ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 3, Issue 6, June 2014*

# Application Development in Android

**Sana[1], Dr. Ravindra kumar[2]**

Department of Computer Engineering, Al-Falah School of Engineering & Technology, Haryana, India[1]

Ex-Director General, VGI Dadri[2]

**Abstract:** Apps are usually available through application distribution platforms. Android came in the world with a boom and has given a new era of technology. A new application is provided as instance to illustrate the basic working processes of Android application components. A guidance to understand the operation mechanism of Android applications and to develop an application on Android platform is described in this paper.

**Keywords:** Dalvik virtual machine; Framework; Activity;Linux kernal

## I. INTRODUCTION

Android is an operating system based on the Linux kernel, and designed primarily for touch screen mobile devices such as smart phones and tablet computers. Initially developed by Android, Inc, which Google backed financially and later bought in 2005. Android" is the package of software used in the mobile devices. It is a comprehensive operating environment which is released on Nov 12, 2007 by the open Handset alliance of Google, a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Android OS is an open source. It is based on Linux V2.6 kernel. The android has a operating system, middleware and core applications. The term "app" is a shortening of the term "application software". It has become very popular and in 2010 was listed as "Word of the Year" by the American Dialect Society. The popularity of mobile apps has continued to rise, as their usage has become increasingly prevalent across mobile phone users. A May 2012 comScore study reported that during the previous quarter, more mobile subscribers used apps than browsed the web on their devices: 51.1% vs. 49.8% respectively. Researchers found that usage of mobile apps strongly correlates with user context and depends on user's location and time of the day. The concept of android is very interesting due to a variety of callable user experience, which has optimized graphic systems, rich media support and a very powerful brown.

## II. BUILDING BLOCKS

A high level overview of what activities are, how intents work, why services are cool, how to use broadcast receivers and content providers to make your app scale, and much more. You should conceptually know when you'd use what component. You will also see how these components relate to a real-world application.

What Are Main Building Blocks?

The main building blocks are components that you use as an application developer to build Android apps. They are the conceptual items that you put together to create a bigger whole. When you start thinking about your application, it is good to take a top down approach. You design your application in terms of screens, features, and the interactions between them. You start with conceptual drawing, something that you can represent in terms of "lines and circles." This approach to application development helps you see the big picture—how the components fit together and how it all makes sense.

### 1. Activities

An activity is usually a single screen that the user sees on the device at one time. An application typically has multiple activities, and the user flips back and forth among them. As such, activities are the most visible part of your application. I usually use a website as an analogy for activities. Just like a website consists of multiple pages, so does an Android application consist of multiple activities. Just like a website has a "home page," an Android app has a "main" activity, usually the one that is shown first when you launch the application. And just like a website has to provide some sort of navigation among various pages, an Android app should do the same. On the Web, you can jump from a page on one website to a page on another. Similarly, in Android, you could be looking at an activity of one application, but shortly after you could start another activity in a completely separate application. For example, if you are in your Contacts app and you choose to text a friend, you'd be launching the activity to compose a text message in the Messaging application.

### 2. Activity Life Cycle

Launching an activity can be quite expensive. It may involve creating a new Linux process, allocating memory for all the UI objects, inflating all the objects from XML layouts, and setting up the whole screen. Since we're doing a lot of work to launch an activity, it would be a waste to just toss it out once the user leaves that screen. To avoid this waste, the activity life cycle is managed via Activity Manager. Activity Manager is responsible for creating, destroying, and managing activities. For example, when the user starts an application for the first time, the Activity Manager will create its activity and put it onto the screen. Later, when the user switches screens, the Activity Manager will move that previous activity to a holding place. This way, if the user wants to go back to an older activity, it can be started more quickly.

- **Starting state**

When an activity doesn't exist in memory, it is in a *starting state*. While it's starting up, the activity will go through a whole set of callback methods that you as a developer have an opportunity to fill out. Eventually, the activity will be in a *running state*. Keep in mind that this transition from starting state to running state is one of the most expensive operations in terms of computing time, and this also directly affects the battery life of the device. This is the exact reason why we don't automatically destroy activities that are no longer shown. The user might want to come back to them, so we keep them around for a while.

