안드로이드 플랫폼과 서비스 기반 모바일 어플리케이션

김 평 중*

1. 서 론

안드로이드(android)란 모바일 기기용 운영체제(operating system)·미들웨어(middleware)· 응용프로그램(key application)을 포함하는 소프트웨어 플랫폼이다. 안드로이드 SDK(Software Development Kit)는 안드로이드 플랫폼에서 자바 프로그래밍 언어를 이용하여 개발할 수 있도록 툴(tool)과 API(Application Programming Interface)를 제공한다[1].

'안드로이드' 어원은 인간의 모습을 한 로봇을 의미하는 그리스어 ανήρ (anēr, man)의 파생 단어인 ανδρός이다. '안드로이드'란 이름은 2005년 7월, 구글은 미국 캘리포니아 팔로알토의 안드로이드社를 인수하고, 2007년 11월, 구글 등 48개 하드웨어, 소프트웨어, 및 통신 회사를 포함하는 OHA(Open Handset Alliance)를 결성한다. OHA는 모바일 기기의 공개 표준을 개발하기로 결정하고, 리눅스(Linux) 커널 2.6에서 빌드된 그들의 첫 번째 모바일 기기 플랫폼 결과물인 안드로이드를 발표한다. 이는 OHA가 개발하여 공개하였지만, 실질적으로 구글이 주도하였으므로 '구글 안드로이드'라고 부른다. 2008년 10월, 구글은 개발된 안드로이드 플랫폼을 오픈 소스로 선언하고 이의 모든 소스 코드를 아파치 라이선스로 배포하고 있다[2].

모바일 디바이스는 기본적으로 휴대폰으로서의 전화 기능을 제공함은 물론, 다양한 소프트웨어 어플리케이션을 설치 운영할 수 있는 컴퓨터의 기능도 제공한다. 따라서 모바일 디바이스는 PC 기능뿐만 아니라, 기업의 엔터프라이즈 어플리케이션의 클라이언트 단말로도 사용될 것으로 예상된다. 또한, 모바일 디바이스는 이동성을 제공하고, 위치 파악, 가속 측정 등 주변상황을 인지하는 기능도 제공하므로, 다양한 어플리케이션 개발과 응용을 가능케 한다. 그러나 모바일 디바이스는 물리적인 작은 크기로 인하여 컴퓨팅 파워와 메모리, 화면 크기, 배터리 수명 등의 자원 측면에서 제약점이 있다. 따라서 복잡한 기능의 어플리케이션은 이 디바이스 상에서 설치와 운영이 어렵다. 이러한 단점을 극복하고 모바일 디바이스의 활용을 최대화하기 위해 서비스 기반의 모바일 어플리케이션(Service-based Mobile Application, SMA)이 각광을 받고 있다[3].

안드로이드는 기존의 WIPI, BREW, GVM 등과 같은 모바일 디바이스를 위한 플랫폼이다. 아주 단순하게 생각하면 PC 위에 돌아가는 Win-

※ 교신저자(Corresponding Author): 김평중, 주소: 대전광역시 유성구 지족동 열매마을 Apt. 501-701호(305-770) 전화: 043)730-6351, FAX: 043)730-6359, E-mail: pjkim@cpu.ac.kr
* 충북도립대학 컴퓨터정보과

dows와 같은 운영체제라고 생각해도 된다. 안드로이드는 리눅스 커널 위에서 동작하며, 포괄적 라이브러리 세트, 풍부한 멀티미디어 사용자 인터페이스, 폰 어플리케이션 등을 제공한다. 안드로이드는 개발자들이 자바 언어로 어플리케이션을 작성 할 수 있게 하였으며, 컴파일된 바이트코드(byetcode)를 구동할 수 있는 런타임 라이브러리(runtime library)를 제공한다.

안드로이드는 휴대폰에 탑재하여 인터넷과 메신저 등을 이용할 수 있으며, 휴대폰뿐 아니라 다양한 정보 가전 기기에 적용할 수 있는 연동성도 갖추고 있다. 안드로이드는 기존의 자바 가상 머신과는 다른 가상 머신인 달빅(dalvik) 가상 머신을 통해 자바로 작성된 어플리케이션을 별도의 프로세스에서 실행하는 구조로 되어 있다[4].

