IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 28-32
www.iosrjournals.org

Research on Development of Android Applications

Mrs. Prachi Sasankar[1]. Mrs. Usha Kosarkar.[2]
[1](Prachi.sasankar@raisoni.net, *BCA, Sadabai Raisoni Women's College,Nagpur SNDT Women's University,Mumbai, India.)*
[2]*(Usha.kosarkar@raisoni.net , BCA, G.H.R.I.I.T.Nagpur, R.T.M.Nagpur University,Nagpur.,India)*

Abstract-*Introduced the Android platform and the features of Android applications, gave a detailed description of Android application framework from the prospective of developers. A simple music player is provided as instance to illustrate the basic working processes of Android application components. This paper provides guidance to understanding the operation mechanism of Android applications and to developing applications on Android platform.*
Keywords-*Linux kernel, Android system, Application framework, Dalvik virtual machine.*

I. INTRODUCTION

Application framework defined the common structure of programs in the specific domain. Essentially, a framework is a component that can be reused, it sets the architecture of applications and incorporated as a set of abstract classes and the cooperation of their instances. Android is an open source operating system based on Linux kernel and launched by Google. Unlike PC operating system, mobile phone operating systems are constrained by their hardware, storage space, power dissipation and mobility conditions. Compared with the development of applications on PC, there are some different features of applications on mobile phone operating systems. This paper introduces the basic architecture and application framework of Android operating system, gives a detailed description of main structure of Android applications and the methods of developing applications based on Android application framework.

II. INTRODUCTION OF ANDROID OPERATING SYSTEM

Android is a comprehensive operating environment that based on Linux® V2.6 kernel, it is also a layered system, the architecture of Android system is shown as picture 2-1[1].

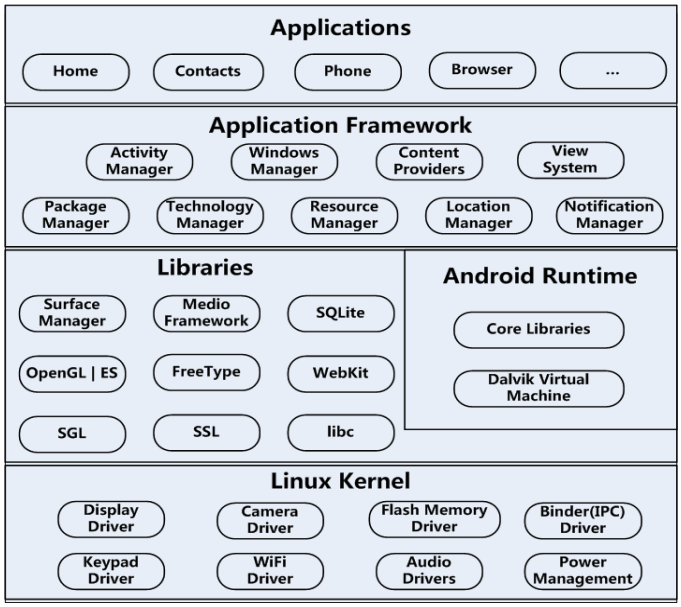

Fig. 2 – 1

Applications layer is the site of all Android applications including an email client, SMS program, maps, browser, contacts, and others. All applications are written using the Java programming language.

Application framework layer defined the Android application framework. All Android applications are based on the application framework. The Android application framework includes:

- A rich and extensible set of Views that can be used to build an application with beautiful user interface, including lists, grids, text boxes, buttons, and even an embeddable web browser.
- A set of Content Providers that enable applications to access data from other applications (such as contacts) or to share their own data.
- A Resource Manager that provides access to non- code resources such as localized strings, graphics, and layout files.
- A Notification Manager that enables all applications to display custom alerts in the status bar.
- An Activity Manager that manages the lifecycle of applications and provides a common navigation
- back stack.[1]

Libraries layer includes a set of C/C++ libraries used by various components of the Android system and provides support to the application framework. Android Runtime includes a set of core libraries and a Java virtual machine (Dalvik virtual machine) which is redesigned and optimized by Google to be suitable for Android platform.