- **Running state**

The activity in a running state is the one that is currently on the screen and interacting with the user. We also say this activity is in focus, meaning that all user interactions— such as typing, touching the screen, and clicking buttons—are handled by this one activity. As such, there is only one running activity at any given time. The running activity is the one that has priority in terms of getting the memory and resources it needs to run as quickly as possible. This is because Android wants to make sure the running activity is zippy and responsive to the user.

- **Paused state**

When an activity is not in focus (i.e., not interacting with the user) but still visible on the screen, we say it's in a *paused state*. This is not a typical scenario, because the device's screen is usually small, and an activity is either taking up the whole screen or none at all. We often see this case with dialog boxes that come up in front of an activity, causing it to become Paused. All activities go through a paused state en route to being stopped.

- **Stopped state**

When an activity is not visible, but still in memory, we say it's in a *stopped state*. Stopped activity could be brought back to the front to become a Running activity again. Or, it could be destroyed and removed from memory. The system keeps activities around in a stopped state because it is likely that the user will still want to get back to those activities some time soon, and restarting a stopped activity is far cheaper than starting an activity from scratch. That is because we already have all the objects loaded in memory and simply have to bring it all up to the foreground. Stopped activities can be removed from memory at any point.

- **Destroyed state**

A destroyed activity is no longer in memory. The Activity Manager decided that this activity is no longer needed and has removed it. Before the activity is destroyed, it can perform certain actions, such as save any unsaved information. However, there's no guarantee that your activity will be stopped prior to being destroyed. It is possible for a paused activity to be destroyed as well. For that reason, it is better to do important work, such as saving unsaved data, en route to a paused state rather than a destroyed state.

## 3. Intents

Intents are messages that are sent among the major building blocks. They trigger an activity to start up, tell a service to start or stop, or are simply broadcasts. Intents are asynchronous, meaning the code that sends them doesn't have to wait for them to be completed. An intent could be explicit or implicit. In an explicit intent, the sender clearly spells out which specific component should be on the receiving end. In an implicit intent, the sender specifies the type of receiver.

## 4. Services

Services run in the background and don't have any user interface components. They can perform the same actions as activities, but without any user interface. Services are useful for actions that we want to perform for a while, regardless of what is on the screen. For example, you might want your music player to play music even as you are flipping between other applications. Don't confuse the Android services that are part of an Android app with native Linux services, servers, or daemons, which are a much lower-level component of the operating system. Services have a much simpler life cycle than activities. You either start a service or stop it. Also, the service life cycle is more or less controlled by the developer, and not so much by the system.

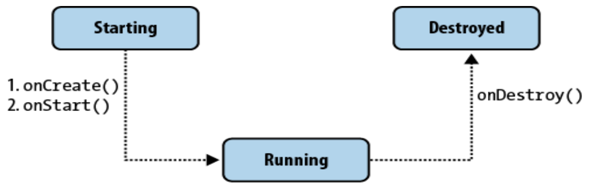

Fig1. Life cycle

## 5. Content Providers

Content providers are interfaces for sharing data between applications. By default, Android runs each application in its own sandbox so that all data that belongs to an application is totally isolated from other applications on the system. Although small amounts of data can be passed between applications via intents, content providers are much better suited for sharing persistent data between possibly large datasets. As such, the content provider API nicely adheres to the CRUD principle. Figure illustrates how the content provider's CRUD interface pierces the application boundaries and allows other apps to connect to it to share data.

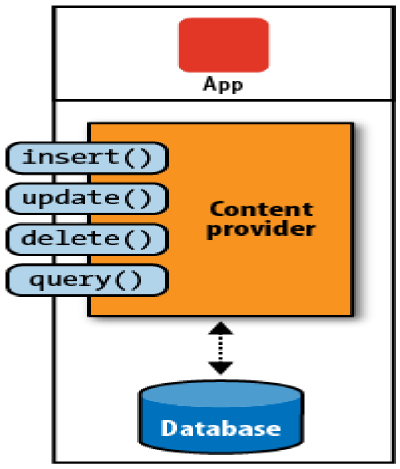

Fig2. Content provider

### 6. Broadcast Receivers

Broadcast receivers are Android's implementation of a system-wide publish/subscribe mechanism, or more precisely, an Observer pattern. The receiver is simply dormant code that gets activated once an event to which it is subscribed happens. The system itself broadcasts events all the time. For example, when an SMS arrives, a call comes in, the battery runs low, or the system gets booted, all those events are broadcasted, and any number of receivers could be triggered by them. In our Twitter app example, we want to start the update service once the system starts up. To do that, we can subscribe to the broadcast that tells us the system has completed booting up. You can also send your own broadcasts from one part of your application to another, or to a totally different application. Broadcast receivers themselves do not have any visual representation, nor are they actively running in memory. But when triggered, they get to execute some code, such as starting an activity, a service, or something else.