개발자들이 Windows에서 어플리케이션을 개발하듯이 안드로이드 SDK를 사용하면 안드로이드 폰에서 동작하는 어플리케이션을 만들 수 있다. 어플리케이션들은 Java 프로그래밍 언어로 작성해야 하고 Dalvik 위에서 실행된다. Dalvik은 구글이 만든 가상 머신인데, Linux 커널의 최상위 영역에서 동작한다.

안드로이드가 기존의 휴대폰 운영체제인 마이크로소프트의 '윈도 모바일'이나 노아의 '심비안'과 차별화되는 것은 완전 개방형 플랫폼이라는 점이다. 종전에는 휴대폰 제조업체와 서비스 업체마다 운영체제가 달라 개별적으로 응용프로그램을 만들어야 하였다. 이에 비하여 안드로이드는 기반 기술인 소스 코드를 모두 공개함으로써 누구라도 이를 이용하여 소프트웨어와 기기를 만들어 판매할 수 있도록 한다. 개발자들은 이를 확장, 대체 또는 재사용하여 사용자들에게 풍부하고 통합된 모바일 서비스를 제공할 수 있게 된 것이다[4].

2. 안드로이드 플랫폼

안드로이드 플랫폼은 운영체제, 미들웨어, 키 응용프로그램들을 포함한 모바일 디바이스를 위한 소프트웨어 집합이다. 그림 1은 안드로이드의 주요 구성 요소를 보여준다. 그림 1은 4개의 계층으로 분류하고 Application, Application Framework, Libraries, Android Runtime, Linux Kernel로 구성한다[5].

2.1 어플리케이션(Application)

안드로이드 어플리케이션은 이메일을 확인할 수 있는 클라이언트, SMS 프로그램, 캘린더, 지도, 브라우저, 주소록 등을 키 어플리케이션으로 제공한다. 모든 어플리케이션들은 자바 언어로 작성한다.

2.2 어플리케이션 프레임워크(Application Framework)

응용 개발자는 키 응용프로그램에서 사용한 것처럼 어플리케이션 프레임워크 API를 모두 접근하여 자신만의 어플리케이션을 개발한다. 어플리케이션 프레임워크는 각종 응용을 개발하기 위한 클래스와 메소드들을 제공한다. 안드로이드 아키텍처는 컴포넌트를 재사용하기 쉽도록 디자인 되어 있다. 하지만 프레임워크의 보안 제약은 따라야 한다. 이와 같은 메커니즘은 컴포넌트를 사용자에 의해 교체할 수 있도록 한다.

2.3 라이브러리(Libraries)

안드로이드는 안드로이드 시스템에서 다양하게 사용되는 C/C++ 라이브러리들을 포함하며, 안드로이드 Application Framework를 통해 개발자

