Android Application Development for beginners- A Review

Dipika Jain
M.Tech Scholar
dipikajain24.12@gmail.com

Abstract: **Android is basically an operating system for smartphones. But we find now integrated into PDAs, touch pads or televisions, even cars (trip computer) or net books. The OS was created by the start-up of the same name, which is owned by Google since 2005. Mobile users demand more functionalities and innovative mobile platform in order to customize their handset according to their personal desires. The mobile community supporters, whether they are application developers, service provider, or mobile device manufacturers, all are trying hard to fulfil the growing demands of customers. Mobile service providers want to offer value-added services to their customers in an organized and meaningful manner; Application developers want the freedom to develop more powerful, affordable, and innovative handset applications; similarly mobile device manufacturers want to produce reliable and affordable mobile devices**

Keywords: Android SDK, DVM, Android Architecture.

1. Introduction

Android Introduction in the development of Android application. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. Developers mostly choose the popular Eclipse Integrated Development Environments for development.
Here in this paper we are going to discuss about the features of android, the basic architecture of Android and the Life Cycle of the Android in three sections consequently.

1. Features of Android

The open source licensing of Android, gives freedom to developers to develop applications without concerning the royalty and licensing cost. The features of Android are as follows:

- Android SDK- Android Software Development Kit
- DVM – Dalvik Virtual Machine
- Graphics Support
- SQLite
- Connectivity
- Media Support

The features are described below in this section.

2.1 Android SDK- The Android SDK includes an emulator, some tools for performance profiling and debugging. Eclipse IDE is natural choice for Android developers. Android Development tool (ADT) is a plugin use to enhance and boost the performance of Eclipse IDE. It provides faster and easier way of creation and debugging of Android application. Note that further plugins are also available to support other IDEs such as IntelliJ and NetBeans for the Android developers.
2.2 Dalvik virtual machine - It is specifically designed for Android platform and optimized for mobile devices, where resource constrains is an issue (like low memory, small size, and lower processing power). Dalvik is register based virtual machine and its interpreter is optimized for faster execution. Dalvik is capable of executing programs written in Java. It does not understand the java code directly, rather a dx tool is use to convert java code into byte code (which is then executed by Dalvik). The purpose of conversion java code into byte code is to optimize the code to be easily compiled over the limited resourced mobile device. Android support the execution of multiple instances of Dalvik VM simultaneously.
2.3 Graphics support - Android have support for both 2D and high performance 3D graphics (the OpenGL API is use to provide support for 3D graphics). The classical 3D game Doom has already been developed for Android; some other powerful 3D application such as Google Earth and fantastic game Second Life likely to be implemented for Android mobile.
2.4 SQLite - Android use small sized SQLite as an RDMS (approximately equals to 500 Kb) for database storage. It has got single _le to store all tables, definitions and other data. SQLite simple architecture and small size made it very suitable to limited resourced mobile devices.
2.5 Connectivity - Android is provided with modern day communication technologies. It supports Bluetooth, Wi-Fi, UMTS, CDMA, EDGE (Enhanced Data GSM Environment) and 3G.
2.6 Media Support - Android has got support for different picture formats, including JPEG, BMP, GIF, PNG etc. H.263 and H.264 are video coding techniques supported by Android. H.263 is specialized for video conferencing; H.264 is basically MPEG-4 standard, use to offer high video compression. MP3, WAV, AMR, AMR-WB are used to support audio compression by Android. All audio and video coding technique mention here, are supported by MPEG-3 and 3GP multimedia

container. GPS, compass, and accelerometer are some other features supported by Android. The implementation of these features varies from hardware to hardware.

2. Architecture of Android

The architecture of android has been explained in five layers by the developers which is described here.

3.1 Linux Kernel

The basic layer is the Linux kernel. The whole Android OS is built on top of the Linux 2.6 Kernel with some further architectural changes made by Google. It is this Linux that interacts with the hardware and contains all the essential hardware drivers. Drivers are programs that control and communicate with the hardware. For example, consider the Bluetooth function. All devices has a Bluetooth hardware in it. Therefore the kernel must include a Bluetooth driver to communicate with the Bluetooth hardware. The Linux kernel also acts as an abstraction layer between the hardware and other software layers. Android uses the Linux for all its core functionality such as Memory management, process management, networking, security settings etc.

3.2 Libraries

The next layer is the Android's native libraries. It is this layer that enables the device to handle different types of data. These libraries are written in C or C++ language and are specific for a particular hardware.