## III. ANDROID APPLICATION

While developing a mobile application there are some factors which are needed to be considered these are:

- Target mobile device
- Mobile functionality
- Usability
- Resource permission
- Software performance

Google provide android software development kit (SDK) to create android application using java. For developing an android application we must need some software and it comes in a package and this package contains:-

- IDE- Integrated Development Environment
- SDK- Software Development Kit

**Chat ON**

As we all know the rapid use of social networking applications has lead to growth of smart phone. The new generation of communication has given boom in smart mobile phone platform. . Here comes the term "Android", is the package of software used in the mobile devices. Here I am developing an android application which has key feature communication. It is an application where we can write text by using voice reorganization technique. It is a messaging application where we can send messages using voice. In this application the word we say is get converted into text format and then we can send the message to the required contact. The major benefit of using this application is security from this messaging technique no one can see what messages we have sent. There is no record of sending messages. The other one is that for sending messages from this application the receiver doesn't require that this application to be installed in his/her phone. This is an application which is made using eclipse platform plug-in with android development kit (ADT) bundle. Designing of this application is done in xml and the development code is done in java language.
Android has a number of layouts given to design the user interface of any android application.

- Linear layout- sequential
- Table layout- rows or column
- Frame layout- section wise division
- Relative layout- position layout
- Absolute layout- position view by own choice
- Nested layout- different layout for different parts of an interface

For creating an application we need to do following steps:

$$\begin{tikzcd}
\text{File} \arrow[d] \\
\text{New android application} \arrow[d] \\
\text{Name of file (Stephen)} \arrow[d] \\
\text{Next} \arrow[d] \\
\text{Blank activity} \arrow[d] \\
\text{Finish} \arrow[d] \\
\text{Stephen(file name)} \arrow[d] \\
\text{src} \arrow[d] \\
\text{Main activity.java (java file)} \\
\text{Generated java file (R. file)} \arrow[d] \\
\text{Res} \arrow[d] \\
\text{Layout} \\
\text{Main activity.xml} \\
\text{Gen} \arrow[d] \\
\text{com.example} \\
\text{R.java (This file can never be modified)}
\end{tikzcd}$$

For running the program made in eclipse we need first to make AVD (android virtual device).

$$\begin{tikzcd}
\text{Windows} \arrow[d] \\
\text{AVD} \arrow[d] \\
\text{New (name………., device…….)} \arrow[d] \\
\text{Start} \arrow[d] \\
\text{Run as configuration}
\end{tikzcd}$$

After making the AVD we need to execute the program and we see the desired output.For this the following steps are required:

$$\begin{tikzcd}
\text{Project} \arrow[d] \\
\text{Project name (Stephen)} \arrow[d] \\
{}
\end{tikzcd}$$

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940
*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 3, Issue 6, June 2014*

$$\begin{tikzcd}[row sep=small]
\text{Main activity} \arrow[d] \\
\text{Target} \arrow[d] \\
\text{Apply} \arrow[d] \\
\text{Run}
\end{tikzcd}$$

This is the final step required to run an application developed in android and the final result will be shown in the emulator.