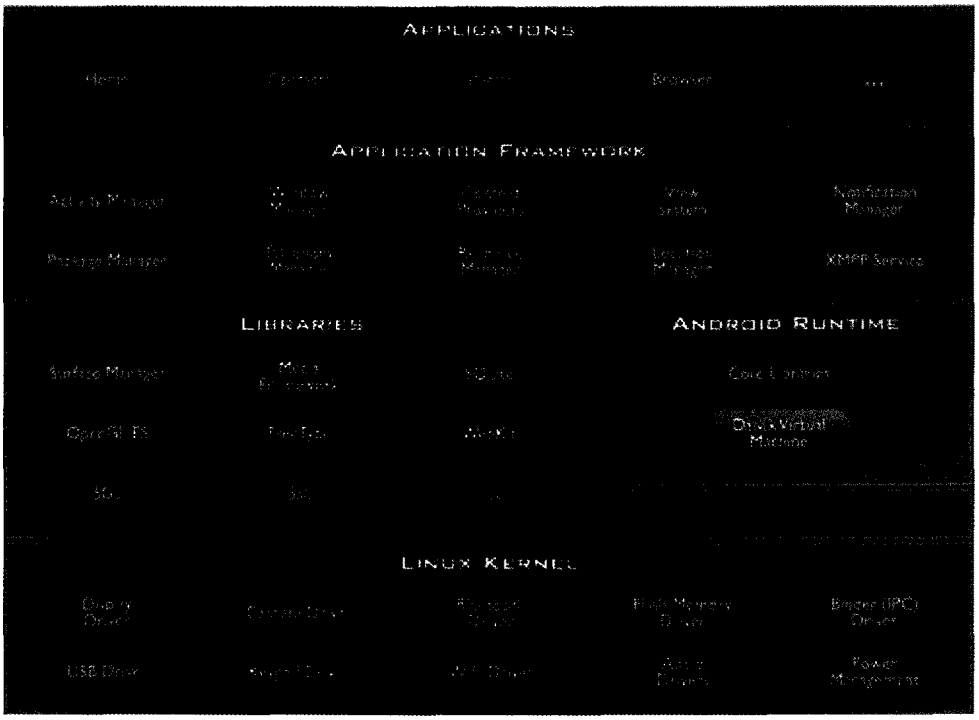


그림 1. 안드로이드 아키텍처

들은 이런 사항을 알 수 있다. 시스템 C 라이브러리, 미디어 라이브러리, Surface 관리자, LibWebCore, 2D 그래픽 엔진, 3D 라이브러리, 경량화된 관계형 데이터베이스 엔진 등이 라이브러리 형태로 제공된다.

- System C library : 임베디드 리눅스 기반 기기를 위한 표준 C 시스템 라이브러리(libc)의 BSD 상속 구현체
- Media Libraries : PacketVideo의 OpenCORE 기반이며, 인기 있는 오디오 및 비디오 포맷, MPEC4 / H.264 / MP3 / AAC / AMR / JPG / PNG를 포함하는 정적 이미지 파일의 재생 및 녹음(녹화)
- Surface Manager : 디스플레이 서브시스템 및 다수의 응용프로그램의 2D, 3D 그래픽 레이어
- LibWebCore : 안드로이드 브라우저 및 Embeddable 웹 뷰와 같은 최신의 웹 브라우저 엔진
- SGL : 2D graphics 지원
- 3D libraries : OpenGL ES 1.0 API 기반를 기반으로 하며, 하드웨어 3D 가속 또는 최적화된 3D S/W rasterized
- FreeType : 비트맵과 벡터 폰트 렌더링
- SQLite : 모든 응용프로그램에서 사용 가능한 강력하고 경량인 관계형 데이터베이스 엔진

2.4 안드로이드 런타임(Android Runtime)

모든 안드로이드 응용프로그램은 각자의 프로세스 상에서 실행되며, 고유의 Dalvik 버추얼 머신의 인스턴스를 가지고 있다. Dalvik은 모바일 기기가 다수의 버추얼 머신에서 효율적으로 실행될 수 있도록 한다. Dalvik은 최소의 메모리 영역에 최적화된 Dalvik Executable(.dex) 포맷 파일을 실행시킨다. 버추얼 머신은 레지스터 기반이

며, 자바 컴파일러로 컴파일된 클래스들을 "dk"툴을 이용하여 .dex 포맷으로 변경한 클래스들을 실행한다. Dalvik 버추얼 머신은 스레딩과 저수준 메모리 관리와 같은 리눅스 커널 기능을 사용한다.

2.5 리눅스 커널(Linux Kernel)

안드로이드 플랫폼은 보안, 메모리 관리, 프로세스 관리, 네트워크 관리, 드라이버 모델 등의 핵심 서비스를 리눅스에 기초하여 구현 되었다. 이 리눅스 커널은 하드웨어와 나머지 소프트웨어 스택 간의 추상화된 계층 역할을 한다.

3. 안드로이드 어플리케이션 구조

안드로이드 어플리케이션은 Activity, Intent Receiver, Service, Content Provider 등 4가지로 구성된다. 모든 어플리케이션이 4가지 구성요소 모두를 갖아야 하는 것은 아니고, 이들의 조합으로 이루어진다. 예를 들어 어플리케이션 A는 Activity로만 구성되고, 어플리케이션 B는 Activity와 서비스로 구성된다. 물론 어플리케이션 C는 4가지 구성요소 모두로 이루어질 수도 있다. 어플리케이션에서 어떤 컴포넌트들을 사용할지 결정하였다면 이 컴포넌트들의 목록을 AndroidManifest.xml 파일에 기록해야 한다. 이 xml 파일은 어느 어플리케이션에서 이 컴포넌트들을 선언했는지, 그들의 기능과 요구 사항은 무엇인지를 기록하는 파일이다[5,6].

