the top layer of android architecture. The pre-installed applications like home, contacts, camera, gallery etc and third party applications downloaded from the play store like chat applications, games etc. will be installed on this layer only. It runs within the Android run time with the help of the classes and services provided by the application framework.

2. Android Framework:

Application Framework provides several important classes which are used to create an Android application. It provides a generic abstraction for hardware access and also helps in managing the user interface with application resources. Generally, it provides the services with the help of which we can create a particular class and make that class helpful for the Applications creation.

It includes different types of services activity manager, notification manager, view system, package manager etc. which are helpful for the development of our application according to the prerequisite.

3. Android Runtime:

The Android Runtime environment is one of the most important parts of Android. It contains components like core libraries and the Dalvik virtual machine(DVM). Mainly, it provides the base for the application framework and powers our application with the help of the core libraries.

Like Java Virtual Machine (JVM), **Dalvik Virtual Machine (DVM)** is a register-based virtual machine and specially designed and optimized for android to ensure that a device can run multiple instances efficiently. It depends on the Linux kernel for threading and low-level memory management. The core libraries enable us to implement android applications using the standard JAVA or Kotlin programming languages.

4. Platform Libraries:

The Platform Libraries include various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide support for android development.

i) **Media** library provides support to play and record audio and video formats.
ii) **Surface manager** responsible for managing access to the display subsystem.
iii) **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
iv) **SQLite** provides database support and FreeType provides font support.
v) **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
vi)**SSL** (Secure Sockets Layer) is security technology to establish an encrypted link between a web server and a web browser.

5. Linux Kernel:

Linux Kernel is the heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime.

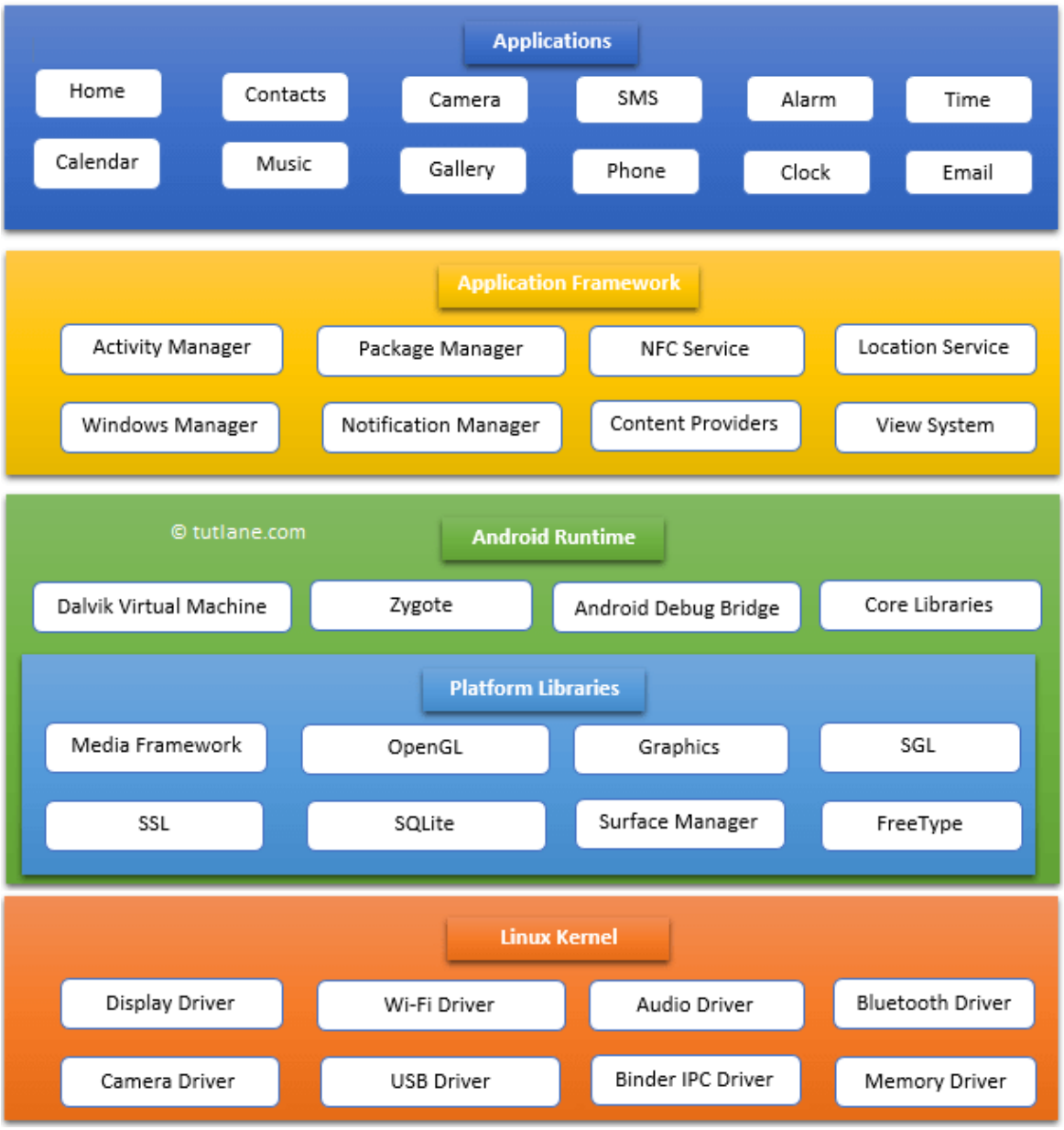
The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc.

In these components, the Linux Kernel is the main component in android to provide its operating system functions to mobile and Dalvik Virtual Machine (DVM) which is responsible for running a mobile application.

Following is the pictorial representation of android architecture with different components.

- **Security**: The Linux kernel handles the security between the application and the system.
- **Memory Management**: It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management**: It manages the process well, allocates resources to processes whenever they need them.

- **Network Stack**: It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.



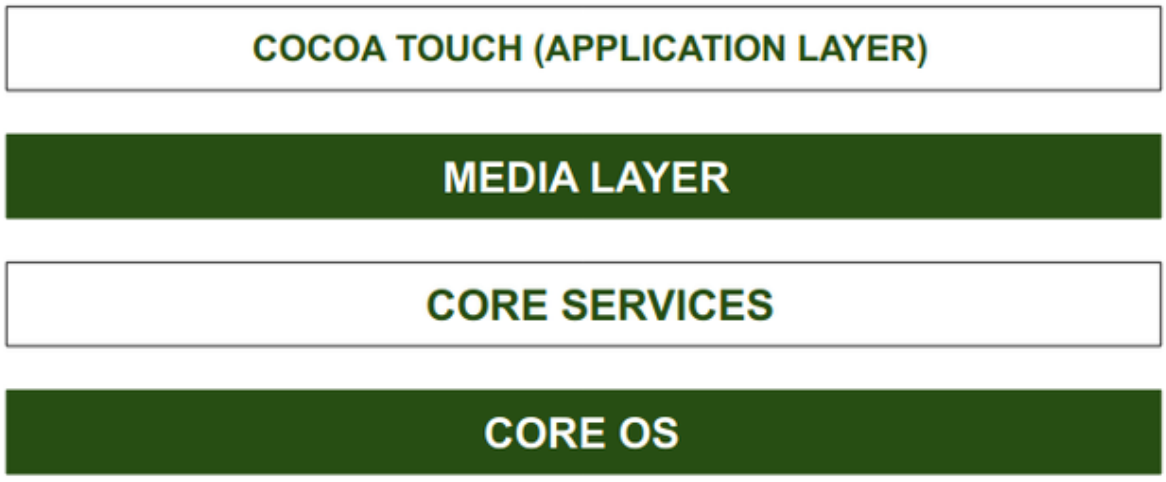
1.3. IOS Architecture

IOS is a Mobile Operating System that was developed by Apple Inc. for iPhones, iPads, and other Apple mobile devices. iOS is the second most popular and most used Mobile Operating System after Android.

The structure of the iOS operating System is Layered based. Its communication doesn't occur directly. The layer's between the Application Layer and the Hardware layer will help for Communication. The lower level gives basic services on which all applications rely and the higher-level layers provide graphics and interface-related services. Most of the system interfaces come with a special package called a framework.

A framework is a directory that holds dynamic shared libraries like .a files, header files, images, and helper apps that support the library. Each layer has a set of frameworks that are helpful for developers.

Architecture of IOS



CORE OS Layer:
All the IOS technologies are built under the lowest level layer i.e. Core OS layer. These technologies include:

1. Core Bluetooth Framework
2. External Accessories Framework
3. Accelerate Framework
4. Security Services Framework
5. Local Authorization Framework etc.

It supports 64 bit which enables the application to run faster.

CORE SERVICES Layer:
Some important frameworks are present in the CORE SERVICES Layer which helps the iOS operating system to cure itself and provide better functionality. It is the 2nd lowest layer in the Architecture as shown above. Below are some important frameworks present in this layer:

1. **Address Book Framework-**

   The Address Book Framework provides access to the contact details of the user.
2. **Cloud Kit Framework-**

   This framework provides a medium for moving data between your app and iCloud.
3. **Core Data Framework-**

   This is the technology that is used for managing the data model of a Model View Controller app.

4. **Core Foundation Framework-**

   This framework provides data management and service features for iOS applications.

5. **Core Location Framework-**

   This framework helps to provide the location and heading information to the application.

6. **Core Motion Framework-**

   All the motion-based data on the device is accessed with the help of the Core Motion Framework.

7. **Foundation Framework-**

   Objective C covering too many of the features found in the Core Foundation framework.

8. **HealthKit Framework-**

   This framework handles the health-related information of the user.

9. **HomeKit Framework-**

   This framework is used for talking with and controlling connected devices with the user's home.

10. **Social Framework-**

    It is simply an interface that will access users' social media accounts.

11. **StoreKit Framework-**

    This framework supports for buying of contents and services from inside iOS apps.

MEDIA Layer:

With the help of the media layer, we will enable all graphics video, and audio technology of the system. This is the second layer in the architecture. The different frameworks of MEDIA layers are:

1. **ULKit Graphics-**

   This framework provides support for designing images and animating the view content.

2. **Core Graphics Framework-**

   This framework support 2D vector and image-based rendering and it is a native drawing engine for iOS.

3. **Core Animation-**

   This framework helps in optimizing the animation experience of the apps in iOS.

4. **Media Player Framework-**

   This framework provides support for playing the playlist and enables the user to use their iTunes library.

5. **AV Kit-**

   This framework provides various easy-to-use interfaces for video presentation, recording, and playback of audio and video.

6. **Open AL-**

   This framework is an Industry Standard Technology for providing Audio.

7. **Core Images-**

   This framework provides advanced support for motionless images.

8. **GL Kit-**

   This framework manages advanced 2D and 3D rendering by hardware-accelerated interfaces.

COCOA TOUCH:

COCOA Touch is also known as the application layer which acts as an interface for the user to work with the iOS Operating system. It supports touch and motion events and many more features. The COCOA TOUCH layer provides the following frameworks :

1. **EvenKit Framework-**

   This framework shows a standard system interface using view controllers for viewing and changing events.

2. **GameKit Framework-**

   This framework provides support for users to share their game-related data online using a Game Center.

3. **MapKit Framework-**

   This framework gives a scrollable map that one can include in your user interface of the app.

4. **PushKit Framework-**

   This framework provides registration support.

Features of iOS operating System:

Let us discuss some features of the iOS operating system-

1. Highly Securer than other operating systems.

2. iOS provides multitasking features like while working in one application we can switch to another application easily.
3. iOS's user interface includes multiple gestures like swipe, tap, pinch, Reverse pinch.
4. iBooks, iStore, iTunes, Game Center, and Email are user-friendly.
5. It provides Safari as a default Web Browser.
6. It has a powerful API and a Camera.
7. It has deep hardware and software integration

Applications of IOS Operating System:

Here are some applications of the iOS operating system-

1. iOS Operating System is the Commercial Operating system of Apple Inc. and is popular for its security.
2. iOS operating system comes with pre-installed apps which were developed by Apple like Mail, Map, TV, Music, Wallet, Health, and Many More.
3. Swift Programming language is used for Developing Apps that would run on IOS Operating System.
4. In iOS Operating System we can perform Multitask like Chatting along with Surfing on the Internet.

Advantages of IOS Operating System:

The iOS operating system has some advantages over other operating systems available in the market especially the Android operating system. Here are some of them-

1. More secure than other operating systems.
2. Excellent UI and fluid responsive
3. Suits best for Business and Professionals
4. Generate Less Heat as compared to Android.

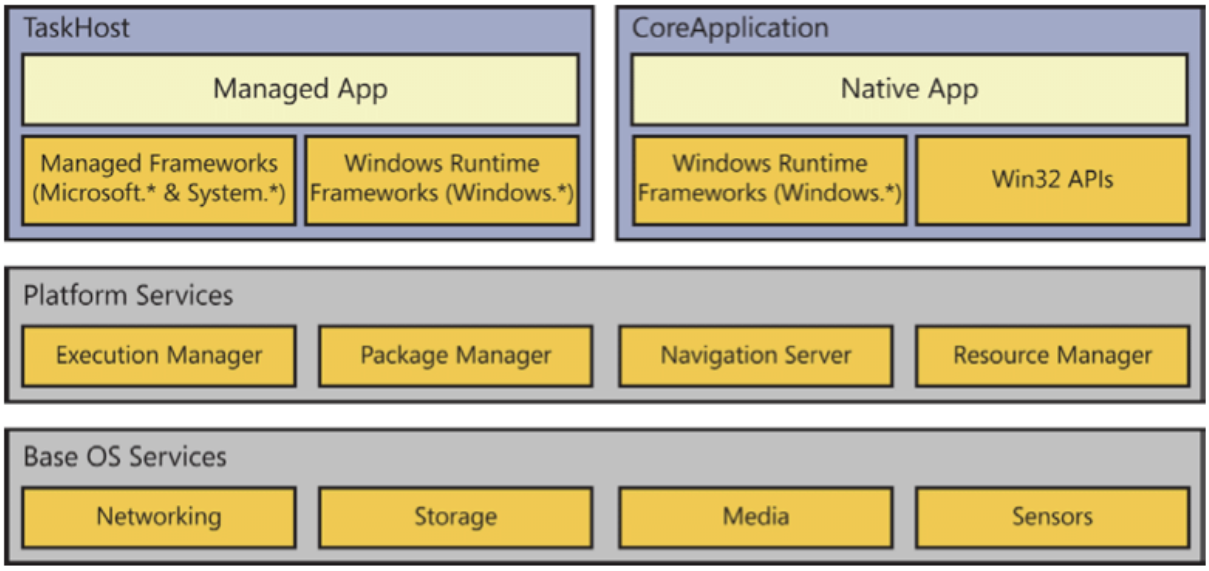
Disadvantages of IOS Operating System:
Let us have a look at some disadvantages of the iOS operating system-

1. More Costly.
2. Less User Friendly as Compared to Android Operating System.
3. Not Flexible as it supports only IOS devices.
4. Battery Performance is poor

1.4. Windows Architecture

Previously, the mobile operating system created by Microsoft was called Windows Mobile. After the changes introduced by Apple (iOS) and Google (Android) in 2007, Microsoft decided to take a new direction and created Windows Phone. Similar to other alternatives, such as iOS and Android, Windows Phone is an operating system for smartphones. It is usually used on touch screen devices, and offers functionality such as networking, sensors and camera integration. Programs for Windows Phone 7 are written in .NET managed code. Managed code is code written in languages that are available for use with the Microsoft .NET Framework, for example C#. One of the benefits is that many of the error-prone and often complex tasks, such as type safety checking, memory management and destruction of unneeded objects, are taken care of. Windows Phone 7 supports two popular programming platforms, namely Silverlight and XNA. Silverlight is an evolution of the Windows Presentation Foundation (WPF). It provides developers with the ability to create

sophisticated user interfaces. The second platform, XNA, is Microsoft's game platform. It supports both 2D and 3D graphics. Development for Windows Phone is done in Visual Studio. There is a range of various editions of Visual Studio, ranging from the free Visual Studio Express to the Ultimate edition. Although the Express edition if enough to get started, the limitations quickly get in the way of productivity. For example, no support for plugins is one of the main limitations. There are two languages that can be used to write programs for Windows Phone, 1) Visual Basic .NET and 2) C#. We will focus on the C# language in this paper. We chose to use this language because we were more familiar with it and also we found more resources, in books and on the Internet, for Windows Phone development with C#. Programs created for Windows Phone are packaged into XAP files, which is the Silverlight application package. According to Gartner, Microsoft currently occupies the 3rd place in regards to market share (second quarter of 2013). For the first time Microsoft has a larger market share compared to Blackberry. Even with the recent increase in popularity, the Windows Phone platform is still a relatively small player with a 3.3% market share. The step up of the iOS (14.2%) and Android (79.0%) is considerable. However, it will be interesting to see how the acquiring of Nokia will affect the further development of Windows Phone and the mobile devices.