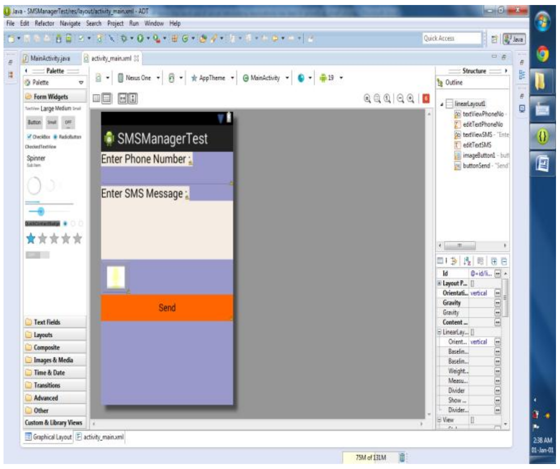

Fig.3-New Developed App

## IV. CONCLUSION

From this paper we are able to get the detailed knowledge of android application. It is an open source and free mobile device platform with its powerful function and good user experience. . The platform of android is based on your programming language. The operating system of android is designed for the smart phones . The concept of android is very attracting due to the reason more and more programmers in the field of mobile computing. It also supports a variety of callable user experience, which has optimized graphic systems, rich media support and a very powerful brown. This article contains the architecture frame work working principal of android. This new application help the user to send the text messages very easily it also make the people of old age comfortable in sending messages without finding any difficulties. Also help the busy persons to send messages without stopping their work. There are further more enhancement is done in this application to make it more simple and user friendly.

## REFRENCES

[1] Android application development by Daniel Switkin
[2] MOBILE APPLICATION DEVELOPMENT "The search for common ground in a divided market" by Ben Feigin
[3] Introducing mobile application development for android
[4] Wikipedia
[5] Open Hanset Alliance, http://www.openhandsetalliance.com/.
[6] Android - An Open Handset Alliance Project, http://code.google.com- /intl/zhCN/android/. J.F. DiMarzio, Android A Programmer's Guide, Chicago: McGraw-Hill, Jul. 2008.
[7] Android Developers, http://www.androidin.com/.
[8] C. Haseman, Android Essentials, PDF Electronic Book, 2008. Available from: http://androidos.cc/dev/index.php.
[9] N. Gramlich, Android Programming , PDF Electronic Book, 2008. Acquisition and analysis of volatile memory from android devices Joe Sylve a, Andrew Case b, Lodovico Marziale b, Golden G. Richard a,* a Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA b Digital Forensics Solutions, LLC, New Orleans, LA 70130, USA
[10] Fernflower - java decompiler. http://www.reversed-java.com/fernflower/.
[11] Fortify 360 Source Code Analyzer (SCA). https: //www.fortify.com/products/fortify360/source-code-analyzer.html.
[12] Jad. http://www.kpdus.com/jad.html.
[13] Jd java decompiler. http://java.decompiler.free.fr/.
[14] Mochathe java de-compiler. http://www.brouhaha.com/~eric/software/mocha/.
[15] ADMOB. AdMob Android SDK: Installation Instructions. http://www.admob.com/docs/AdMob_Android_SDK_Instructions.pdf. Accessed November 2010.
[16] ASHCRAFT, K., AND ENGLER, D. Using Programmer-Written Compiler Extensions to Catch Security Holes. In Proceedings of the IEEE Symposium on Security and Privacy (2002).
[17] BBC NEWS. New iPhone worm can act like botnetsay experts. http://news.bbc.co.uk/2/hi/technology/8373739.stm, November 23, 2009.
[18] BORNSTEIN, D. Google i/o 2008 - dalvik virtual machine internals. http://www.youtube.com/watch?v=ptjedOZEXPM.
[19] BURNS, J. Developing Secure Mobile Applications for Android. iSEC Partners, October 2008. http://www.isecpartners.com/files/iSEC_Securing_Android_Apps.pdf.
[20] CHEN, H., DEAN, D., AND WAGNER, D. Model Checking One Million Lines of C Code. In Proceedings of the 11th Annual Network and Distributed System Security Symposium (Feb. 2004).
[21] EGELE, M., KRUEGEL, C., KIRDA, E., AND VIGNA, G. PiOS: Detecting Privacy Leaks in iOS Applications. In Proceedings of the Network and Distributed System Security Symposium (2011).
[22] EGELE, M., KRUEGEL, C., KIRDA, E., YIN, H., AND SONG, D. Dynamic Spyware Analysis. In Proceedings of the USENIX Annual Technical Conference (June 2007), pp. 233–246.
[23] ENCK, W., GILBERT, P., CHUN, B.-G., COX, L. P., JUNG, J., MCDANIEL, P., AND SHETH, A. N. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (2010).

## BIOGRAPHY

**I SANA** is a student of computer science & engineering at the Al-Falah University, Faridabad. I am pursuing my M.Tech and eventually earned my degree in Bachelor of Technology from WBUT. I have recently attended international conferences and seminars with friend and colleague. I am presenting research compiled during my course work. I love family a lot they are key of my success.