Linux kernel is located at bottom layer of Android system and acts as an abstraction layer between the hardware and the rest of the software stack. It provides core system services such as security, memory management, process management, network stack, and driver model. In addition, some bottom functions such as management of threads of Dalvik virtual machine also rely on the Linux kernel.

III. DALVIK VIRTUAL MACHINE

As previously mentioned, Android is running on the Linux kernel and its applications are written by Java programming language, so Android applications are running on a Java virtual machine named Dalvik virtual machine. Dalvik virtual machine has been redesigned and optimized by Google for the hardware features of mobile devices. In Android system, there is a tool named dx, included in the Android SDK transforms the Java Class files (which are compiled by a regular Java compiler) into the .dex format.The .dex format files integrate all Java class files and delete redundant information in every Java class files.

There are several other features of Dalvik virtual machine:

- Dalvik virtual machine could have multiple instances on one device and every instance runs in a separate Linux process, an Android application runs in an instance of a Dalvik virtual machine.
- Dalvik virtual machine relies on the underlying operating system (Linux kernel) for process isolation, memory management and threading support.
- Dalvik virtual is register-based.

The follow figure (figure3-1) shows the position of Dalvik virtual machine in Android system.

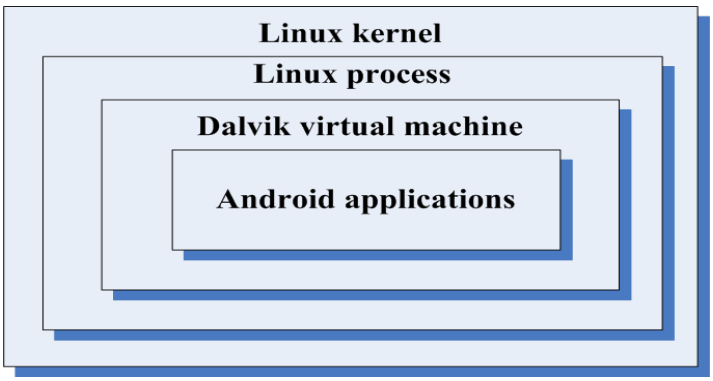

Fig-3-1

IV. ANDROID APPLICATION COMPONENT

A core feature of Android is that one application could use component element that belong to another application (if the component is permitted using). In order to achieve such functions, Android system must launch the application while any part of the application is asked and instantiate Java objects that being asked. Unlike most operating system, there is no single point that the system can enter in an Android application (for

example, there no main ( ) function in an Android application). Instead, each component is a different point through which the system can enter an application and instantiate component object independently.

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

Activity

An activity represents a single screen with a user interface. The activities in an application work together to form a cohesive user experience, but each one is independent of the others. As such, a different application can start any one of these activities. An activity is implemented as a subclass of Activity. The particular form that an activity show users and the amount of activities in an application depend on how the developer design the application. In a multiple activities application, typically, one activity is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). The figure 4-1 shows the lifecycle of an activity.[1]