1.5. Hybrid Architecture

What is hybrid application:

- A hybrid application is a software application that combines elements of both native apps and web applications.
- It is the combination of web apps and native apps which needs to be downloaded inside your devices like native apps but the program that are used to build the Hybrid application are written in HTML, CSS and JavaScript.
- The browser of our devices access is HTML, JavaScript and native APIs to the particular hardware.
- It can run both online and offline, if the hybrid software does not depend on data from the database then it can be use offline.
- It runs across various platform but need to deploy the app's wrapper.
- Example of hybrid application: Uber, Ola, Twitter etc.

What is Native app:

- Native apps are created by software programs (like java, kotlin, ruby etc) that runs on the particular devices and platforms.
- We need to download the native apps from the app stores (Google play, Apple's store) as it does not run in the browser Unlike our smartphones where each application after installation from the app stores have access to an icon on the screen of our device home screen.
- Native apps are developed specifically for one platform and can accessed all the features of our devices like camera, file manager, contacts, GPS etc.
- They are various platform on which we can build Native apps like desktop, smartphones, smartwatch etc.
- Native apps can work offline by using system notification.
- Example of Native apps: Facebook, WhatsApp etc.

What is web application:

- Web application is a software or program that runs and accessible using web browser unlike Native app which run on particular devices.
- We don't need any particular SDK for developing web application. Frontend part is mostly created using HTML, CSS, JavaScript, bootstrap etc. and backend part could use MEAN stack, Hibernate etc.
- Unlike Native app, web application cannot be installed as it run inside the browser.
- Web application are connected through the server. The server requires bandwidth which helps web application to run on the browser all the time.
- Client displays the data of web application and take very little disk space on the client side. If the server connection get lost then the whole data may be lost.
- Example of web application: MakeMyTrip, Oyo, Flipkart, Amazon etc.

Advantages and Disadvantages of hybrid apps:

Advantage:

- Here we are free to use the Hybrid app across various platform.
- It can be work both offline and online.
- Hybrid apps comes with various updates across the time which improves the quality of apps.
- It is cheaper to develop unlike native apps where it cost double to develop two versions for two platforms.

Disadvantage:

- As hybrid application works on various platform, the GUI or the appearance may differ from platform to platform from user side.
- The hybrid application needs to be tested across various platform, it runs to ensure proper operation of app.

Various frameworks used for Hybrid application:

1. React Native framework:

- It is the most popular framework for hybrid application development.
- It supports various IDE and tools for the development.
- One of the best thing about React Native is that you can see the result of the code.
- Because of the faster result, it is time efficient.

2. Flutter:

- It is very easy to use and implement. For the developer who is new can easily get their hands on it.
- Flutter bear various languages thus it helps the developer to use the language of their own on various platform.

3. Ionic:

- It is best used for mobile app development.
- It uses HTML, CSS and JavaScript.
- It is used open source HTML5 development platform.
- It uses single database for the development of hybrid application.

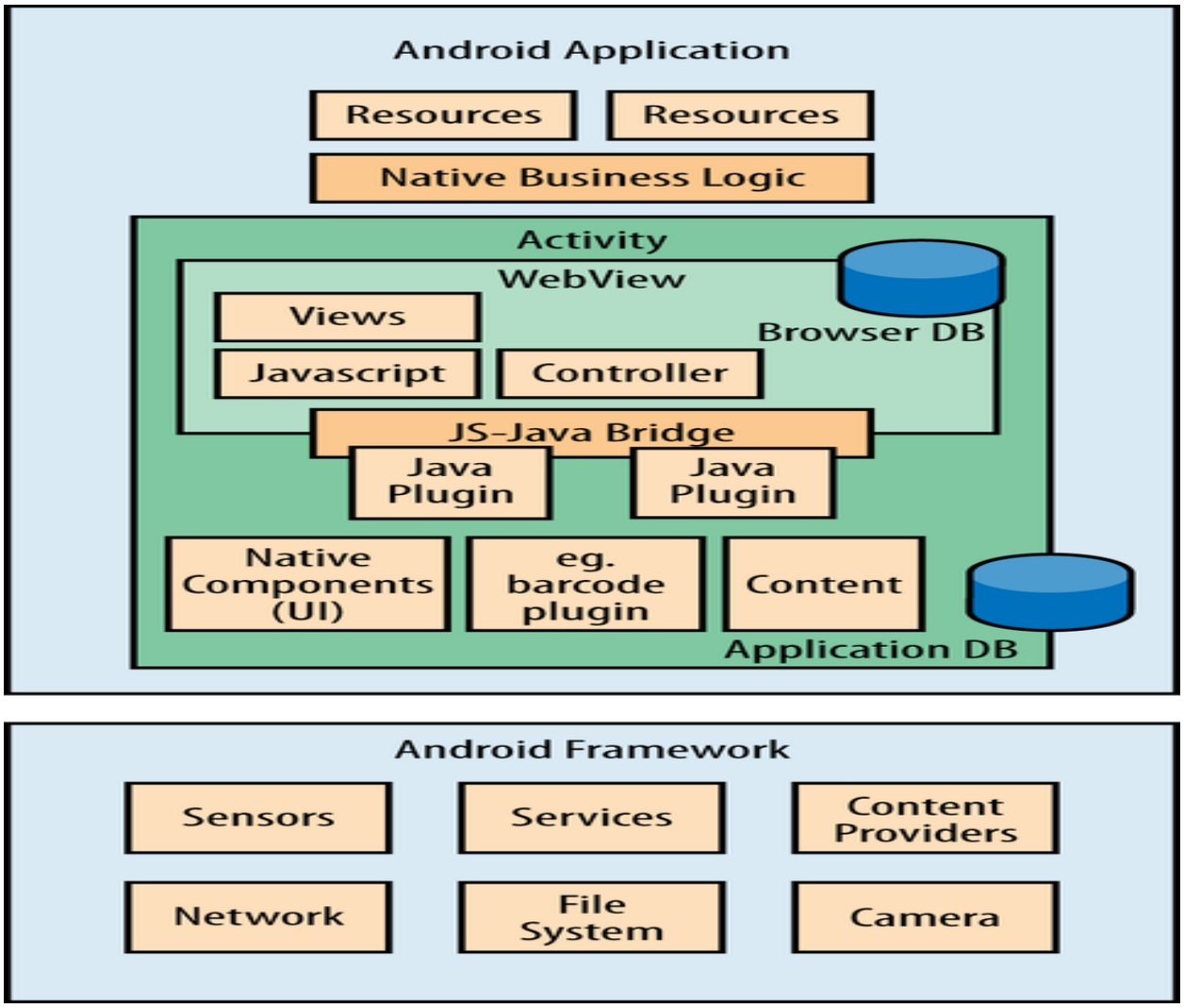
4. jQuery Mobile:

- It is the JavaScript framework fully dependent on the plugins available in JavaScript like Content Slider, Image Slider, Pop-up Boxes, etc.
- It is easier to implement as compared to other JavaScript libraries.
- In this very less code is required to program.

5. Appcelerator Titanium:

- The benefit to use Appcelerator Titanium is that it uses its own API i.e. it has independent API that easily access device hardware.
- It can be reused across different platforms and apps.

- It receives UI component.



Unit 2.Creating Android Application

- 2.1. Creating Android project
- 2.2. Project Structure
- 2.3. Activity and Activity Life Cycle
- 2.4. Fragment and Fragment Life Cycle
- 2.5 Views and View groups

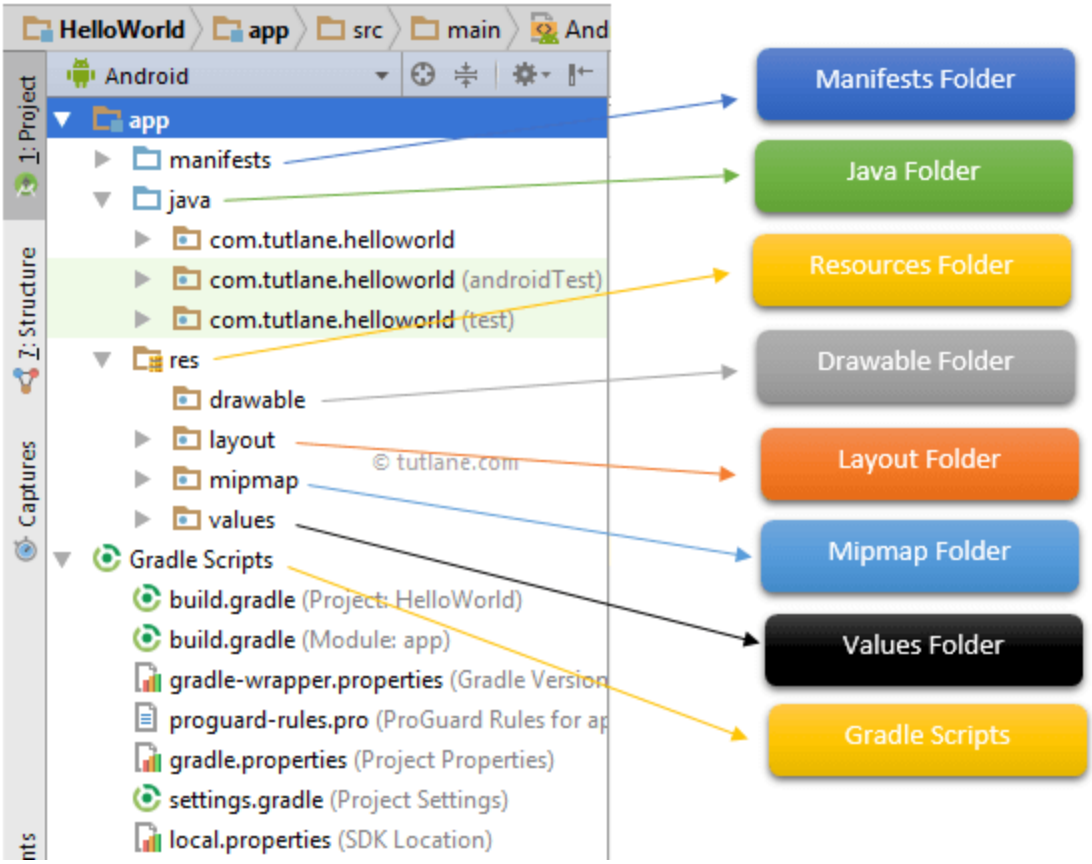
2.1. Creating Android project:

https://developer.android.com/codelabs/build-your-first-android-app#0

2.2. Project Structure:

To implement android apps, Android Studio is the official IDE (Integrated Development Environment) which is freely provided by Google for android app development.

Once we setup android development environment using android studio and if we create a sample application using android studio, our project folder structure will be like as shown below. In case if you are not aware of creating an application using an android studio please check this Android Hello World App.

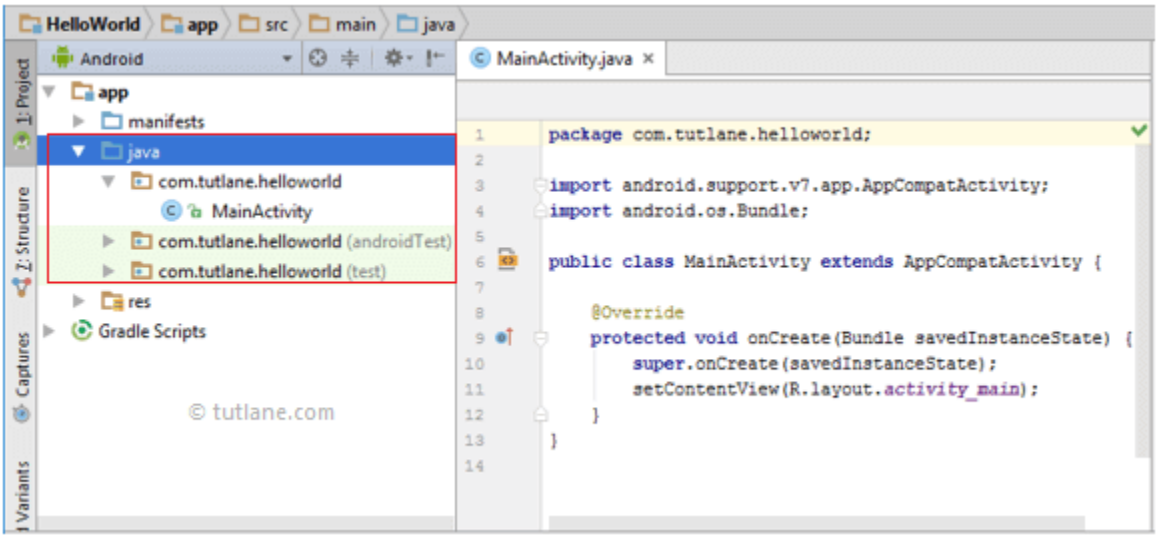


The Android project structure on the disk might be differs from the above representation. To see the actual file structure of the project, select Project from the Project dropdown instead of Android.

The android app project will contain different types of app modules, source code files, and resource files. We will explore all the folders and files in the android app.

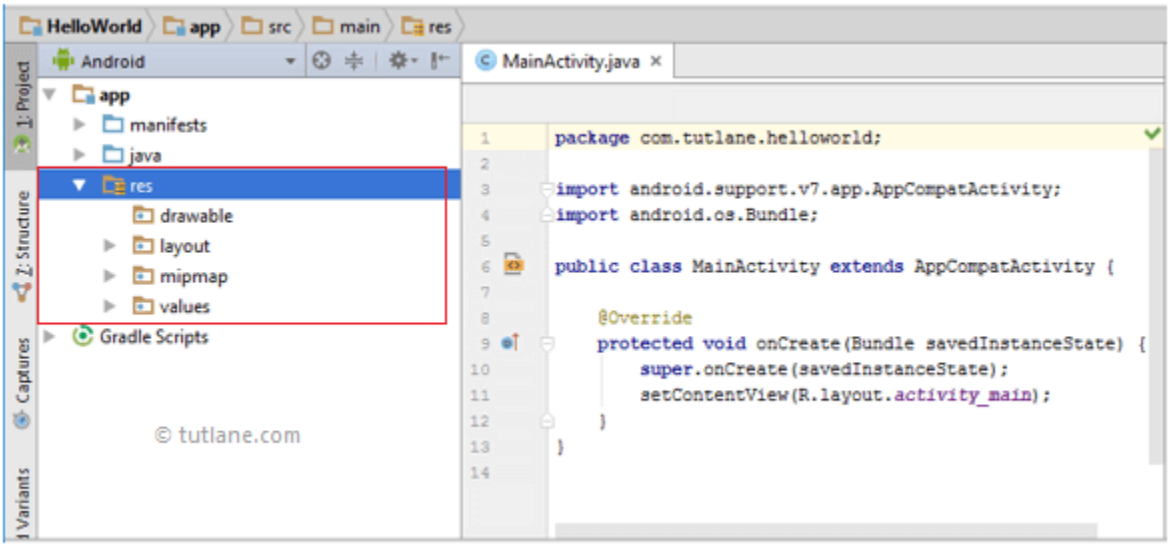
Java Folder

This folder will contain all the java source code (.java) files which we'll create during the application development, including JUnit test code. Whenever we create any new project/application, by default the class file MainActivity.java will create automatically under the package name "com.tutlane.helloworld" like as shown below.



res (Resources) Folder

It's an important folder that will contain all non-code resources, such as bitmap images, UI strings, XML layouts like as shown below.