3.1 안드로이드 어플리케이션 구조

(1) Activity

Activity는 어플리케이션에서 하나의 화면을 지칭한다. 각 Activity는 Activity 베이스 클래스를 상속하여 구현한 하나의 클래스이며, 사용자에게 View와 Event 응답으로 이루어진 인터페이스를 화면상에 보여준다. 예를 들어 텍스트 메시징 어플리케이션은 첫 화면은 contact 목록을 보여줄 것이고, 두 번째 화면은 선택한 contact에게 메시지를 사용하도록 하고, 나머지 화면은 보낸 메시지를 확인하거나 환경설정을 바꾸는 화면이 될 수 있다. 이러한 화면이 Activity가 되며, 다른 화면으로의 이동은 새로운 Activity를 시작하는 것과 같다. 어떤 상황에서는 Activity가 이전의 Activity로 값을 보내줄 수도 있다. 예를 들면 사용자가 사진을 선택했을 경우를 이전의 Activity에 선택한 사진을 보내주는 것과 같은 것이다.

화면이 오픈되면, 이전 화면은 멈추게 되며, history stack에 저장된다. 사용자는 history내에 있는 이전 화면으로 돌아갈 수 있다. 화면들은 history stack내에 저장될 필요가 없는 경우 삭제될 수 있으며, 안드로이드는 이 history stack을 home screen으로부터 실행시킨 각 어플리케이션마다 유지하게 된다.

안드로이드 Activity 라이프 사이클은 MIDlet과 유사하게 라이프 사이클 관리를 위한 8개의 메소드를 제공한다. 각각의 메소드들은 다음과 같다[5].

onCreate() onStart() onRestart() onResume()
onFreeze() onPause() onStop() onDestroy()

Activity가 시작되고 실행 중에 있을 때 다른 Activity가 시작되면 그 때까지 실행되던 Activity는 pause 상태로 전환되어야 한다. pause 상태로 존재하다가 다른 어플리케이션에서 메모리를 많이 요구하게 되면 pause 상태에 있던 Activity는 죽을(killed) 수도 있다. 만약 pause 상태에 있던 Activity가 사용자의 선택에 의해 다시 포그라운드로 돌아가게 되면 resume 상태가 된다. 또 pause 상태에 있던 Activity가 더 이상 보이지 않

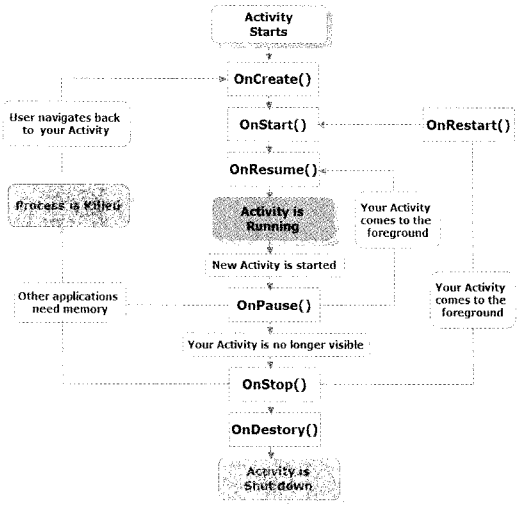


그림 2. Activity 라이프사이클

으면 stop 상태로 전환된다. stop 상태에 있던 Activity가 다시 포그라운드로 되면 restart에 의해 start 상태로 돌아간다. 물론 stop 상태에 있던 Activity도 다른 어플리케이션이 메모리를 요구할 때 죽을 수도 있다. 또는 destory 상태로 전환될 수도 있다. 위와 같이 한 Activity가 시작되고 멈추어지고 다시 시작되고 그리고 결국은 죽는 과정을 Activity 라이프 사이클이라고 한다.

