Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.712 – 720

RESEARCH ARTICLE

Xamarin for Android Application

Ankit Chandna[1], Neetu Sharma[2]

1Research Scholar, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana

Email ID: ankit.vcer@gmail.com

2HOD, Department of Computer Science and Engineering, Ganga Institute of Technology & Management, Kablana

Email ID: neetush75@gmail.com

Abstract: Xamarin Studio is a modern, sophisticated IDE with many features for creating iOS and Android applications. It includes a rich editor, debugging, native platform integration with iOS and Android, and integrated source control to name just of few of its many features nearly every imaginable platform including Linux, Unix, FreeBSD, and Mac. Efficiency of Xamarin can be defined as the extent of the code shared or reused across all the platforms. It depends on following factors Coupling between UI and Backend Code .Native-OS features to be used in appUI Controls used in application proposed solution is based on the use of mobile devices operating in wireless environments.

INTRODUCTION

Xamarin offers two commercial products, MonoTouch and Mono for Android, also known as Xamarin.iOS and Xamarin.Android, respectively. Both are built on top of Mono, an open-source version of the .NET Framework based on the published .NET ECMA standards.

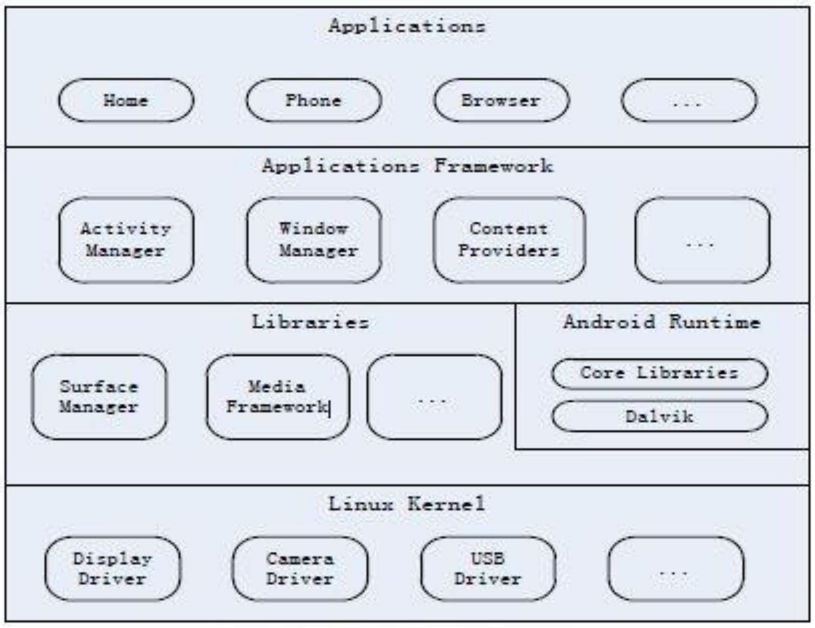
Mono has been around almost as long as the .NET framework itself, and runs on To get started, we are going to walk through the steps you need to take to create a Xamarin.Androidapplication.Xamarin.Android works with Xamarin Studio on both OS X and Windows; it also works on Windows with Visual Studio 2010 Professional (or greater) on each of these platforms. This walkthrough assumes you already have Xamarin.Android installed.

We studied the Android system architecture. Android system is a Linux-based system, Use of the software stack architecture design patterns [1-2].
As shown in the figure, the Android architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework and Applications [5-8].
Each layer of the lower encapsulation, while providing call interface to the upper.
Applications Applications Framework Libraries Android Runtime Linux Kernel Home Phone ... Browser Activity Manager Window Manager Content Providers ... Core Libraries Dalvik Surface Manager Media Framework ... Display Driver Camera Driver USB Driver ...

Android Architecture

Applications

Android app will be shipped with a set of core applications including client, SMS program, calendar, maps, browser, contacts, and others. All these application programs are developed in Java.



Android Architecture

Application Framework

The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its components. Any other app can release its functional components and all other apps can access and use this component (but have to follow the security of the framework). Same as the users can be able to substitute the program components with this reuse mechanism.

Libraries and Android Runtime

The library is divided in to two components: Android Runtime and Android Library. Android Runtime is consisted of a Java Core Library and Dalvik virtual machine. The Core Library provides Java core library with most functions. Dalvik virtual machine is register virtual machine and makes some specific improvements for mobile device.
Android system library is support the application framework, it is also an important link connecting between application framework and Linux Kernel. This system library is developed in C or C++ language. These libraries can also be utilized by the different components in the Android system. They provide service for the developers through the application framework.