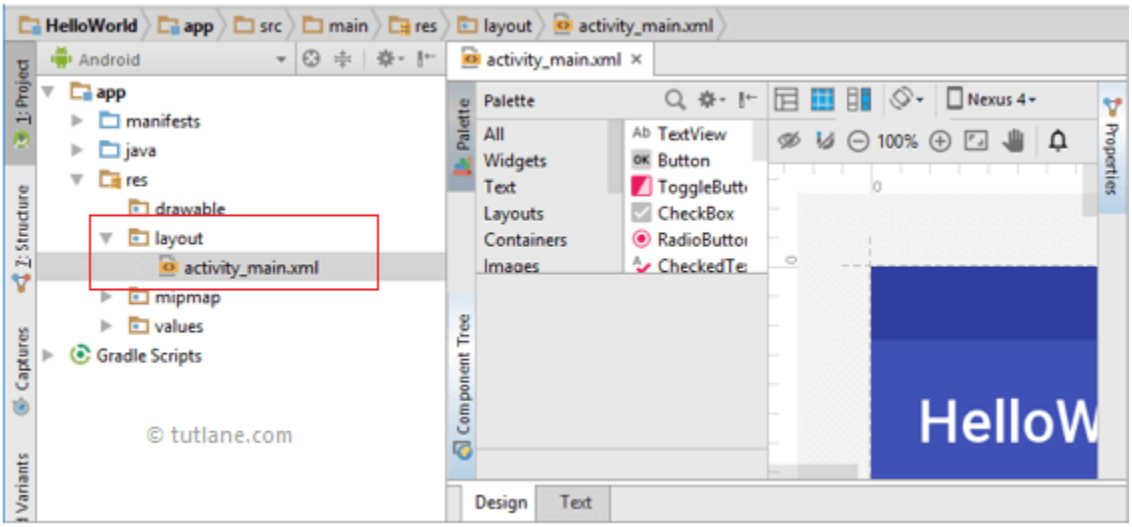
The res (Resources) will contain a different type of folders that are

Drawable Folder (res/drawable)

It will contain the different types of images as per the requirement of application. It's a best practice to add all the images in a drawable folder other than app/launcher icons for the application development.

Layout Folder (res/layout)

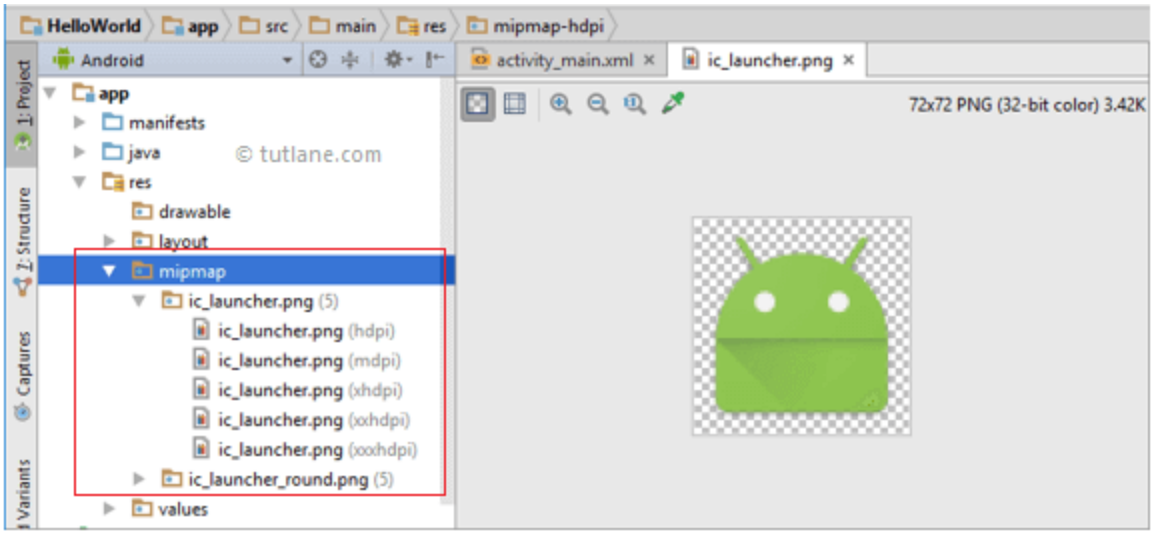
This folder will contain all XML layout files which we used to define the user interface of our application. Following is the structure of the layout folder in the android application.



Mipmap Folder (res/mipmap)

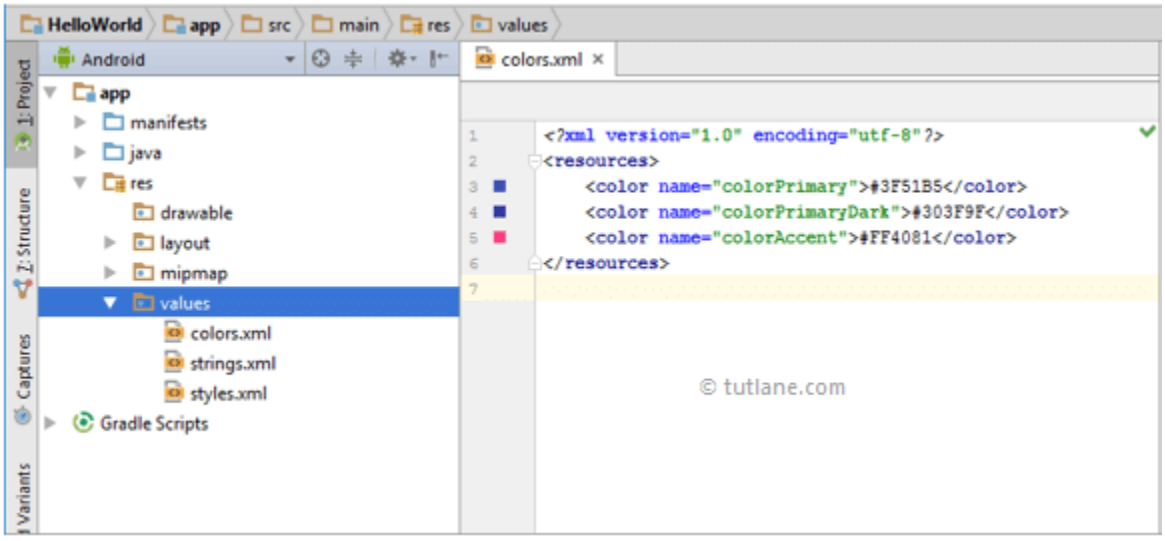
This folder will contain app / launcher icons that are used to show on the home screen. It will contain different density type of icons such as hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi, to use different icons based on the size of the device.

Following is the structure of the mipmap folder in the android application.



Values Folder (res/values)

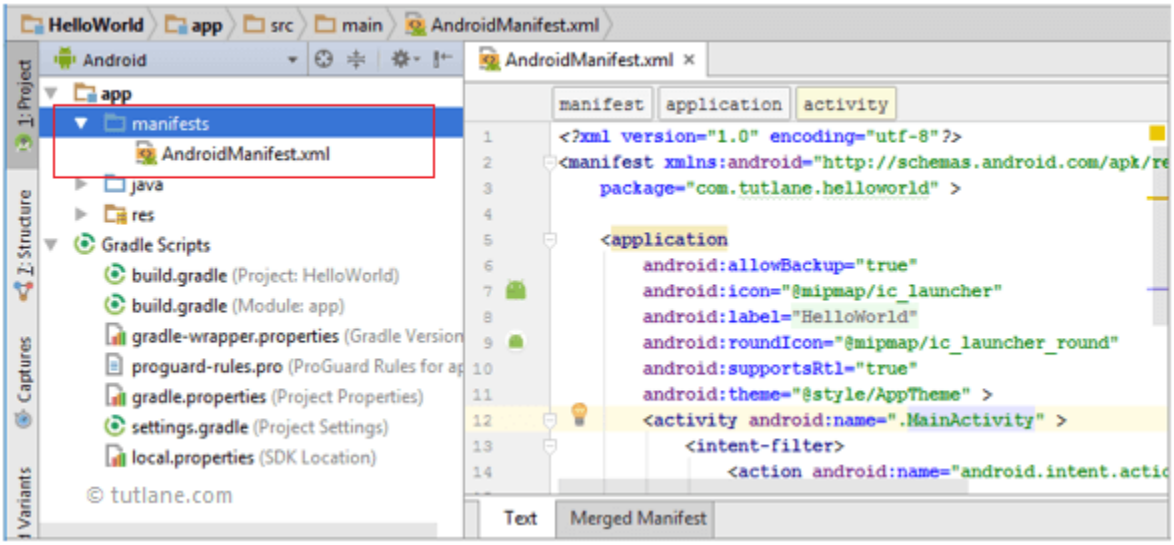
This folder will contain various XML files, such as strings, colors, style definitions and a static array of strings or integers. Following is the structure of the values folder in android application.



Manifests Folder

This folder will contain a manifest file (AndroidManifest.xml) for our android application. This manifest file will contain information about our application such as android version, access permissions, metadata, etc. of our application and its components. The manifest file will act as an intermediate between android OS and our application.

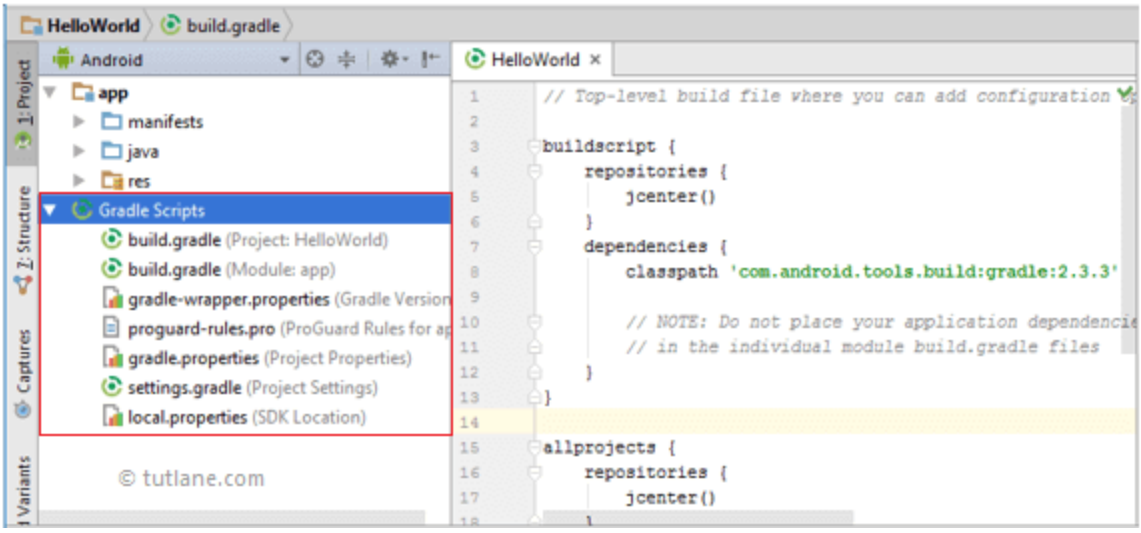
Following is the structure of the manifests folder in the android application.



Gradle Scripts

In android, Gradle means automated build system and by using this we can define a build configuration that applies to all modules in our application. In Gradle build.gradle (Project), and build.gradle (Module) files are useful to build configurations that apply to all our app modules or specific to one app module.

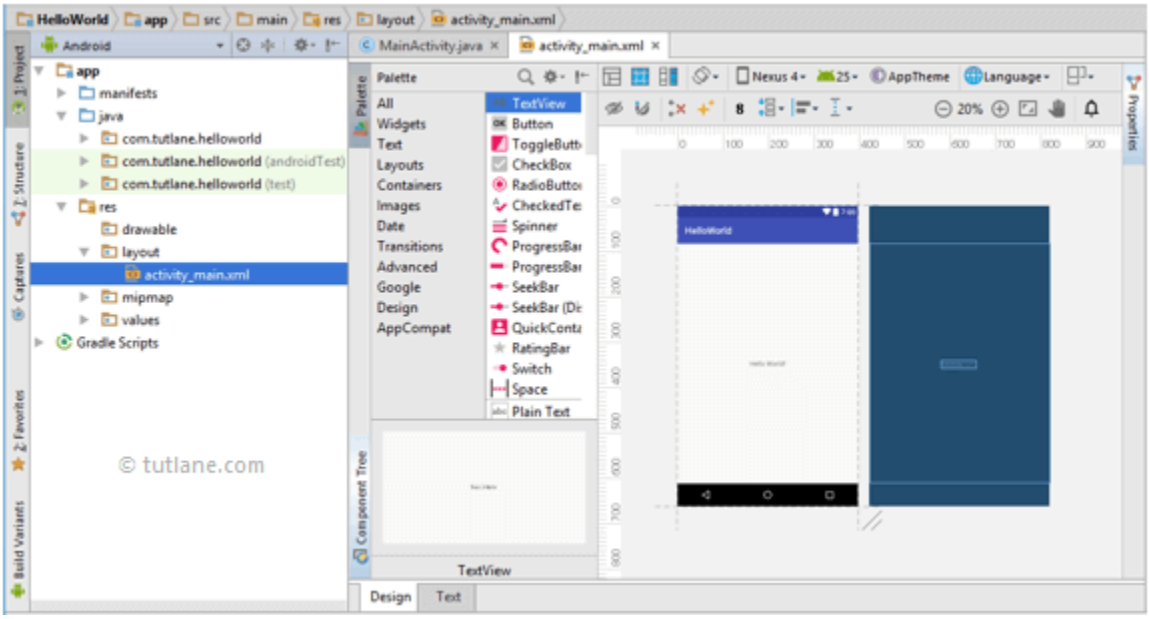
Following is the structure of Gradle Scripts in the android application.



Following are the important files which we need to implement an app in android studio.

Android Layout File (activity_main.xml)

The UI of our application will be designed in this file and it will contain Design and Text modes. It will exists in the layouts folder and the structure of activity_main.xml file in Design mode like as shown below.



We can make required design modifications in activity_main.xml file either using Design or Text modes. If we switch to Text mode activity_main.xml file will contain a code like as shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="com.tutlane.helloworld.MainActivity">
        <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintRight_toRightOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Android Main Activity File (MainActivity.java)

The main activity file in the android application is MainActivity.java and it will exist in the java folder. The MainActivity.java file will contain the java code to handle all the activities related to our app.

Following is the default code of MainActivity.java file which is generated by our HelloWorld application.

```
package com.tutlane.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        }
}
```

Android Manifest File (AndroidManifest.xml)

Generally, our application will contain multiple activities and we need to define all those activities in the AndroidManifest.xml file. In our manifest file, we need to mention the main activity for our app using the MAIN action and LAUNCHER category attributes in intent filters (<intent-filter>). In case if we didn't mention MAIN action or LAUNCHER category for the main activity, our app icon will not appear in the home screen's list of apps.