(2) Intent와 Intent Filter

안드로이드는 화면과 화면 사이를 이동할 때, Intent라는 특별한 클래스를 사용한다. Intent는 어플리케이션이 무엇을 하기를 원하는지에 대한 정보를 담고 있다. A Activity에서 B Activity로 화면 전환이 이루어질 때 Intent 클래스가 사용된다. Intent 클래스는 다음과 같이 사용된다.

```
Intent i = new Intent(this, NoteEdit.class);
startActivityForResult(i, ACTIVITY_CREATE);
```

호출하는 측은 this 이고 호출당하는 측은 NoteEdit 클래스이다. this 클래스에서 NoteEdit 클래스의 화면으로 전환이 이루어진다. 화면 전환은 startActivity(), startActivityForResult() 메소드를 통해 이루어진다.

Intent의 가장 중요한 부분은 action과 data가 어떤 동작을 할 것인지에 대한 자료구조이다. action의 일반적인 값은 MAIN(activity의 시작점), VIEW, PICK, EDIT 등이며, 데이터는 URI로 표현된다. 예를 들어 어떤 사람의 연락처를 보려고 할 때 VIEW action에 대한 intent를 생성하고 그 사람을 표현하는 URI data를 만들어 VIEW action 데이터로 지정해주면 된다.

이와 관련한 클래스로는 IntentFilter가 있다. Intent가 어떤 작업에 대한 요청이라면 IntentFilter는 Activity 또는 Intent Receiver 상의 Intent를 핸들링 할 수 있는지를 표현하는 것이다. 예를 들어 어떤 사람에 대한 연락처를 보여줄 수 있는 Activity가 사람에 대한 정보를 표현하는 데이터에 적용될 때 VIEW 액션을 핸들링 할 수 있는 IntentFilter를 생성한다. 또한 Activity는 AndroidManifest.xml 파일에 기록된 IntentFilter를 생성한다.

화면과 화면 사이를 이동하는 것은 Intent를 해석하는 것으로 동작된다. 앞으로(forward) 이동하려면 Activity는 startActivity(myIntent)를 호출하고, 그 다음에 시스템은 어플리케이션에 설정된 모든 IntentFilter를 검사하고, myIntent와 가장 일치하는 IntentFilter를 가진 Activity를 가져오며, 이 Activity의 시작을 Intent로부터 통보받는다. Intent 해석 과정은 startActivity가 호출될 때 런타임으로 일어난다. 이러한 과정은 첫째, Activity는 다른 컴포넌트들이 간단하게 Intent를 요청하는 것만으로 간단하게 재사용 가능하고, 둘째, 동일한 IntentFilter를 가진 새로운 Activity로 Acitivity는 언제든지 교체 가능하다는 두 가지 장점을 지닌다.

(3) Intent Receiver

IntentReceiver는 작성한 어플리케이션 코드 내에서 핸드폰으로 전화가 걸려오거나, 데이터 네트워크 접속이 활성화되는 것과 같은 외부 이벤트를 처리하는데 사용된다. IntentReceiver는 UI를 그려주는 것은 아니고, NotificationManager를 이용하여 사용자에게 어떤 일이 발생했다는 것을 알려준다. IntentReceiver도 AndroidManifest.xml 파일에 등록되지만, Context.registerReceiver()를 이용하여 코드 상에서 등록할 수 있다. 어플리케이션이 호출된 IntentReceiver에 의해 실행할 수 없을 때는 IntentReceiver가 트리거되면 필요에 따라 시스템이 어플리케이션을 실행한다. 또한 어플리케이션은 Context.bradcastIntent()를 호출하여 자신의 Intent를 다른 어플리케이션에 broadcast할 수 있다.