Linux Kernel

The kernel system service provided by Android inner nuclear layer is based on Linux 2.6 kernel, Operations like internal storage, process management, internet protocol, bottom-drive and other core service are all based on Linux kernel.

Android Security

The open nature of Android and its large user base have made it an attractive and profitable platform to attack. Common exploits and tool kits on the OS can be utilized across a wide number of devices, meaning that attackers can perform exploits en masse and re-use attack vectors. Google did take measures in the development of the Android kernel to build security measures in; the OS is sandboxed, preventing malicious processes from crossing between applications. Whilst this attempt to eliminate the concept of infection is admirable in some regards, it fails to address the issue of infection altogether. Android is a victim of its own success, not just in the way it has attracted malicious attention, but in its very nature. One of the reasons the OS has succeeded in gaining market share so
rapidly is that it is open source; it is essentially free for manufacturers to implement. Additionally this has led to substantial fragmentation of Android versions between devices and means that vendors have been reluctant to roll-out updates, presumably out of some concern regarding driving demand for future devices

Service

A Service is code that is longlived and runs without a UI. A good example of this is a media player playing songs from a play list. In a media player application, there would probably be one or more activities that allow the user to choose songs and start playing them. However, the music playback itself should not be handled by an activity because the user will expect the music to keep playing even after navigating to a new screen. In this case, the media player activity could start a service using Context.startService () to run in the background to keep the music going. The system will then keep the music playback service running until it has finished. Note that you can connect to a service (and start it if it's not already running) with the Context.bindService () method. When connected to a service, you can communicate with it through an interface exposed by the service. For the music service, this might allow you to pause, rewind, etc.

Multiple language support:

Android supports multiple languages.

Web browser:

The web browser available in Android is based on the opensource Web Kit layout engine, coupled with Chrome’s V8 JavaScript engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

Java support

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third party applications.

Multi-touch:

Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touchscreen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively. 8) Bluetooth: Supports A2DP, AVRCP, sending files (OPP), accessing the phone book (PBAP), voice dialing and sending contacts between phones. Keyboard, mouse and joystick (HID) support is available in Android 3.1+, and in earlier versions through manufacturer

Tethering

Android supports tethering, which allows a phone to be used as wireless/wired Wi-Fi hotspot. Before Android 2.2 this was supported by thirdparty applications or manufacturer customizations.

Screen capture

Android supports capturing a screenshot by pressing the power and volume-down buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android

Experimental Methods

Video Player

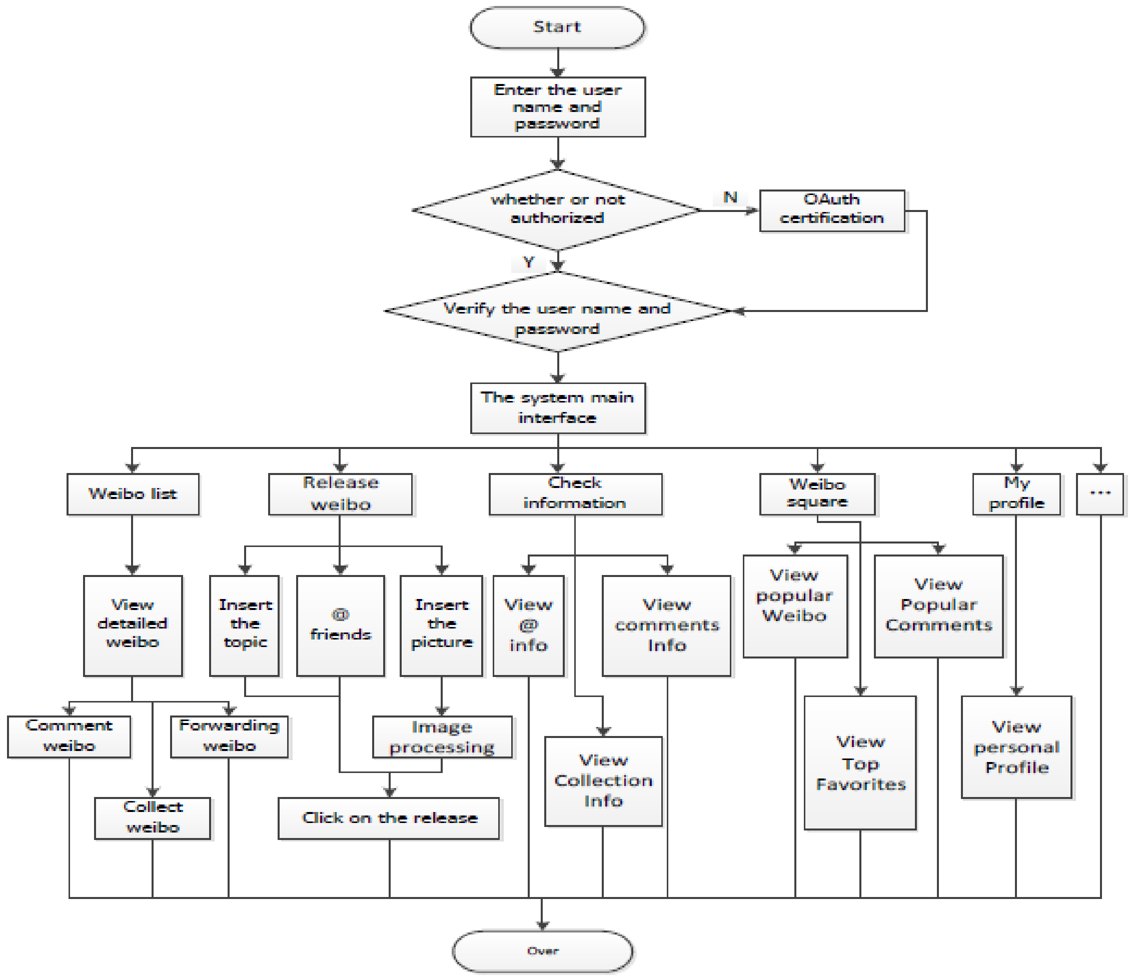
Video Player is achieved through the Eclipse platform. In order to develop android app, we will install a plug-in for Eclipse: Android Development Tools (ADT). Once installed, download Android SDK [10, 12], install and configure the SDK, then we can develop a video player.