Following is the default code of AndroidManifest.xml file which is generated by our HelloWorld application.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.tutlane.helloworld" >
        <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
        <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        </activity>
```

```
        </application>

</manifest>
```

These are the main folders and files required to implement an application in android studio. If you want to see the actual file structure of the project, select Project from the Project dropdown instead of Android.

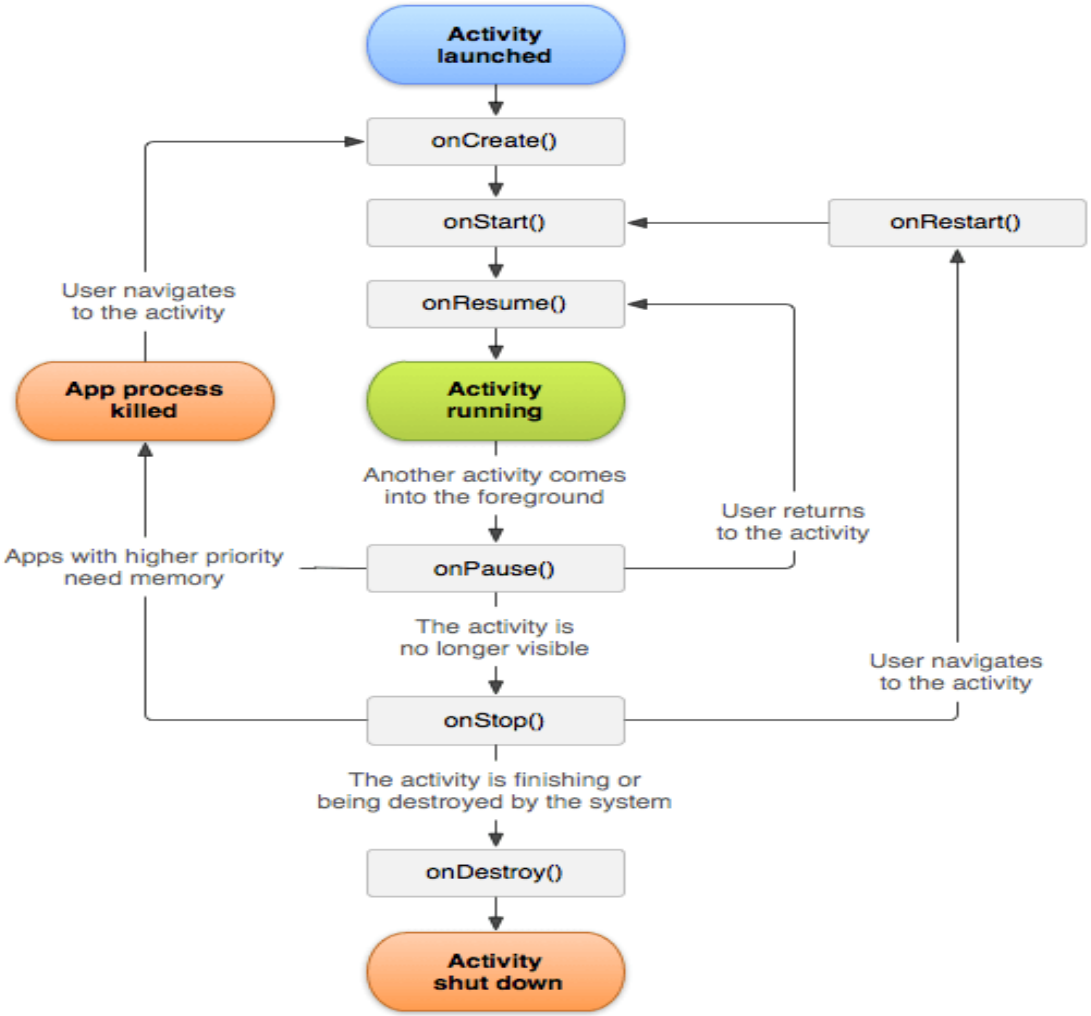
2.3. Activity and Activity Life Cycle:

Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.

By the help of activity, you can place all your UI components or widgets in a single screen.

If you have worked with C, C++ or Java programming language then you must have seen that your program starts from main() function. Very similar way, Android system initiates its program with in an Activity starting with a call on *onCreate()* callback method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram



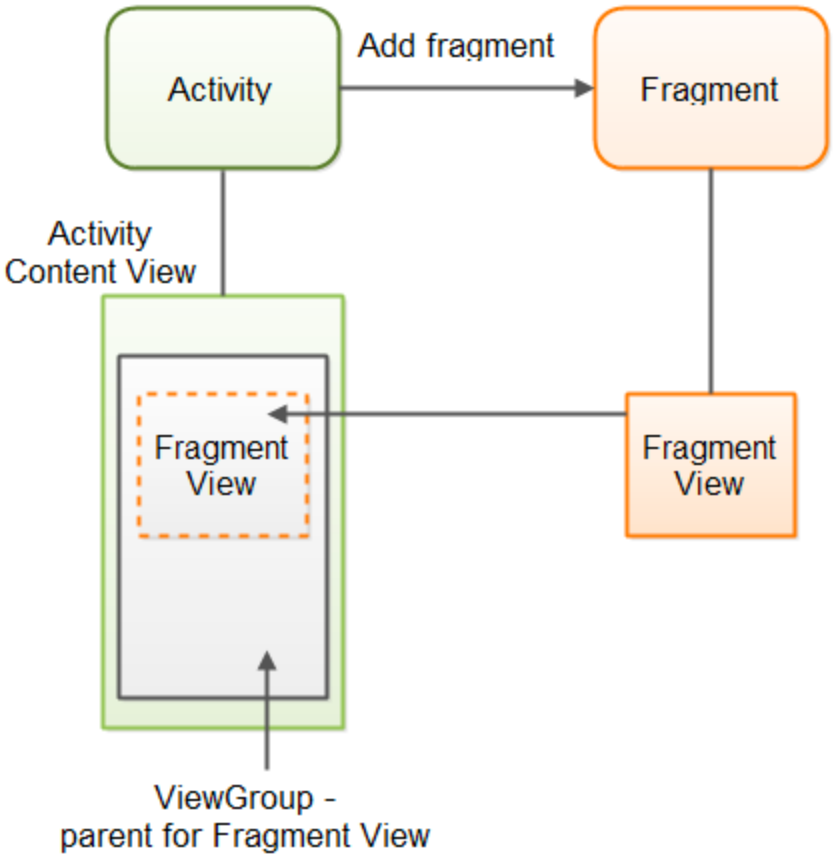
Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.

onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.

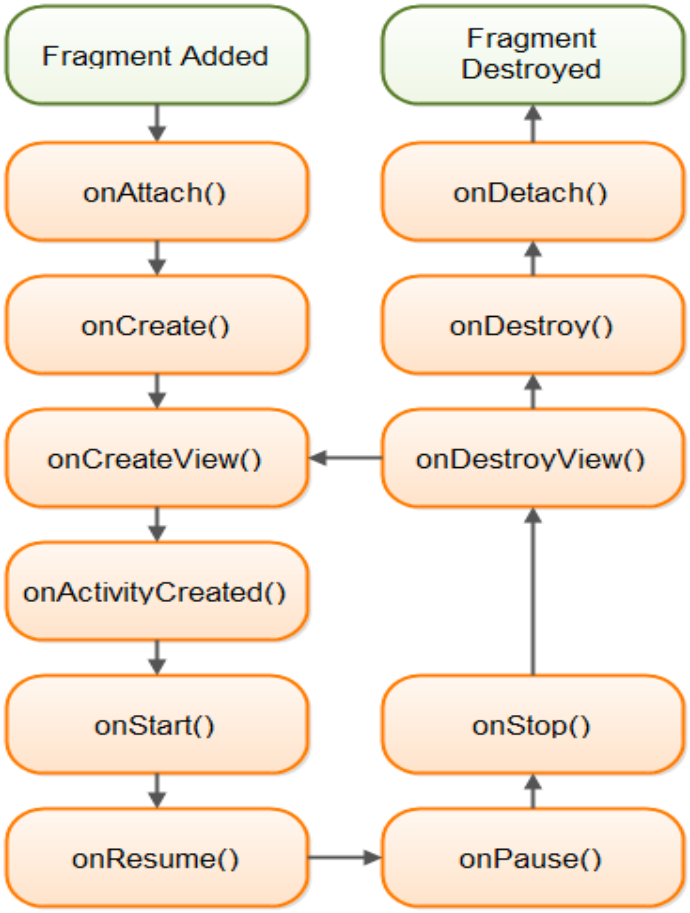
2.4. Fragment and Fragment Life Cycle:

In Android, the fragment is the part of Activity which represents a portion of User Interface(UI) on the screen. It is the modular section of the android activity that is very helpful in creating UI designs that are flexible in nature and auto-adjustable based on the device screen size. The UI flexibility on all devices improves the user experience and adaptability of the application. Fragments can exist only inside an activity as its lifecycle is dependent on the lifecycle of host activity. For example, if the host activity is paused, then all the methods and operations of the fragment related to that activity will stop functioning, thus fragment is also termed as sub-activity. Fragments can be added, removed, or replaced dynamically i.e., while activity is running.

<fragment> tag is used to insert the fragment in an android activity layout. By dividing the activity's layout multiple fragments can be added in it.



Fragment Lifecycle:



Below are the methods of fragment lifecycle.

1. **onAttach()** :This method will be called first, even before onCreate(), letting us know that your fragment has been attached to an activity. You are passed the Activity that will host your fragment
2. **onCreateView()** : The system calls this callback when it's time for the fragment to draw its UI for the first time. To draw a UI for the fragment, a View component must be returned from this method which is the root of the fragment's layout. We can return null if the fragment does not provide a UI
3. **onViewCreated()** : This will be called after onCreateView(). This is particularly useful when inheriting the onCreateView() implementation

but we need to configure the resulting views, such as with a ListFragment and when to set up an adapter

4. **onActivityCreated()** :This will be called after onCreate() and onCreateView(), to indicate that the activity's onCreate() has completed. If there is something that's needed to be initialised in the fragment that depends upon the activity's onCreate() having completed its work then onActivityCreated() can be used for that initialisation work
5. **onStart()** : The onStart() method is called once the fragment gets visible
6. **onResume()** : This method is called to make the visible fragment interactive.
7. **onPause()** : The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session
8. **onStop()** : Fragment going to be stopped by calling onStop()
9. **onDestroyView()** : It's called before onDestroy(). This is the counterpart to onCreateView() where we set up the UI. If there are things that are needed to be cleaned up specific to the UI, then that logic can be put up in onDestroyView()
10. **onDestroy()** : onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.
11. **onDetach()** : It's called after onDestroy(), to notify that the fragment has been disassociated from its hosting activity

2.5 Views and View groups:

View: The view is the component which Android provides us to design the layouts of the app. So, we can understand the view as a rectangular area which

is going to contain some element inside it. A *View* is a superclass for all the UI components.

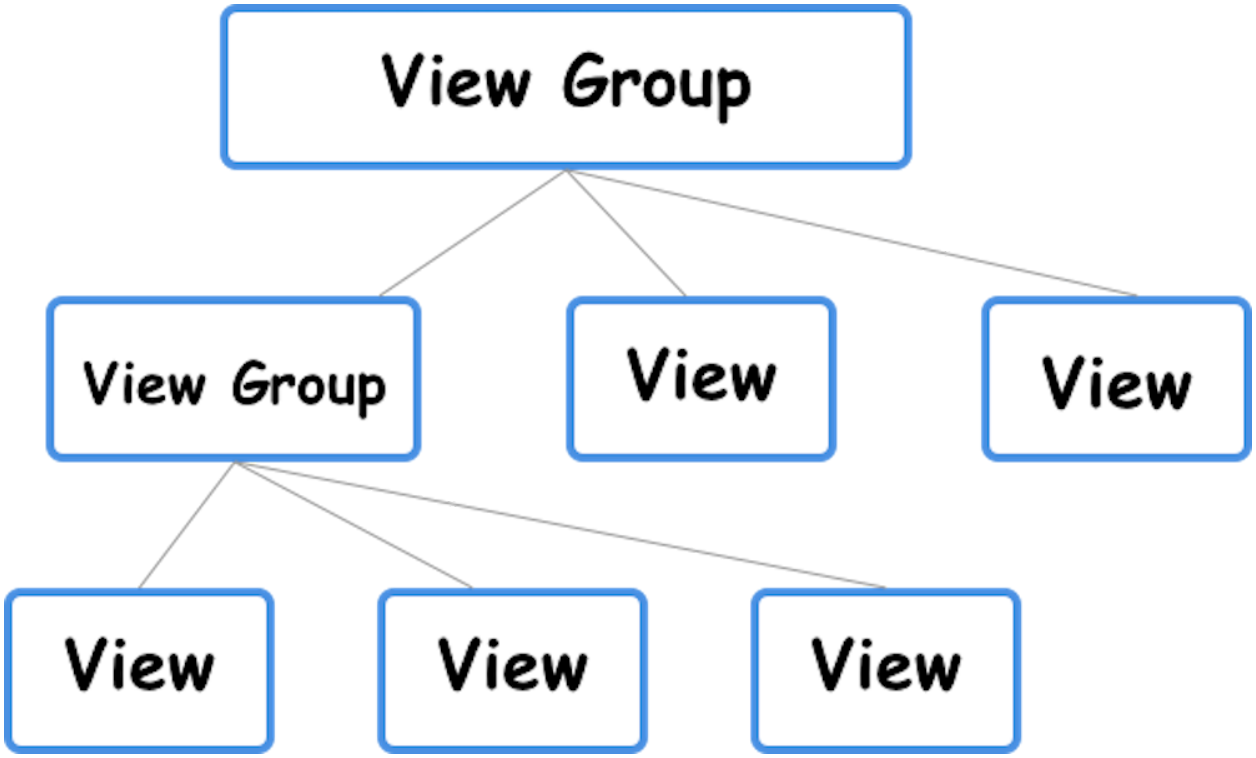
examples:

- **TextView:** To add some text in your application.
- **EditText:** This is used when you want to take some input from the users.
- **ImageView:** To add some image in the application.
- **ProgressBar:** To show the progress to something. For example, the loading screen.
- **Button:** Buttons are used to trigger some action on the click of the button. It can be starting a new activity or something else.
- **ImageButton:** It is used to make a clickable image.
- **CheckBox:** CheckBox is used to select some options out of many available options.
- **DatePicker:** To select some particular date.

View Group: A group of views is known as *ViewGroup*. The Top-level ViewGroup is a parent, and under it, all the view and other view groups are its children. *For example*, under a LinearLayout, you can add two Buttons and one EditText. Here, LinearLayout is the parent view and the Buttons and EditTexts are the children.

examples:

- **Linear Layout**
- **Relative Layout**
- **Constraint Layout**
- **Radio Group etc**



Views:

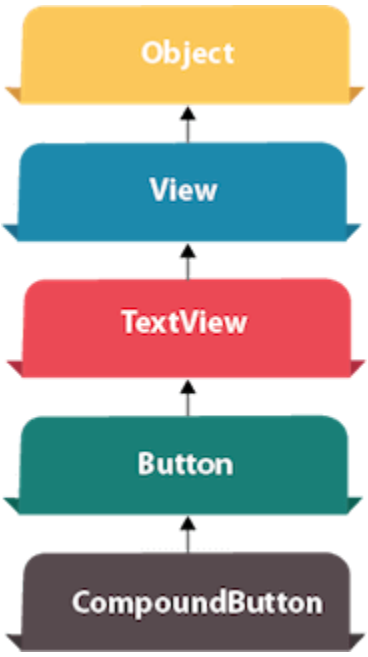
Button:

In android, Button is a user interface control that is used to perform an action whenever the user clicks or taps on it.

Generally, Buttons in android will contain a text or an icon or both and perform an action when the user touches it.

Following is the pictorial representation of using Buttons in android applications.

In android, we have a different type of buttons available to use based on our requirements, those are ImageButton, ToggleButton, RadioButton.
In android, we can create a Button control in two ways either in the XML layout file or create it in the Activity file programmatically.



Example:

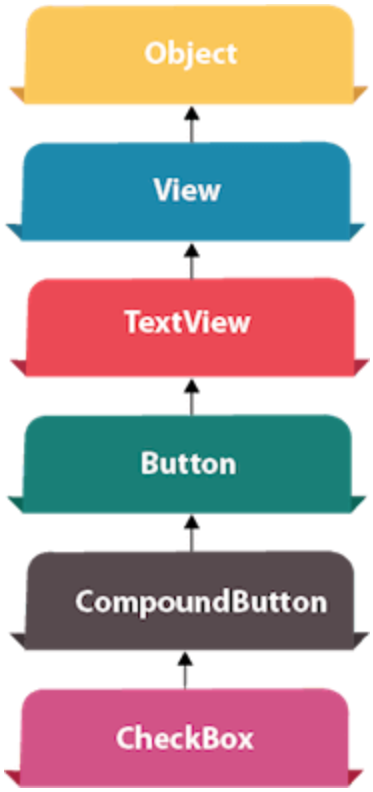
```
<Button
android:id="@+id/simpleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Kailas"/>
```

Check Box:

Android CheckBox is a type of two state button either checked or unchecked.

There can be a lot of usage of checkboxes. For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.

Android CheckBox class is the subclass of CompoundButton class.



Example:

```
<CheckBox
android:id="@+id/simpleCheckBox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Simple CheckBox"/>
```

Radio Button:

In Android, RadioButton are mainly used together in a RadioGroup. In RadioGroup checking the one radio button out of several radio buttons added in it will automatically unchecked all the others. It means at one time we can check only one radio button from a group of radio buttons which belong to the same radio group. The most common use of the radio button is in Quiz Android App code.

Radio Button is a two state button that can be checked or unchecked. If a radio button is unchecked then a user can check it by simply clicking on it. Once a RadioButton is checked by the user it can't be unchecked by simply pressing on the same button. It will automatically unchecked when you press any other RadioButton within the same RadioGroup.

Important Note: RadioGroup is a widget used in Android for the grouping of radio buttons and provides the feature of selecting only one radio button from the set. When a user tries to select any other radio button within the same radio group the previously selected radio button will be automatically unchecked.

RadioGroup And RadioButton code in XML:

```
<RadioGroup
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
    <RadioButton
    android:id="@+id/simpleRadioButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

```
    <RadioButton
    android:id="@+id/simpleRadioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
</RadioGroup>
```

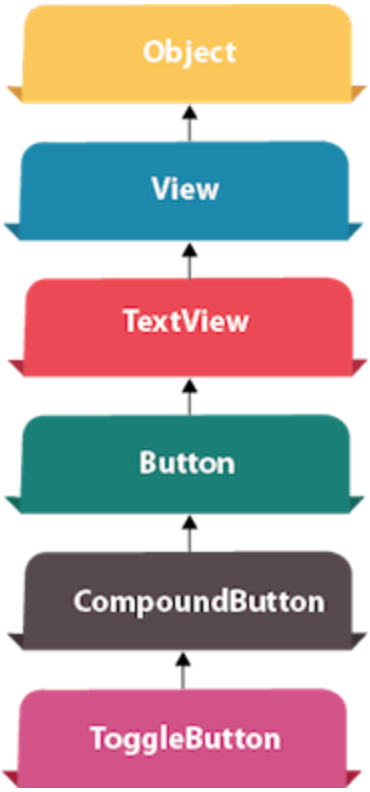
Toggle Button:

Android Toggle Button can be used to display checked/unchecked (On/Off) state on the button.

It is beneficial if users have to change the setting between two states. It can be used for On/Off Sound, Wifi, Bluetooth etc.

Since Android 4.0, there is another type of toggle button called *switch* that provides slider control.

Android ToggleButton and Switch both are the subclasses of CompoundButton class.



Example:

```
<ToggleButton
android:id="@+id/simpleToggleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

Image Button:

In Android, ImageButton is used to display a normal button with a custom image in a button. In simple words we can say, ImageButton is a button with an image that can be pressed or clicked by the users. By default it looks like a normal

button with the standard button background that changes the color during different button states.

An image on the surface of a button is defined within a xml (i.e. layout) by using src attribute or within a java class by using setImageResource() method. We can also set an image or custom drawable in the background of the image button.

Important Note: Standard button background image is displayed in the background of the button whenever you create an image button. To remove that image, you can define your own background image in xml by using background attribute or in java class by using setBackground() method.

Below is the code and image which shows how custom image button looks in Android:

Important Note: ImageButton has all the properties of a normal button so you can easily perform any event like click or any other event which you can perform on a normal button.

ImageButton code in XML:

```
<!--Make Sure To Add Image Name home in Drawable Folder-->

<ImageButton

android:id="@+id/simpleImageButton"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:src="@drawable/home" />
```

Text Fields(Edit Text):

In Android, EditText is a standard entry widget in android apps. It is an overlay over TextView that configures itself to be editable. EditText is a subclass of TextView with text editing operations. We often use EditText in our applications in order to provide an input or text field, especially in forms. The most simple example of EditText is Login or Sign-in form.

Important Note: An EditText is simply a thin extension of a TextView. An EditText inherits all the properties of a TextView.

We can create an EditText instance by declaring it inside a layout(XML file) or by instantiating it programmatically (i.e. in Java Class).

EditText code in XML:

```
<EditText

android:id="@+id/simpleEditText"

android:layout_height="wrap_content"

android:layout_width="match_parent"/>
```

Retrieving / Getting the Value From EditText In Java Class:

Below is the example code of EditText in which we retrieve the value from an EditText in Java class. We have used this code in the example.

```
EditText simpleEditText = (EditText) findViewById(R.id.simpleEditText);

String editTextValue = simpleEditText.getText().toString();
```

Spinner:

In Android, Spinner provides a quick way to select one value from a set of values. Android spinners are nothing but the drop down-list seen in other programming languages. In a default state, a spinner shows its currently selected value. It provides an easy way to select a value from a list of values.

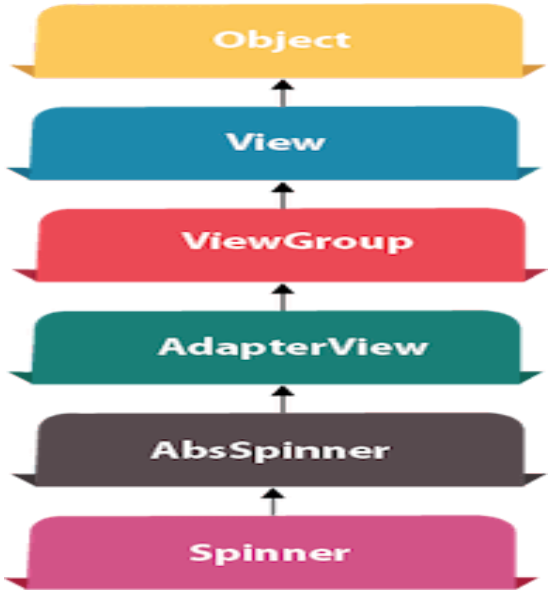
In Simple Words we can say that a spinner is like a combo box of AWT or swing where we can select a particular item from a list of items. Spinner is a subclass of AsbSpinner class.

Important Note: Spinner is associated with Adapter view so to fill the data in spinner we need to use one of the Adapter classes.

Here is the XML basic code for Spinner:

```
<Spinner

android:id="@+id/simpleSpinner "

android:layout_width="fill_parent"

android:layout_height="wrap_content" />
```

Important Note: To fill the data in a spinner we need to implement an adapter class. A spinner is mainly used to display only text fields so we can implement an Array Adapter for that. We can also use Base Adapter and other custom adapters to display a spinner with a more customized list. Suppose if we need to display a textview and an imageview in the spinner item list then an array adapter is not enough for that. Here we have to implement a custom adapter in our class. Below image of Spinner and Custom Spinner will make it more clear.



Java File Code:

```
String[] country = {"India", "USA", "China", "Japan", "Other"};

Spinner spinner= findViewById(R.id.spinner);
```

```
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, country);

arrayAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
pdown_item);

spinner.setAdapter(arrayAdapter);
```

List View:

List of scrollable items can be displayed in Android using ListView. It helps you to display the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so we do not need to use scroll View or anything else with ListView.

ListView is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.

Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.

```
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/simpleListView"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   tools:context="abhiandroid.com.listexample.MainActivity">
</ListView>
```

Java File Code:

```
ArrayList<String> items = new ArrayList<String>();
items.add("Android");
items.add("Java");
items.add("Kotlin");

ArrayAdapter<String>itemsAdapter=newArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, items);
```

```
ListView listView = findViewById(R.id.listView);

listView.setAdapter(itemsAdapter);
```

Toast:

Android Toast can be used to display information for a short period of time. A toast contains a message to be displayed quickly and disappears after some time.

The android.widget.Toast class is the subclass of java.lang.Object class.

You can also create custom toast as well for example toast displaying images. You can visit the next page to see the code for custom toast.

Toast class is used to show notification for a particular interval of time. After some time it disappears. It doesn't block the user interaction.

Example:

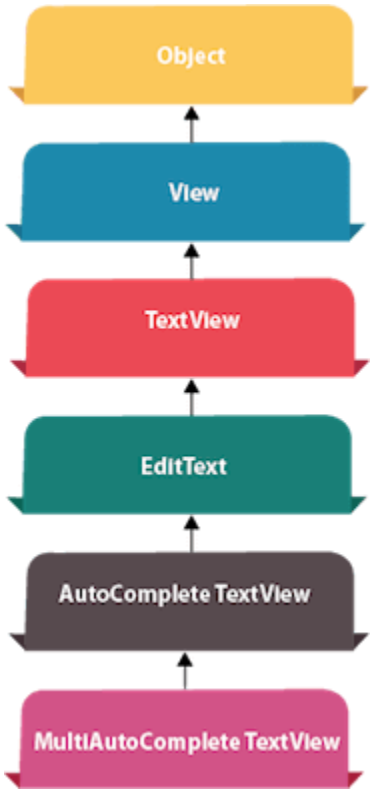
```
Toast.makeText(context, ,"Hello, I am Toast",Toast.LENGTH_SHORT).show();
```

Auto Complete TextView:

Android AutoCompleteTextView completes the word based on the reserved words, so no need to write all the characters of the word.

Android AutoCompleteTextView is an editable text field, it displays a list of suggestions in a drop down menu from which the user can select only one suggestion or value.

Android AutoCompleteTextViewis the subclass of EditText class. The MultiAutoCompleteTextView is the subclass of AutoCompleteTextView class.



ViewGroups:

Linear Layout:

Linear means in a line, either horizontal or vertical. We can understand here as all the elements inside the linear layout get arranged linearly, one after the other. If you are using horizontal orientation, then all the views inside the LinearLayout will be arranged horizontally one after the other.

For example: A Button is there on the screen, and now we need to put some text below the button. Here, we need to define the orientation of the LinearLayout to be vertical.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Button" />

    <TextView

        android:id="@+id/textView2"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="MindOrks" />

</LinearLayout>
```

Relative Layout:

Relative means *concerning another*, or we can understand this as a *relative to one another.* In this layout, all the components are arranged concerning each other.

For example: If we have one *Button* on the screen and now we want to put the *TextView* below the button i.e. the *TextView* is relatively below the *Button*. Following is the code of doing the same:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout      xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"

   android:layout_height="match_parent">
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me!!" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button2"
        android:text="KAILAS" />
</RelativeLayout>
```

Table Layout:

TableLayout positions its children into rows and columns. TableLayout containers do not display border lines for their rows, columns, or cells. The table will have as many columns as the row with the most cells. A table can leave cells empty. Cells can span multiple columns, as they can in HTML.

TableRow objects are the child views of a TableLayout (each TableRow defines a single row in the table). Each row has zero or more cells, each of which is defined by any kind of other View. So, the cells of a row may be composed of a variety of View objects, like ImageView or TextView objects. A cell may also be a ViewGroup object (for example, you can nest another TableLayout as a cell).

The following sample layout has two rows and two cells in each.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_open"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
```

```
    <TableRow>
        <TextView
            android:text="@string/table_layout_4_save"
            android:padding="3dip" />
        <TextView
            android:text="@string/table_layout_4_save_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

Grid Layout:

GridLayout is a type of android layout to create android applications which display widgets and text fields in grid format. Grid layout is usable with rows and columns like application developers can define how many rows and columns will be created in layout files. It's like creating a matrix layout in an android application. So here is the complete step by step tutorial for Android GridLayout Example Tutorial.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:columnCount="4"
 android:rowCount="3"
 tools:context="com.android_examples.com.gridlayout.MainActivity" >
```

```
<TextView

android:id="@+id/textView1"

android:layout_row="0"

android:layout_column="0"

android:text="A"

android:textAppearance="?android:attr/textAppearanceLarge"

android:padding="30dp"/>


<TextView

android:id="@+id/textView2"

android:layout_row="0"

android:layout_column="1"

android:text="B"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView

android:id="@+id/textView3"

android:layout_row="0"
```

```
android:layout_column="2"

android:text="C"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView

android:id="@+id/textView4"

android:layout_row="0"

android:layout_column="3"

android:text="D"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


//2nd row starts from here.


<TextView

android:id="@+id/textView5"

android:layout_row="1"

android:layout_column="0"

android:text="E"
```

```
android:textAppearance="?android:attr/textAppearanceLarge"

android:padding="30dp"/>


<TextView

android:id="@+id/textView6"

android:layout_row="1"

android:layout_column="1"

android:text="F"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView

android:id="@+id/textView7"

android:layout_row="1"

android:layout_column="2"

android:text="G"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView
```

```
android:id="@+id/textView8"

android:layout_row="1"

android:layout_column="3"

android:text="H"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


//3rd row starts from here.


<TextView

android:id="@+id/textView9"

android:layout_row="2"

android:layout_column="0"

android:text="I"

android:textAppearance="?android:attr/textAppearanceLarge"

android:padding="30dp"/>


<TextView

android:id="@+id/textView10"

android:layout_row="2"
```

```
android:layout_column="1"

android:text="J"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView

android:id="@+id/textView11"

android:layout_row="2"

android:layout_column="2"

android:text="K"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />


<TextView

android:id="@+id/textView12"

android:layout_row="2"

android:layout_column="3"

android:text="L"

android:padding="30dp"

android:textAppearance="?android:attr/textAppearanceLarge" />
```

</GridLayout>

Constraint Layout:

It makes handling complex screen designs easier. It also improves the performance of complex layouts.

With ConstraintLayout, even complex screen designs with nested layouts can be fast.

ConstraintLayout provides a level of flexibility that allows many of the features of older layouts to be achieved with a single layout instance. Before, you needed to nest multiple layouts.

This has the benefit of avoiding many problems inherent in nesting layouts. It allows designing so-called *flat* or *shallow* layout hierarchies. This leads to less complex layouts and improved user interface rendering performance at runtime.

ConstraintLayout is also implemented with an eye toward addressing the wide range of Android device screen sizes available on the market today.

The flexibility of ConstraintLayout makes it easier for user interfaces to be designed that respond and adapt to the device on which the app is running.