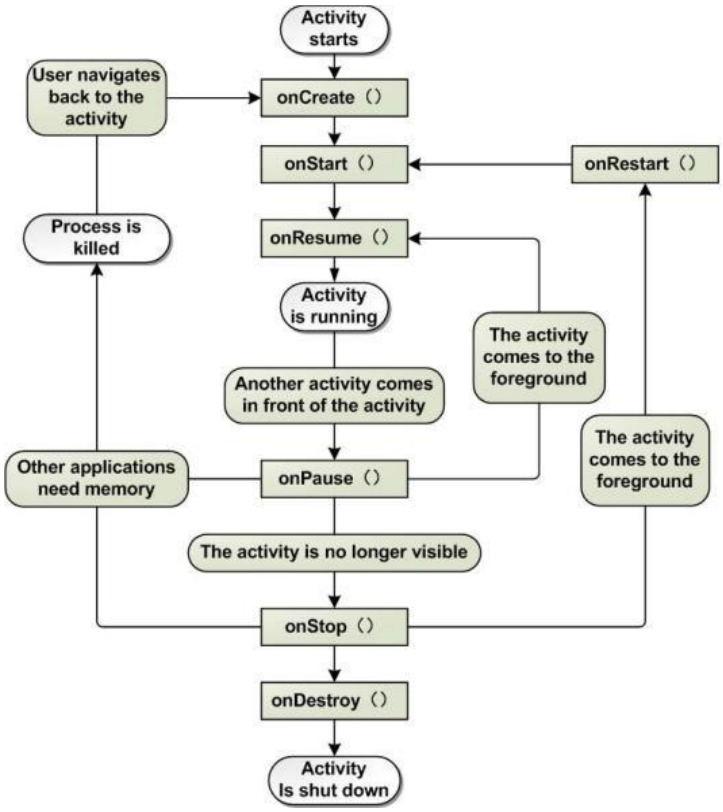

Fig.4 - 1

Service

A service is an Android component that runs in the background to perform long-running operations or to perform work for remote processes and does not provide a user interface. An activity can connect or bind a service that is running. (if the service is not running, launch it). When connected to a service, the activity can communicate with the service through the interface that the service exposed. Like other application components, service components always running in the main thread of an application by default. So for the intensive or blocking operating a service performs (may slow down activity performance), it is usually start a new thread inside the service.

Content Providers

Content providers provide data share mechanism among applications. The data that be shared could in the file system, a SQLite database, or any other persistent storage location an application can access. A content provider is implemented as a subclass of Content provider, it defines the data format it supported and provides a set of method to enable other applications to query or modify the data. But an application does not call these

methods immediately, instead, it call these methods by an object named Content Resolver. Content Resolver can communicate with every Content Provider. Content Resolver cooperated with Content Provider to manger IPC (inter process communication) while sharing data.

Broadcast Receivers

Broadcast Receivers is in charge of the reception of system wide broadcast and take response aiming at the information that a broadcast transmitted. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low. Applications can also initiate broadcasts. There could be any number of Broadcast Receivers in an application and each Broadcast Receiver implemented as a sub class of Broadcast Receiver. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work.[1]

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message named Intent. Intents bind individual components to each other at runtime no matter the component belongs to the same application. An Intent can create with an Intent object, which defines the messages by which can activate either a specific component or a specific type of component. For activities and services, an intent defines the action to perform and may specify the URI of the data to act ion. For broadcast receivers, the intent simply defines the announcement being broadcast. The other component type, content provider, is not activated by intents. Rather, it is activated when targeted by a request from a Content Resolver.[1]

V. NEW FEATURES OF ANDROID APPLICATIONS

As a young operating system, on one hand, Android could benefit from mature technology of other operating system. On the other hand, Android could also improve the blemish that appears in other operating systems. On a developer's prospective, Android possess following new features:

The permission that an application possessed has been defined clearly. In android applications, all components that could be independently launched by system need to be declared in a XML file named AndroidManifest. The AndroidManifest does a number of things in addition to declaring the application's components, including:

- Identify any user permissions the application requires, such as Internet access. Only identify the permissions that an application requires, the application have permissions to perform operations.
- Declare the minimum API level required by the application.
- Declare hardware and software features used or required by the application.
- Declare API libraries the application needs to be linked against.

Resources are separate from the source code. In Android, all non code resources are defined in XML files. For every resource that included in an Android project, the SDK build tools define a unique integer ID, which can be used to reference the resource from the application code or from other resources defined in XML files. Providing resources separate from source code makes it easy to update various characteristics of an application without modifying code and by providing sets of alternative resources enables developers to optimize the application for a variety of device configurations, such as different languages and screen sizes.

VI. AN MUSIC PLAYER

Here is a simple music player, the four components of Android have been defined in this example. MusicMainActivity is an object of Activity type, it provides interface to users and communicates with a Service used to play music in background by a BroadcastReceiver. MusicPlayerService is an object of extensional service type, it mainly used to play music in background and return play statues to MusicMainActivity by broadcast. MusicInfoManager is a custom class, it packs a ContentProviders provided by system to get music information from flash card. By the above components cooperate with each other can realize the function of play music on android platform.

The following figure (figure 6-1) show the interactive mode among the above components.