Our research begins with the study of operating mechanism, Android platform media layer structure, xml customizable interface, Content Providers [9] achieves file scanning to get a list of media files, MediaPlayer class, file parsing, Surface Flinger interface. After that, we could develop an Android-based mobile video player. Realize media library, video player, file opening, audio, video, photographs and other functions. The Xml file definition interface (Included in the application Framework layer) File list obtained through the Content Providers (Included in the application Framework layer) Using multimedia framework for video file playback (Included in the Libraries layer)

System Flow Chart

The software interface is defined through XML files. XML layout files control view, is not only simple, but also isolated the View control logic from Java code and controlled by inserted into XML files. Reflects the MVC principle in a better way and also reflects the principle of separation of logic and views. This software obtains the list of media files by scanning through Content Providers. Content Providers is recognized as a bridge between the data storing and searching across programs. The function is to achieve data sharing among different Apps, it is the only way to share data with other apps. Below figure shows the media layer structure.

APP（Java） JNI（libmedia_jni.so） libmedia.so mediaplayerserver mediaplayer interface MIDI Player Vorbis Player OpenCore PlayerAudioFlinger AudioFlinger


Start
Enter the user name and password
whether or not authorized
N
OAuth certification
Y
Verify the user name and password
The system main interface
Weibo list
Release weibo
Check information
Weibo square
My profile
...
View detailed weibo
Insert the topic
@ friends
Insert the picture
View @ info
View comments Info
View popular Weibo
View Popular Comments
Comment weibo
Forwarding weibo
Image processing
Collect weibo
Click on the release
View Collection Info
View Top Favorites
View personal Profile
Over

In the Android media layer, the most important class is MediaPlayer.
Class MediaPlayer （Java） JNI Class MediaPlayer Class IMediaPlayerService Class BPMediaPlayerService Class IMediaPlayer Class BPMediaPlayer Class IserviceManager Class BPServiceManager Client Class parcel Class parcel Class parcel Class BnMediaPlayerService Class MediaPlayerService Class BnMediaPlayer Class MediaPlayerService : : client Class BnServiceManager Class BServiceManager Class MediaPlayerBase Class MediaPlayerInterface Class PVPlayer（opencore） Server Create() getService() operations

MediaPlayer Class Hierarchy Diagram

Upper JAVA program calls the underlying MediaPlayer class to implement Media streaming. First, the MediaPlayer class obtains a name for media.player services through IService _Manager getService interface. After that, all the operations are conducted through this MediaPlayer player and the interface is IMediaPlayer. All BpXXXX classes are proxy classes, the responsibility is to realize message forwarding by sending the client requests to the service through the Binder mechanism. A corresponding BnXXXX subclass on the service side is responsible for implementing specific functions. The play of final broadcast media stream is achieved by calling the underlying Opencore libraries through MediaPlayerInterface interface. This class loads pre-played files through Uri, calling the OpenCore multimedia libraries to implement file parsing via JNI, by calling the SurfaceFlinger interface to realize the playing of video file, by calling the AudioFlinger interface to realize the playback of audio data. The software interface is simple, feature-rich, smooth operation and also by calling an external program to achieve audio and image playback.