Frame Layout:

Frame Layout is one of the layouts which helps us to create a more complex design easily. When we are required to create a design where the components are on top of each other, we use the FrameLayout.

To define which component will be on top, we put it in the end. *For example*, if we want some text over an image, then we will put the TextView in the end.

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent">
```

```
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/mindorkslogo"
        android:text="Button" />
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="MindOrks" />
</FrameLayout>
```

Scroll Layout (ScrollView):

A view group that allows the view hierarchy placed within it to be scrolled. Scroll view may have only one direct child placed within it. To add multiple views within the scroll view, make the direct child you add a view group, for example LinearLayout, and place additional views within that LinearLayout.

Scroll view supports vertical scrolling only. For horizontal scrolling, use HorizontalScrollView instead.

Never add a RecyclerView or ListView to a scroll view. Doing so results in poor user interface performance and a poor user experience.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fillViewport="false">
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView android:id="@+id/loginscrn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="80dp"
        android:text="ScrollView"
        android:textSize="25dp"
        android:textStyle="bold"
        android:layout_gravity="center"/>
```

```
<TextView android:id="@+id/fstTxt"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="20dp"

android:text="Welcome to Tutlane"

android:layout_gravity="center"/>

<Button android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_gravity="center"

android:layout_marginTop="60dp"

android:text="Button One" />

<Button android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_gravity="center"

android:layout_marginTop="60dp"

android:text="Button Two" />

<Button android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_gravity="center"

android:layout_marginTop="60dp"
```

```
android:text="Button Three" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="60dp"
android:text="Button Four" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="60dp"
android:text="Button Five" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="60dp"
  android:text="Button Six" />
<Button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="60dp"
```

```
        android:text="Button Seven" />

        <Button android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_gravity="center"

        android:layout_marginTop="60dp"

        android:text="Button Eight" />

        <Button android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_gravity="center"

        android:layout_marginTop="60dp"

        android:text="Button Nine" />

</LinearLayout>

</ScrollView>
```

Unit 3. Interactivity Tools

- **3.1. Intents and Filters**
- **3.2. Adapters**
- **3.3. Dialogs**
- **3.4. Menus**
- **3.5. Notifications**

3.1. Intents and Filters:

Introduction to Intent

=> An Intent is a messaging object you can use to request an action from another app component.

Let's look upon the informal way of defining Intents. You can think of Intents as a messaging service that is used to communicate between various components of the Android application. For example, if you want to send some message from Delhi to Mumbai using the Post Office facility then you can do so by buying an Envelope and then pass the message in the Envelope and send the message to the desired location.

Types of Intent(Implicit and Explicit Intent)

=> There are two types of Intents:

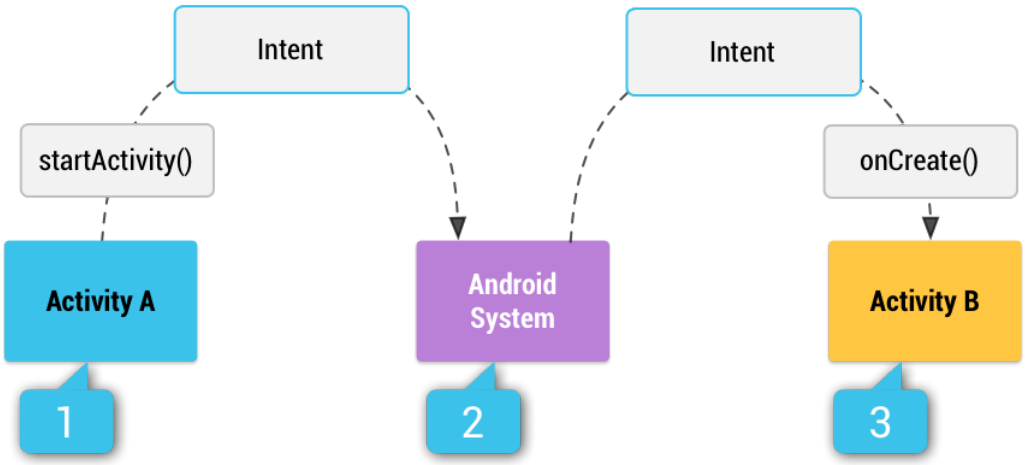
1. **Explicit Intents:** If you want communication between the components of your application only then you can use the Explicit Intents. Explicit Intents are used to communicate with a particular component of the same application. For example, if you want to launch an Activity by clicking some button on the present Activity then you can specify the fully-qualified address of the desired Activity to launch that Activity. Since this approach requires a fully-qualified address, you can use this approach in your own application i.e. you can use Explicit Intents to have communication in your own application.

   Example: Shift from one activity to another activity

2. **Implicit Intents:** Here, you don't need to specify the fully-qualified address. All you need to do is just specify the action that is to be performed by an Intent. By using the Implicit Intents you can communicate between various applications present in the mobile device.

   Example: Start camera intent on button click or share string value.

Some More Information about Intent:



In the above figure, you can find that if you call an Explicit Intent i.e. you are passing the fully-qualified address of the action to be performed then, the system will directly launch that activity or will start performing the desired activity.

If you call the Implicit Intent then, the Android System will search for all the available components that can be used to start that activity. This process is done by comparing the contents of the intent with the content present in the *intent-filters* declared in the *AndroidManifest.xml* file. If there is only one intent-filter that is compatible with the content of the intent then the Android system will start the desired component. But if there are a number of *intent-filters* that are compatible with the content of the Intent then the Android System will show you a list of applications that can be used to perform that particular action. For example, if you want to share some image from the Gallery, then you will get a number of choices like WhatsApp, Facebook, Instagram, Shareit, Gmail and many more image sharing applications. Now, you can choose any of the available choices.

Intent Filters are expressions that are used to specify the type of components or actions that can be received by the application and this Intent Filter is declared

in the *AndroiManifest.xml* file. If you are not declaring any Intent Filters, then you have to use the Explicit Intents only.

Information present in Intent

So, we have seen that an Activity or an action can be called by using Explicit Intents or by using some Implicit Intents. But the question that arises here is that how does the Android System come to know that a particular Activity or Action is to be called? This is done by reading the information that is present in the Intent. The Android System reads the information present in the Intent and based on this information, the Android System decides which Activity is to be launched. So, some of the basic information that an Intent contains are:

1. **Action:** An action is a string that specifies the action to be performed by a particular Activity. For example, you can use the *ACTION_VIEW* with *startActivity()* when your application contains some information like images that can be shown to the user. Another action that can be performed is *ACTION_SEND*, which is used to share some data with another application like in Email applications.

2. **Data:** While creating an Intent, you can pass the data and the type of data on which the action is to be performed by the Android System with the help of Intents. The URI object is used to reference the data that will be used to perform some action on it. For example, if you want to edit some data then you have to pass the URI of the data in your *ACTION_EDIT* action.

3. **Category:** Category is used in case of Explicit Intents where you need to specify the type of application that will be used to perform a particular

action. For example, if you want to send some data then only data sending applications should be made available for choice to the users. You can specify a category with the help of *addCategory()*. Any number of categories can be added to the Intent.

4. **Component Name:** The component name is the name of the component that is to be started. You can set the component name by using *setComponent()* or *setClass()* or with the Intent Constructor.

5. **Extras:** You can add extra data to an Intent in the form of key-value pairs and this extra information can be passed from one Activity to the other. *putExtra()* is used to add some extra data to the Intents and this method accepts two parameters i.e. the key and its corresponding value.

Intent Filter

- Implicit intent uses the intent filter to serve the user request.
- The intent filter specifies the types of intents that an activity, service, or broadcast receiver can respond.
- Intent filters are declared in the Android manifest file.
- Intent filter must contain <action>

Most of the intent filter are describe by its

1. <action>,
2. <category> and
3. <data>.

1. <action>

Adds an action to an intent filter. An <intent-filter> element must contain one or more <action> elements. If there are no <action> elements in an intent filter, the filter doesn't accept any Intent objects.

Examples of common action:

- **ACTION_VIEW:** Use this action in intent with startActivity() when you have some information that activity can show to the user like showing an image in a gallery app or an address to view in a map app
- **ACTION_SEND:** You should use this in intent with startActivity() when you have some data that the user can share through another app, such as an email app or social sharing app.

2. <category>

Adds a category name to an intent filter. A string containing additional information about the kind of component that should handle the intent.

Example of common categories:

- **CATEGORY_BROWSABLE:** The target activity allows itself to be started by a web browser to display data referenced by a link.

3. <data>

Adds a data specification to an intent filter. The specification can be just a data type, just a URI, or both a data type and a URI.

Examples:

```
<!--LAUNCHER INTENT FILTER-->
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

```
<!--SEND INTENT FILTER-->
<intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
</intent-filter>
```

```
<!--SEND INTENT FILTER-->
<intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
</intent-filter>
```

```
<!--VIEW INTENT FILTER-->
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.BROWSABLE"/>
    <data android:scheme="http"/>
</intent-filter>
```

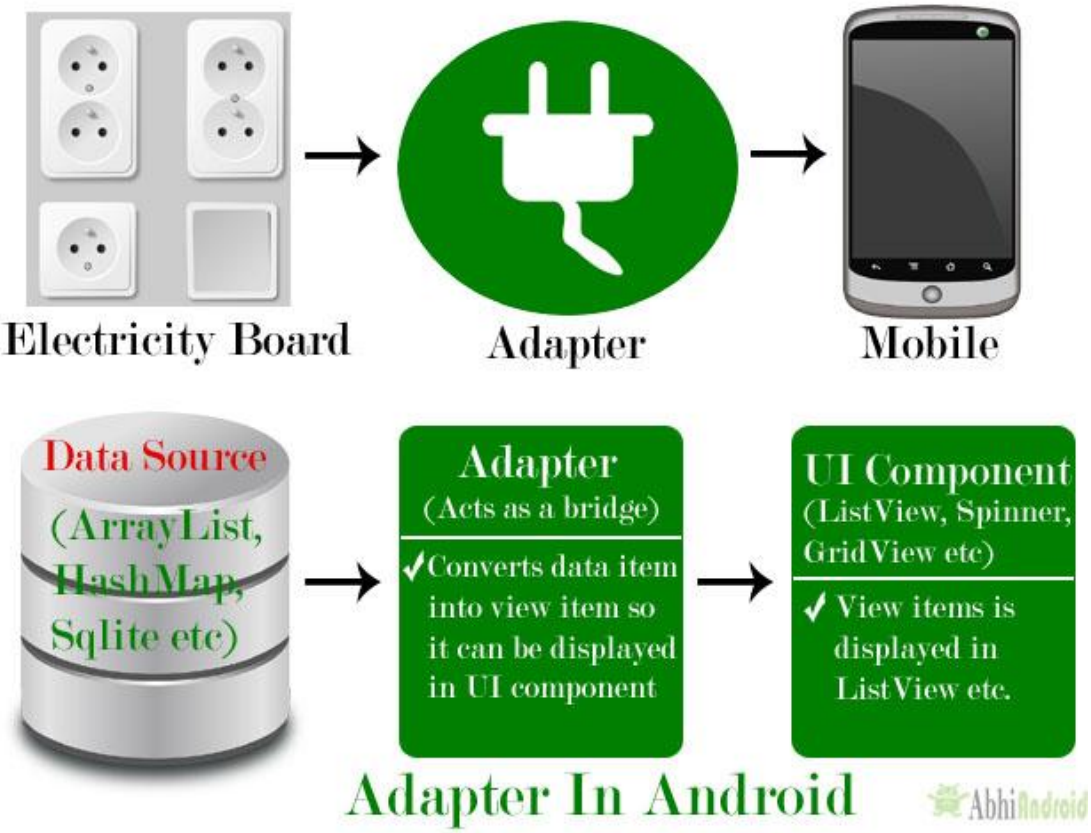
Demo: https://www.geeksforgeeks.org/intent-filter-in-android-with-demo-app/

3.2. Adapters:

In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc. For more customization in Views we uses the base adapter or custom adapters.

To fill data in a list or a grid we need to implement Adapter. Adapters acts like a bridge between UI component and data source. Here data source is the source from where we get the data and UI components are list or grid items in which we want to display that data.

Below is a conceptual diagram of Adapter:



Adapters In Android:

There are the some commonly used Adapter in Android used to fill the data in the UI components.

1. **BaseAdapter** – It is parent adapter for all other adapters
2. **ArrayAdapter** – It is used whenever we have a list of single items which is backed by an array
3. **Custom ArrayAdapter** – It is used whenever we need to display a custom list
4. **SimpleAdapter** – It is an easy adapter to map static data to views defined in your XML file
5. **Custom SimpleAdapter** – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

Now we describe each Adapters one by one in detail:

1. BaseAdapter In Android:

BaseAdapter is a common base class of a general implementation of an Adapter that can be used in ListView, GridView, Spinner etc. Whenever we need a customized list in a ListView or customized grids in a GridView we create our own adapter and extend base adapter in that. Base Adapter can be extended to create a custom Adapter for displaying a custom list item. ArrayAdapter is also an implementation of BaseAdapter.

Custom Adapter code which extends the BaseAdapter in that:

```
public class CustomAdapter extends BaseAdapter {

@Override
public int getCount() {
return 0;
}
```

```
@Override
public Object getItem(int i) {
return null;
}

@Override
public long getItemId(int i) {
return 0;
}

@Override
public View getView(int i, View view, ViewGroup viewGroup) {

return null;
}
```

In above code snippet we see the overrided functions of BaseAdapter which are used to set the data in a list, grid or a spinner.

2. ArrayAdapter In Android:

Whenever we have a list of single items which is backed by an Array, we can use ArrayAdapter. For instance, list of phone contacts, countries or names.

Here is how android ArrayAdapter looks ::

```
ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)
```

3. Custom ArrayAdapter In Android:

ArrayAdapter is also an implementation of BaseAdapter, so if we want more customization then we can create a custom adapter and extend ArrayAdapter in

that. Since array adapter is an implementation of BaseAdapter, so we can override all the function's of BaseAdapter in our custom adapter.

Below Custom adapter class MyAdapter extends ArrayAdapter in that:

```
public class MyAdapter extends ArrayAdapter {

public MyAdapter(Context context, int resource, int textViewResourceId, List
objects) {
super(context, resource, textViewResourceId, objects);
}
@Override
public int getCount() {
return super.getCount();
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
return super.getView(position, convertView, parent);
}
}
```

4. SimpleAdapter In Android:

In Android SimpleAdapter is an easy Adapter to map static data to views defined in an XML file(layout). In Android we can specify the data backing to a list as an ArrayList of Maps(i.e. hashmap or other). Each entry in a ArrayList is corresponding to one row of a list.

The Map contains the data for each row. Here we also specify an XML file(custom list items file) that defines the views which is used to display the row, and a mapping from keys in the Map to specific views.

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item's backed by an Array. So if we need more customization in a ListView or a GridView we need to implement simple adapter.

SimpleAdapter code in Android:

```
SimpleAdapter (Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to)
```

5. Custom SimpleAdapter In Android:

Whenever we have to create a custom list we need to implement custom adapter. As we discuss earlier ArrayAdapter is used when we have a list of single item's backed by an Array. So if we need customization in a ListView or a GridView we need to implement simple Adapter but when we need more customization in list or grid items where we have many view's in a list item and then we have to perform any event like click or any other event to a particular view then we need to implement a custom adapter who fulfills our requirement's and quite easy to be implemented.

BaseAdapter is the parent adapter for all other adapters so if we extends a SimpleAdapter then we can also override the base adapter's function in that class.

Important Note: We can't perform events like click and other event on child item of a list or grid but if we have some requirements to do that then we can create our own custom adapter and extends the simple adapter in that.

Custom Adapter extends SimpleAdapter in that:

```
public class CustomAdapter extends SimpleAdapter {
public CustomAdapter(Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to) {
super(context, data, resource, from, to);
```

```
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
return super.getView(position, convertView, parent);

}

@Override
public int getCount() {
return super.getCount();
}
}
```

3.3. Dialogs:

1) Alert Dialog:

Android AlertDialog can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

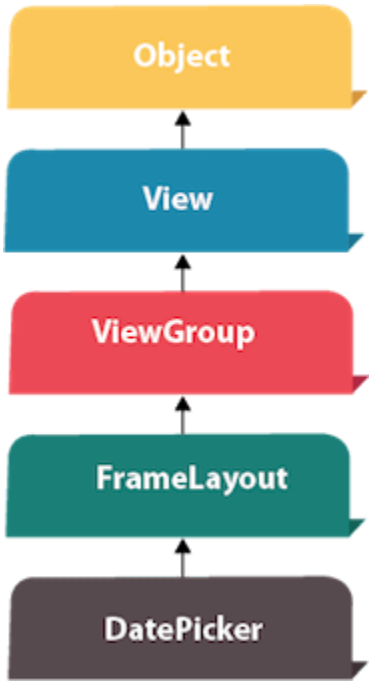
Android AlertDialog is the subclass of Dialog class.