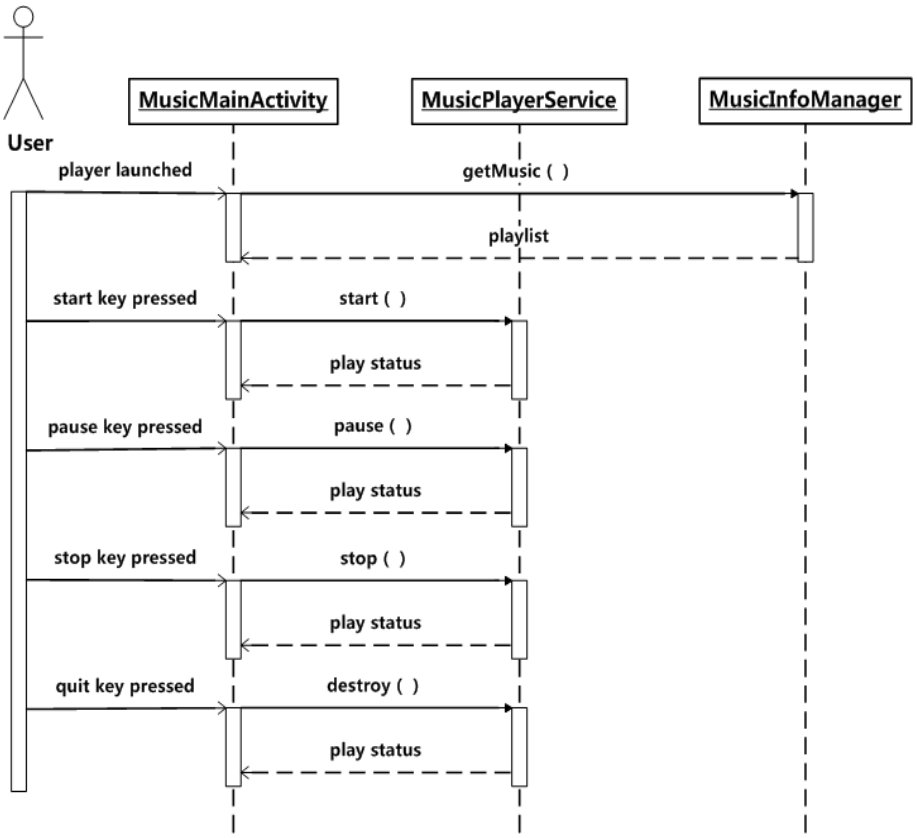

Fig. 6 - 1

As is showed in the figure above, the MusicPlayerService run in background returns all play status to the MusicMainActivity run in foreground by broadcast (use intent as carrier). A BroadcastReceiver has registered in MuxicMainActivity to receive all broadcast from MusicPlayerService. The received broadcast will be resolved by MusicMainActivity. According to the broadcast content, the MusicMainActivity will do some actions (show users the play status). After the music player launched, the MusicMainActivity will send messages to MusicInfoManager for the information of music files and the MusicInfoManager will activate a ContentProvider that provided by system, then get the music files list and return it to the MusicMainActivity.

VII. CONCLUSION

Android as a full, open and free mobile device platform, with its powerful function and good user experience rapidly developed into the most popular mobile operating system. This article gives a detailed introduction of Android application framework and the working principal of Android applications. Finally, a music player on the android platform was put forward as an example to illustrate this mechanism.

REFERENCES

[1]. OL. Google Android Developers, Android Develop Guide, http://developer.android.com/guide/topics/fundamentals.html
[2]. M. Fengsheng Yang, Android Application Development Revelation, China Machine Press, 2010,1
[3]. M. Zhengguo Hu, Jian Wu, Zhenggong Deng, Programming Methodology, National Defence Industry Press, 2008,6
[4]. M. Junmin Ye, Software Engineering, Tsinghua University Press, 2006,6
[5]. J. Dongjiu Geng, Yue Suo, Yu Chen, Jun Wen, Yongqing Lu, Remote Access and Control System Based on Android Mobile Phone'vol.2. Journal of Computer Applications, 2011, pp. 560-562
[6]. J. Li Lin, Changwei Zou, Research on Cloud Computing Based on Android Platform, vol.11. Software Guide, 2010, pp.137-139