Audio Player
The audio player development tool is the same as the one of video player. System structure and the process is the same as the process of video player. Also defines the interface in the Application Framework layer, and then acquires music files through ContentResolver in the Android Framework layer. Finally, plays the music by using the Service component calling the MediaPlayer class in the Libraries layer.
Main Interface Music List Artist List Album List Audio Playback Interface backstage service

System Structure
The main interface module is the entrance of the application. Users will see the main interface modules after starting the application. The module itself does not reflect any of the information to the user, just call list module to display. Three lists are demonstrated:

music list, album list and artists list. The main interface module is realized by calling MusicListActivity, AlbumListActivity and ArtistListActivity module.
MainActivity MusicListActivity AlbumListActivity ArtistListActivity Intent Intent Intent

Mail Interface

ListView component.

When the user selects an element of the ListView, this module will encapsulate the information into an intent object and sent it to music playback module.
Music playback module collects the intent sent from List module and analyzed it, then calls the background music services to play the audio file. The View components provides player with some basic functions, such as play, pause, fast forward, fast rewind, single play, random play, etc. This module will make the corresponding logical analysis after the users did operations to the components. Appropriate response and changes will be done according to the results analyzed. Audio player interface is shown in

Start Enter the user name and password whether or not authorized OAuth certification Verify the user name and password The system main interface Weibo list Release weibo Check information Weibo square My profile … Y N Insert the topic Insert the picture @ friends Image processing Click on the release View detailed weibo Comment weibo Forwarding weibo Collect weibo Over View @ info View comments Info View Collection Info View popular Weibo View Popular Comments View Top Favorites View personal Profile Specific functions of this system development are based on Android Weibo SDK, calling its wrapper classes to complete the corresponding task.
For example, the main interface is divided into three parts: the top is a toolbar, the middle area is a ListView, bottom is the button bar. Weibo List is in the middle of the main interface part, displayed through the ListView. As long as Weibo data was obtained from the service side, it will be shown directly on the ListView control through the Adapter.
The function of posting is achieved through WeiboManager.update method. This method can submit text and message containing pictures.
Browse Weibo interface is used to display all the information of Weibo and realize forwarding and commenting. The Weibo browse window class is WeiboViewer and the interface layout file is weibo_viewer.xml. When displaying the Weibo information, accessing the current Weibo Status objects and transfers them to the WeiboViewer, and then calls the loadContent method to load Weibo information

Conclusion

The test involves three environments including hardware, software and network. Test hardware environment is Lenovo Y460 laptop and millet M1 phone; software environment is windows 7 and phone system environment is Android 4.0.3. Network environment Mobile which is 10M broadband, WIFI LAN and Mobile GPRS network.
By testing each function on mobile phone and the computer simulator, the result showed that video player and audio player run well and no advertising. Sina weibo client can successfully complete OAuth2.0 certificate authority and login and collect the basic data of the user information from sina server and no redundant information. Expected effect is achieved after testing all the functions.
Since the Weibo client has to access to the network, when tested on an Android phone, the performance under the environment of WIFI network and mobile 3G GPRS network can meet the expected requirements.

References

[1] h t t p : / / w w w 2 . d c s e c . u n i - hannover.de/files/android/p50- fahl.pdf
[2] http://digitalforensicssolutions. com/papers/android-memoryanalysis-DI.pdf
[3] http://www.uandistar.org/2011/ 0 6 / p a p e r-p r e s e n t a t i o n -o n - android.html
[4] http://www.studymode.com/ essays/Android-Research-Paper- 1068648.html
[5] h t t p : / / w w w . 4 s h a r e d . c o m / office/0RX_5-iE/file.html
[6] h t t p : / / w w w . i m m a g i c . c o m / e L i b r a r y / A R C H I V E S / G E N E R A L / W I K I P E D I / W110410O.pdf
[7] http://students.mint.ua.edu/ ~pmkilgo/etc/android-os.pdf
[8] h t t p : / / w w w . a c u m i n . c o . u k / download_files/WhitePaper/ android_white_pap
[9] M. Butler, “Android: Changing the Mobile Landscape”, Pervasive Computing, (2011), pp. 4-7.
[10] B. Proffitt, “Open Android-For better and for worse”, Spectrum, (2011), pp. 22– 24.