```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
    alertDialogBuilder.setMessage("Are you sure,
      You wanted to make decision");
    alertDialogBuilder.setPositiveButton("yes",
      new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface arg0, int arg1) {
        Toast.makeText(MainActivity.this,"You clicked yes
          button",Toast.LENGTH_LONG).show();
      }
    });

    alertDialogBuilder.setNegativeButton("No",new
DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
        finish();
      }
    });

    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
```

2) Date Picker Dialog:

Android DatePicker is a widget to select dates. It allows you to select date by day, month and year. Like DatePicker, android also provides TimePicker to select time.

The android.widget.DatePicker is the subclass of FrameLayout class.




```
DatePickerDialog datePickerDialog = new DatePickerDialog(MyDatePicker.this,
    new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker datePicker, int year, int month, int day) {
            String formattedDate = getFormattedDate(day, month, year);
```

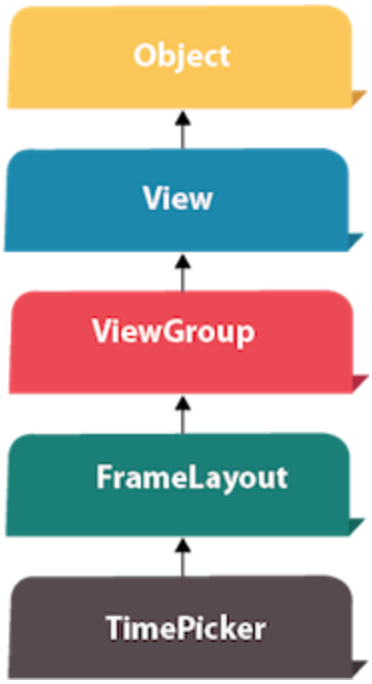
```
            textViewCurrentDateValue.setText(formattedDate);
        }
    }, 0, 0, 0);
datePickerDialog.show();
```

3) Time Picker Dialog:

In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode. The displayed time consists of hours, minutes and clock format. If we need to show this view as a Dialog then we have to use a TimePickerDialog class.

It allows you to select time by hour and minute. You cannot select time by seconds.

The android.widget.TimePicker is the subclass of FrameLayout class.




```
<TimePicker
  android:id="@+id/timePicker1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content" />
```

4) Custom Dialog:

The custom dialog uses DIALOG to create custom alerts in android studio. Dialog displays a small window i.e a popup which draws the user attention over the activity before they continue moving forward. The dialog appears over the current window and displays the content defined in it.

AlertDialog Vs Custom AlertDialog:

The Alert Dialog and Custom Alert Dialog both prompt a small window to make decisions. The AlertDialog makes use of the defined components or methods like setIcon, setTitle, setmessage etc but with Custom AlertDialog we can have the dialog customized and can define the layout of dialog as required. You can read more about it in the alert dialog tutorial.

Example:

```
LayoutInflater layoutInflater = LayoutInflater.from(this);

final View customLayout= layoutInflater.inflate(R.layout.forget_password_custom_layout, null);

final AlertDialog alertDialog = new AlertDialog.Builder(this).create();

alertDialog.setView(forgetPasswordView);

customLayout.findViewById(R.id.buttonReset).setOnClickListener(new View.OnClickListener() {

  public void onClick(View v) {

Toast.makeText(MyCustomDialog.this, "Custom Toast Msg", Toast.LENGTH_SHORT).show();

    alertDialog.dismiss();

  }
```

});

alertDialog.show();

5) Progress Bar:

In Android, ProgressBar is used to display the status of work being done like analyzing status of work or downloading a file etc. In Android, by default a progress bar will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal. It mainly uses the “android.widget.ProgressBar” class.

Important Note: A progress bar can also be made indeterminate. In this mode a progress bar shows a cyclic animation without an indication of progress. This mode is used in application when we don’t know the amount of work to be done.

To add a progress bar to a layout (xml) file, you can use the <ProgressBar> element. By default, a progress bar is a spinning wheel (an indeterminate indicator). To change to a horizontal progress bar, apply the progress bar’s horizontal style.


```
<ProgressBar

android:id="@+id/simpleProgressBar"
```

```
android:layout_width="wrap_content"