3.3 Android Runtime

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

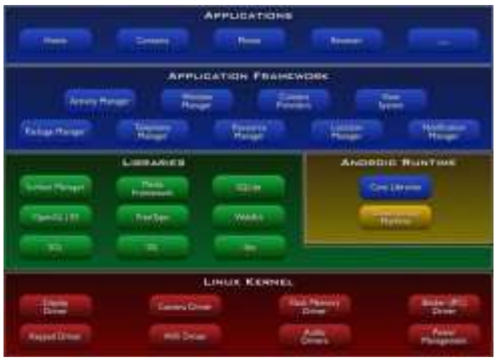

Fig1- Architecture of Android

3. Life Cycle of Android

Here in this section we discuss about the life cycle of the android activity starting from its creation till it is destroyed. It is somewhat similar like that of the applets in Java. The functions of the android lifecycle are as below:

onCreate() :
Called when the activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, etc. This method also provides you with a Bundle containing the activity's previously frozen state, if there was one. Always followed by onStart().

onRestart() :
Called after your activity has been stopped, prior to it being started again. Always followed by onStart().

onStart() :
Called when the activity is becoming visible to the user. Followed by onResume() if the activity comes to the foreground, or onStop() if it becomes hidden.

onResume() :
Called when the activity will start interacting with the user. At this point your activity is at the top of the activity stack, with user input going to it. Always followed by onPause().

onPause ():
Called as part of the activity lifecycle when an activity is going into the background, but has not (yet) been killed. The counterpart to onResume(). When activity B is launched in front of activity A, this callback will be invoked on A. B will not be created until A's onPause() returns, so be sure to not do anything lengthy here.

onStop():
Called when you are no longer visible to the user. You will next receive either onRestart(), onDestroy(), or nothing, depending on later user activity. Note that this method may never be called, in low memory situations where the system does not have enough memory to keep your activity's process running after its onPause() method is called.

onDestroy() :
The final call you receive before your activity is destroyed. This can happen either because the activity is finishing (someone called finish() on it, or because the system is temporarily destroying this instance of the activity to save space. You can distinguish between these two scenarios with the isFinishing() method.

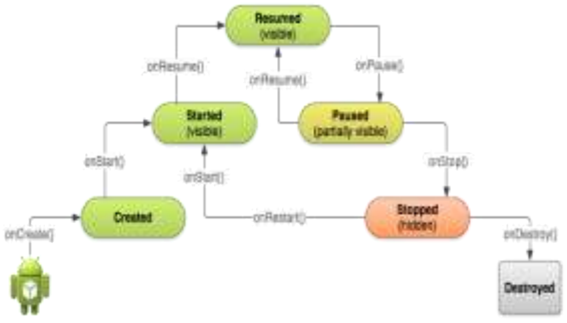

Fig2: Life Cycle of Android

During the life of an activity, the system calls a core set of lifecycle methods in a sequence similar to a step pyramid. That is, each stage of the activity lifecycle is a separate step on the pyramid. As the system creates a new activity instance, each callback method moves the activity state one step toward the top. The top of the pyramid is the point at which the activity is running in the foreground and the user can interact with it.
the activity can exist in one of only three states for an extended period of time:
Resumed: In this state, the activity is in the foreground and the user can interact with it. It is also sometimes referred to as the "running" state.
Paused: In this state, the activity is partially obscured by another activity; the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.

Stopped: In this state, the activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables are retained, but it cannot execute any code. The other states (Created and Started) are transient and the system quickly moves from them to the next state by calling the next lifecycle callback method. That is, after the system calls onCreate(), it quickly calls onStart(), which is quickly followed by onResume().
Now moving on to the final section of the review paper, we here discuss about the basics of the android software for the beginners.

4. ECLIPSE

The software requirements for the installation of Android software are:

- JDK – Java Development Kit
- ECLIPSE IDE
- ADT- Plugins.

5.1 Eclipse IDE
The first step towards developing any applications is obtaining the integrated development environment (IDE). In the case of Android, the recommended IDE is Eclipse, a multi-language software development environment featuring an extensible plug-in system. It can be used to develop various types of applications, using languages such as Java, Ada, C, C++, COBOL, Python, etc.

Fig3: icon screen of ECLIPSE

5.2 ADT Plugins
The Android Development Tools (ADT) plug-in for Eclipse is an extension to the Eclipse IDE that supports the creation and debugging of Android applications. Using the ADT, you will be able to do the following in Eclipse:

- Create new Android application projects.
- Access the tools for accessing your Android emulators and devices.
- Compile and debug Android applications.
- Export Android applications into Android Packages (APK).

JDK is the basic necessity for the system in order to install the Eclipse IDE.

CONCLUSION

This paper is such to keep the basic view for the beginners to as to help them install their software on their system. The programming in Android comprises of JAVA at the back end and XML at the front end. XML is used for the layout while JAVA for the functionality of the app. This app developed in this software is by default saved with **.apk** as the file extension.

REFERENCES

[1] http://www.tutorialspoint.com/android/
[2]http://www.tutorialspoint.com/android/android_architecture.htm
[3]https://www.youtube.com/watch?v=dGyWkDqmjIA
[4] http://www.sitepoint.com/12-android-tutorials-beginners/
[5]http://developer.android.com/design/handhelds/index.html
[6] "Android Development Tutorials", Nikhil Yadav, CSE 40816/ 60816,- Pervasive Health Fall-2011.