android:layout_height="wrap_content" />
```

3.4. Menus:

1) Option Menu:

A Menu is a crucial part of the User Interface that handles frequent functionalities of the application. The menu helps us provide a user-friendly interface that handles a lot of actions.

You can declare items for the options menu from either your Activity subclass or a Fragment subclass.

If both your activity and fragment(s) declare items for the options menu, they are combined in the UI. The activity's items appear first, followed by those of each fragment in the order in which each fragment is added to the activity.

The options menu is the primary collection of menu items for an activity. It's where you should place actions that have an overall impact on the app, such as Search, Compose Email and Settings.

2) Context Menu:

A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

Android context menu appears when the user presses a long click on the element. It is also known as a floating menu.

It affects the selected content while doing action on it.

It doesn't support item shortcuts and icons.

In Android, the context menu is like a floating menu and arises when the user has long pressed or clicks on an item and is beneficial for implementing functions that define the specific content or reference frame effect. The Android context menu is similar to the right-click menu in Windows or Linux. In the Android system, the context menu provides actions that change a specific element or context frame in the user interface and one can provide a context menu for any view. The context menu will not support any object shortcuts and object icons.

3) Popup Menu:

Android Popup Menu displays a list of items in a vertical list which presents to the view that invoked the menu and useful to provide an overflow of actions that related to specific content.

In android, Popup Menu displays a list of items in a modal popup window that is anchored to the view. The popup menu will appear below the view if there is a room or above the view in case there is no space and it will be closed automatically when we touch outside of the popup.

The android Popup Menu provides an overflow style menu for actions that are related to specific content.
The popup menu won’t support any item shortcuts and item icons.

In android, the Popup menu is available with API level 11 (Android 3.0) and higher versions. If you are using Android 3.0 +, the Popup Menu won't support any item shortcuts and item icons in the menu.

3.5. Notifications:

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

To see the details of the notification, you will have to select the icon which will display the notification drawer having details about the notification. While working with an emulator with a virtual device, you will have to click and drag down the status bar to expand it.

- setSmallIcon(): It sets the icon of notification.
- setContentTitle(): It is used to set the title of notification.
- setContentText(): It is used to set the text message.
- setAutoCancel(): It sets the cancelable property of notification.
- setPriority(): It sets the priority of notification.

Unit 4. Interaction with Database

4.1. Introduction to Database (SQLite and Firebase)

4.2. Cursors and content values

4.3. CURD Operations

4.1. Introduction to Database (SQLite and Firebase):

SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.

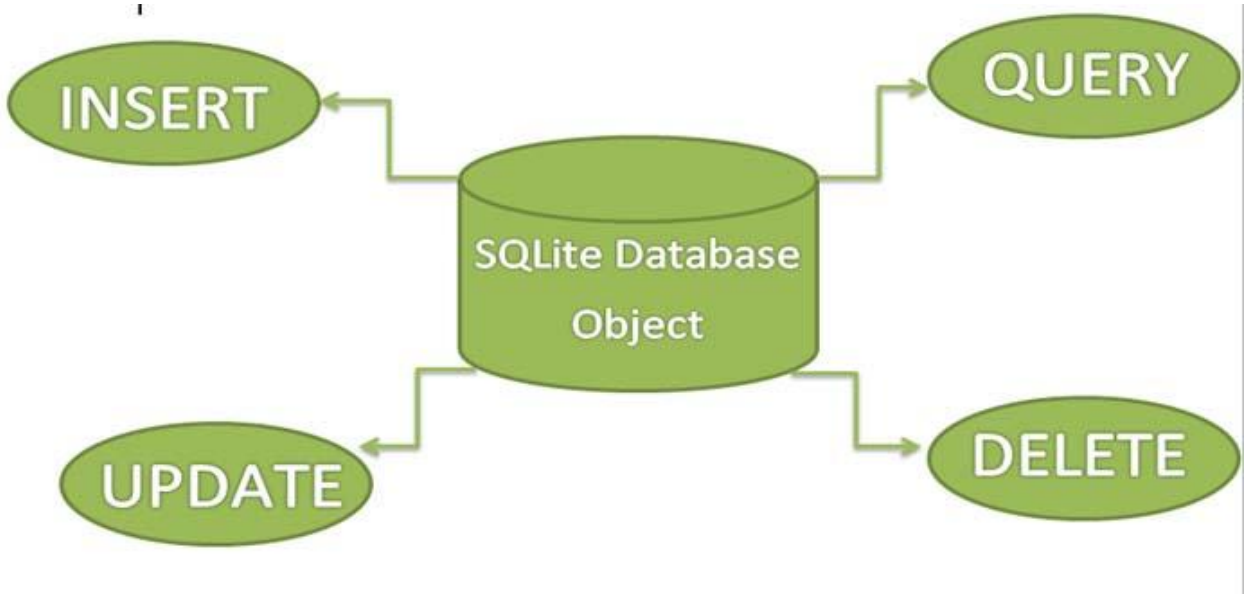
It is embedded in android by default. So, there is no need to perform any database setup or administration task.

SQLiteOpenHelper class provides the functionality to use the SQLite database.

SQLiteOpenHelper:

The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management. For performing any database operation, you have to provide the implementation of onCreate() and onUpgrade() methods of SQLiteOpenHelper class.

SQLite is a Structure query base database, open source, light weight, no network access and standalone database. It supports embedded relational database features.



Whenever an application needs to store large amounts of data then using sqlite is more preferable than other repository systems like SharedPreferences or saving data in files.

Android has built in SQLite database implementation. It is available locally over the device(mobile & tablet) and contains data in text format. It carries light weight data and is suitable with many languages. So, it doesn't require any administration or setup procedure of the database.

Creating And Updating Database In Android

For creating, updating and other operations you need to create a subclass of SQLiteOpenHelper class. SQLiteOpenHelper is a helper class to manage database creation and version management. It provides two methods onCreate(SQLiteDatabase db), onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion).

The SQLiteOpenHelper is responsible for opening the database if it exists, creating database if it does not exist and upgrading if required. The

SQLiteOpenHelper only requires the DATABASE_NAME to create a database. After extending SQLiteOpenHelper you will need to implement its methods onCreate, onUpgrade and constructor.

onCreate(SQLiteDatabase sqLiteDatabase) method is called only once throughout the application lifecycle. It will be called whenever there is a first call to getReadableDatabase() or getWritableDatabase() functions available in super SQLiteOpenHelper class. So SQLiteOpenHelper class calls the onCreate() method after creating the database and instantiating the SQLiteDatabase object. Database name is passed in constructor call.

onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) is only called whenever there is an update in the existing version. So to update a version we have to increment the value of version variable passed in the superclass constructor.

In the onUpgrade method we can write queries to perform whatever action is required. In most examples you will see that existing table(s) are being dropped and again the onCreate() method is being called to create tables again. But it's not mandatory to do so and it all depends upon your requirements.

We have to change the database version if we have added a new row in the database table. If we have a requirement that we don't want to lose existing data in the table then we can write an alter table query in the onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) method.

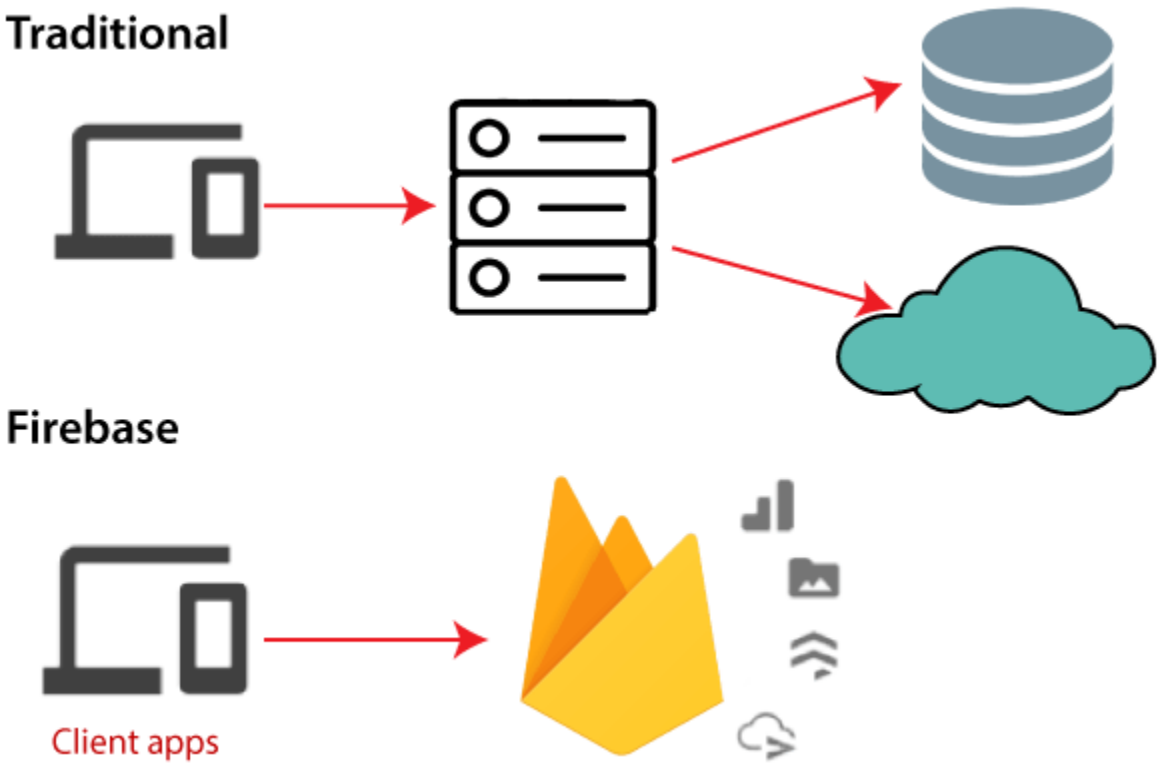
Firebase:

Firebase tutorial is designed for both beginners and professionals. Our tutorial provides all the basic and advanced services knowledge, such as Real-time Database, Cloud Messaging, Hosting and Crash Reporting, etc.

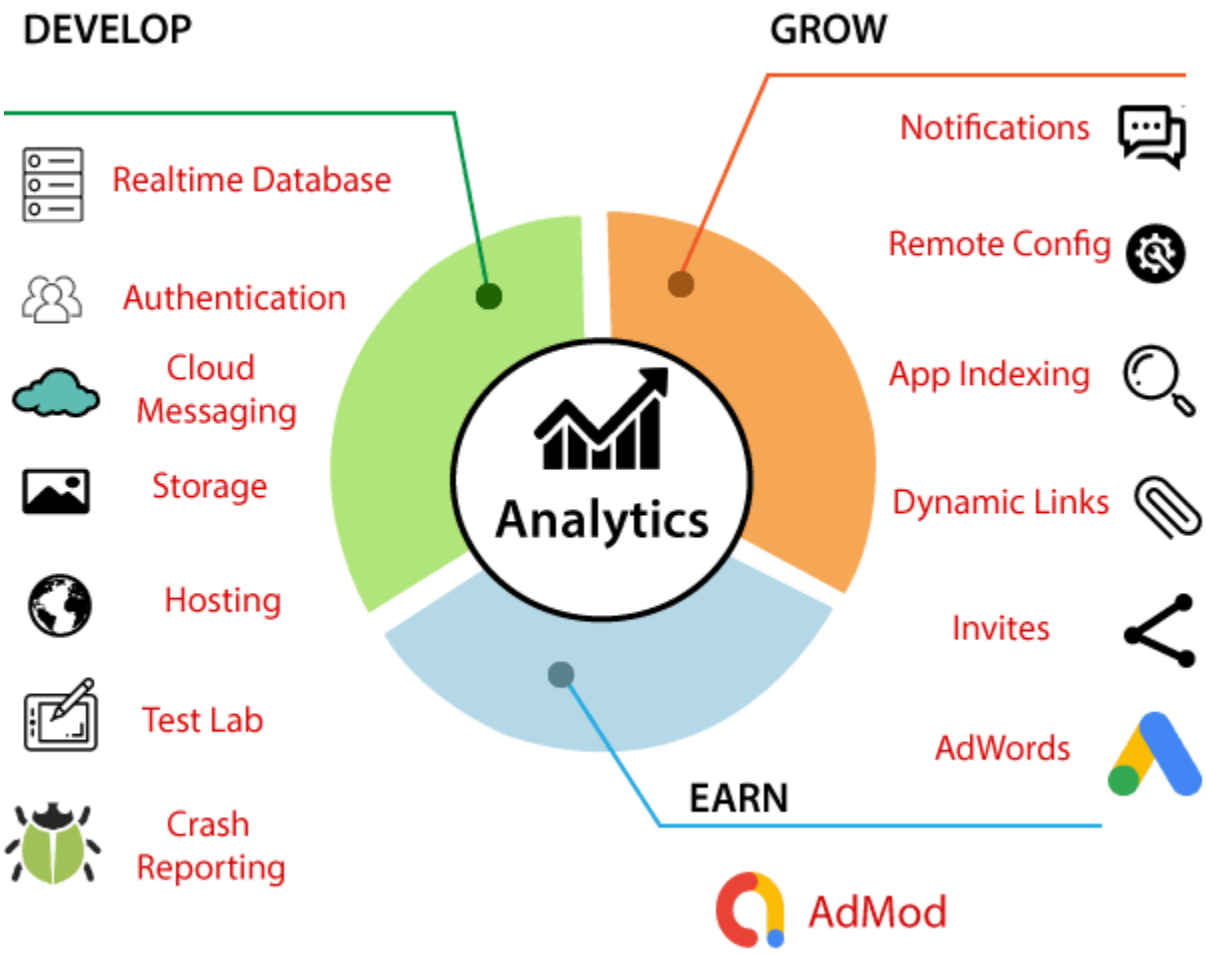
Firebase is a Backend-as-a-Service, and it is a real-time database which is basically designed for mobile applications. This tutorial is designed in such a way that we can easily understand or can perform the service of Firebase in a very efficient way.

In the era of rapid prototyping, we can get bright ideas, but sometimes they are not applicable if they take too much work. Often, the back-end is the limiting factor - many considerations never apply to server-side coding due to lack of knowledge or time.

Firebase is a Backend-as-a-Service(BaaS) which started as a YC11 startup. It grew up into a next-generation app-development platform on Google Cloud Platform. Firebase (a NoSQLjSON database) is a real-time database that allows storing a list of objects in the form of a tree. We can synchronize data between different devices.



Google Firebase is Google-backed application development software which allows developers to develop Android, IOS, and Web apps. For reporting and fixing app crashes, tracking analytics, creating marketing and product experiments, firebase provides several tools.



Firebase has three main services, i.e., a real-time database, user authentication, and hosting. We can use these services with the help of the Firebase iOS SDK to create apps without writing any server code.

Unit 5. Web Services and Web View

5.1. Introduction to web services:

A web service is a standard for exchanging information between different types of applications irrespective of language and platform. For example, an android application can interact with java or .net application using web services.

What are Android Web Services?

Android Web Services is a standardised system that helps various applications and systems to communicate with each other. While communicating, they can exchange information and also share some services among themselves. Android web services can run on the internet or private local networks depending on the

requirements. Android Web Services are pretty helpful in establishing connections and ensuring security while sharing data in the network.

Below is a ubiquitous example of android web services, which would help you understand the android web services. Suppose your application can seek restaurant data from the server and then display it in your application. The application can then send back the desired list of items to the server, and then the admins can fulfil your order.

So, you can notice that data has to flow from server to application and from application back to the server. In such scenarios, Android Web Services play a significant role in establishing and deciding the protocols for communication.

How do Web Servers work?

When we consider web services, there are two essential parts of a web service known as client and server.

Client: The client is the user or the requesting application that requests data or information from a server.

Server: Server is like an admin who responds to client's requests. Servers can handle more than one client and decide whether to fulfil or decline client requests. The server is the place where our web service is hosted globally or locally.

Components of Android Web Services

Now, let's see some of the components present in the web server and understand their role.

1. Publisher – Publisher provides web services to clients and is also known as a service provider.

2. Subscriber – The subscriber is the user or the application that requests services from the publisher.

3. Broker – Usually, the Subscriber is unknown about the publisher and needs something to guide the location of the web service. So, the broker is the application that helps the subscriber to identify the web service. The broker gives the subscriber access to UDDI(User descriptive, discovery, and integration).

Now, let's see the roles and operations each of them carries out.

- **Publish** – Publish means creating the web service and describing its location to the broker for its easy identification by subscribers.
- **Subscribe** – Subscribe means that the subscriber locates the web service with the help of the broker.
- **Bind** – After the subscriber successfully fetches the location, the subscriber binds itself with the web service to exchange information.

Characteristics of Web Services in Android

I hope until now you are clear with what web services are and the components involved in them. Now, it's time for us to look at some of the web services in android.

a. Web Services are XML-Based – Both client and server use XML as their communication language. In other words, the client requests in XML and receives a response, which is XML.

b. Web services are not tied to one specific operating system or programming language. For example, a Java-based application can communicate with a Perl based application.

c. Web Services are available on both the internet or on the local network.

d. Web Services are not tightly coupled. In other words, the client-side web service and the provider side web service are not directly tied.

e. Web Services can be either synchronous or asynchronous. By being synchronous, the clients can directly perform functionalities without establishing a connection. By being asynchronous, the client first needs to establish a connection and then perform the functionalities.

f. Web Services allow you to share multiple files, including documents and complex ones.

Types of Web Services in Android

There are four types of Web Services available in android and are listed below:

1. XML-RPC

XML-RPC, popularly known as Remote Procedure Calls, are used to exchange information among large devices. Every call is encoded using XML, and HTTP is used for its transmission.

2. UDDI

UDDI is an acronym for Universal Descriptive, Discovery, and Integration. It is an XML-based standard that is used to describe, publish, and discover new web services.

3. SOAP

SOAP refers to the Simple Object Access Protocol and is an XML-based web service protocol for exchanging data or documents over HTTP (Hypertext

transfer protocol) or SMTP (Simple Message Transfer Protocol). It allows separate processes on different platforms to communicate with one another.

4. REST

REST(REpresentational State Transfer) is an architectural pattern that allows multiple web service-based systems to interact and communicate efficiently. RESTful systems(the system in compliance with REST service) are distinguished by their statelessness and separation of client and server concerns.

Advantages of Android Web Services

1. Web services make it possible for various applications to communicate with one another.

2. Reusability is one of the essential benefits of using web services.

3. Web services allow for more efficient communication within and across applications and organisations.

4. They communicate across various apps using a high-quality industry-standard protocol.

5. They employ SOAP over HTTP to enable web services via a low-cost internet connection.

6. Web Services are made available using conventional internet protocols.

7. They enable us to make the functionalities of current programmes available to the public via the internet.

Limitations of Android Web Services

Even though web services are pretty beneficial still there are certain demerits of Web Service, which are listed below:

1. They don’t take advantage of new Web advancements.

2. Web services can’t be accessed using a browser.

3. Web services utilise the HTTP protocol, which is unreliable and unsafe.

Summary

Through this article, you came across web services and understood what it means. You came across the working and the components of web services. You saw what is meant by client and server and also saw the mode of communication they follow. Then you came across the characteristics of web services in android.

Moving further, your XML-RPC and also saw the other types of web services present in android. Finally, you came across the advantages and demerits of having a web service.

Example 2:

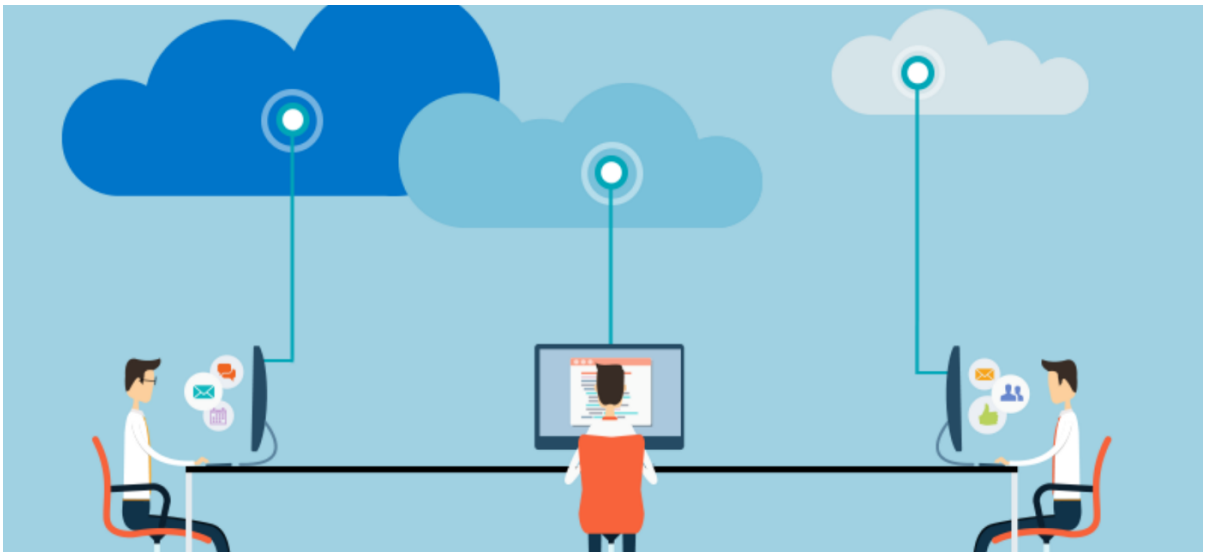
There are two types of applications in the market first Static application and second Dynamic application. Static application has fixed and unchangeable content whereas Dynamic application change their content based on web services data. The static application can not take or process user data but the Dynamic application has this type of facility. These facilities are generally provided by Web services(also called hosting services). Web Services provides its Server to store or process the data.

Introduction

It is a server-side platform that receives and process request from different types of application irrespective of language and platform.

Web services are an essential component when dealing with an application that is dynamic because most of the applications running these days use web

services. Web services provide an application to host their data on the cloud (or servers).



All web services are API, but all APIs are not web services. Most people get confused about the minor difference between web services and API. Let's understand this in very simple terms. If API is running on any web server then it is a web service or provides any kind of service over the web hence it is called a web service whereas web services can be of many more types like providing SAAS(software as a service), IAAS(infrastructure as a service), PAAS(Platform as a service) these all will provide services but API can also be present of offline like if we have to connect an application to another application within devices so there is no need of a server to be involved in that case.

So now let's have a look at API

API(Application Programming Interface)

An API act as an interface between two different application so they can communicate with each other.

- API is a messenger which delivers the request to the provider and responds.

- API can communicate over a network and also without a network.

Some popular APIs are as follows,

- Google Map API
- Youtube API
- Facebook Login API
- Gmail login API

How API helps Developers

An API helps developers in many ways some of them are as follows-

- API helps developers to use the third-party application
- Do not reinvent the wheel over and over.
- Efficient product in a short period.

Types of Web Services

There are mainly two types of web services SOAP and REST. Let's understand each one in brief.

SOAP(Simple Object Access Protocol)

SOAP or Simple Object Access Protocol is a technique to send an XML request over the internet using HTTP protocol and return an XML response. Here envelope is used to send data hence its data transfer is secure. It was used in previous applications and it is a very secure protocol.

REST(Representational State Transfer Protocol)

Any Web Service that is defined on the Principle of REST uses HTTP verbs of getting, POST, PUT, and DELETE.

REST allocates resources on URL and acts. it allows multiple web service-based systems to interact and communicate with them.

SOAP	REST
1. This is a function-based protocol. 2. It only uses XML. 3. It can't be cached. 4. It has a strict communication 5. Build-in ACID compliance. 6. used in banking applications where security is prior.	1. This is Database based Protocol. 2. This Permit HTTP,plain text,XML,JSON 3. It can be Cached. 4. It has an easy communication

	5. Lack of ACID compliance 6. It is advance and simple

REST API mostly uses JSON format so let us see what is JSON.

JSON(JavaScript object Annotation)

It is not a programming language, it s a data interchange format.JSON is mostly used to get and post the data in web services. it is the mostly used formate in today's application development.

- It is a file in which data is written in text formate.
- it is easy and simple to understand

Data types of JSON

1. String - "C-sharp Corner"
2. numbers - 1,2,-1,-2
3. Boolean - true,false
4. Array - ["java","python","HTML"] or [2,6,-1]
5. object - { "key1" : "value1", "key2": "value2"}

Example of JSON formate,

{ "name": "ravi", "age": 23, "email": "ravi@gmail.com", "programming": ["java", "C++", "python"], "experience": [{ "company Name": "Company1", "years": 2, "location": "noida" } { "company Name": "company2", "years": 2, "location": "noida" }] }

Conclusion

In this article we have learned what is web Services in android and what is API, we have covered the difference between web services and API and after that, we have learned types of Web services SOAP and REST. After that, we have seen the table of REST and SOAP differences, and at last we have seen the JSON format and a simple Connection API.

Web View:

If you want to deliver a web application (or just a web page) as a part of a client application, you can do it using WebView. The WebView class is an extension of Android's View class that allows you to display web pages as a part of your activity layout. It does *not* include any features of a fully developed web browser, such as navigation controls or an address bar. All that WebView does, by default, is show a web page.

A common scenario in which using WebView is helpful is when you want to provide information in your app that you might need to update, such as an end-user agreement or a user guide. Within your Android app, you can create an Activity that contains a WebView, then use that to display your document that's hosted online.

Another scenario in which WebView can help is if your app provides data to the user that always requires an Internet connection to retrieve data, such as email. In this case, you might find that it's easier to build a WebView in your Android app that shows a web page with all the user data, rather than performing a network request, then parsing the data and rendering it in an Android layout. Instead, you can design a web page that's tailored for Android devices and then implement a WebView in your Android app that loads the web page.

Ex:

```
<WebView
    android:id="@+id/webview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

To load a web page in the WebView, use loadUrl().

For example:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```