(4) Service

Service는 UI와 상관없이 아주 오랫동안 존재하며 실행되는 코드이다. 예를 들면 재생 목록에서 노래를 재생하는 미디어 플레이어 같은 것이다. 미디어 플레이어 어플리케이션은 사용자가 곡을 선택하고 재생을 시작하게 하는 하나 이상의 Activity를 가지고 있지만, 스스로 음악 재생을 하는 것은 Activity에 의해 실행되는 것은 아니다. 사용자는 새로운 화면으로 이동하고 나서도 음악을 계속 재생하기를 기대하기 때문이다. 이런 상황에서 미디어 플레이어의 Activity는 Context.startService()를 이용하면 Service로 실행된다. 시스템은 음악 재생 서비스를 멈출 때까지 계속 재생할 것이다. Context.bindService() 메소드는 서비스에 연결하거나 아직 시작하지 않은 Service를 시작한다. Service에 연결이 되면 Service에 접근 가능한 인터페이스를 통해 멈춤, 다시 재생 등을 할 수 있도록 접근 가능하다.

(5) Content Provider

어플리케이션은 자신의 데이터를 SQLite 데이터베이스 또는 다른 방법으로 파일에 저장할 수 있다. Content Provider는 어플리케이션 데이터가 다른 어플리케이션과 공유할 필요가 있을 때 아주 유용하다. 이 클래스는 다른 어플리케이션이 데이터를 저장하거나, 가져오는 것과 같은 작업을 할 수 있다.

3.2 안드로이드 어플리케이션 라이프사이클

안드로이드 어플리케이션은 자신 고유의 리눅스 프로세스에 의해 실행된다. 이 프로세스는 어플리케이션이 특정 코드를 수행할 필요가 있을 때 생성되고, 시스템이 다른 어플리케이션에 사용할 메모리를 요청 또는 더 이상 필요 없어질 때까지 실행 상태를 유지한다.

안드로이드의 중요한 특성은 어플리케이션 프로세스 라이프사이클이 어플리케이션 자신에 의해 직접 컨트롤되지 않는다는 것이다. 그것은 시스템이 실행 상태를 알고 있고, 어플리케이션이 얼마나 사용자에게 중요한 것이고, 시스템의 여유 메모리가 얼마나 더 남아있는가 등으로 결정한다. 어플리케이션 개발자들이 알아야 할 중요한 사항은 어떻게 서로 다른 어플리케이션 컴포넌트(Activity, Service, IntentReceiver)들이 어플리케이션의 프로세스에 영향을 미치는가 하는 것이다. 이러한 컴포넌트들을 정확하게 사용하지 않을 경우, 시스템은 중요 어플리케이션 프로세스를 종료시킬 수 있다.

유휴 메모리가 부족할 때 어떤 프로세스가 종료되어야 하는지 결정하는 것은 중요하다. 안드로이드는 이것을 Importance hierarchy를 베이스로 하고, 실행중인 컴포넌트와 그 컴포넌트들의 상태에 따라 중요도를 결정한다. 중요도의 순서는 다

음과 같다.

1) foreground process : 사용자가 조작 중인 최상위 화면의 Activity(onResume() 메소드가 호출된 경우) 또는 현재 실행중인 IntentReceiver (onReceiveIntent() 메소드가 실행 중)를 잡고 있는 프로세스이다. 시스템에는 아주 적은 수의 이런 프로세스가 있고, 이러한 프로세스가 실행되기에 극히 부족한 메모리가 있을 경우에만 종료된다. 이러한 동작은 일반적으로 기기가 메모리 페이징 상태에 다다랐을 때이며, 사용자 인터페이스 응답을 유지하기 위해서 취해진다.

2) visible process : 사용자 화면에는 보이지만 foreground가 아닌(onPause() 메소드를 호출한 경우) Activity를 잡고 있는 프로세스이다. 예를 들면 foreground activity가 다이얼로그를 보여주며 이전 activity는 그 아래에 위치하는 것과 같다. 이런 프로세스는 매우 중요하며, 모든 foreground 프로세스 실행이 종료되기 전까지는 종료되지 않는다.

3) service process : startService()를 호출한 Service를 잡고 있는 프로세스이다. 이러한 프로세스는 사용자에게 직접적으로 보이는 것은 아니지만 사용자와 관련된 일반적인 일(mp3를 배경음으로 연주한다던가, 네트워크 데이터를 업로드/다운로드 한다던가 하는 등)들을 한다. 그래서 이러한 프로세스는 모든 foreground, visible 프로세스를 유지하기에 메모리가 충분하지 않을 때까지 계속 유지된다.

4) background process : 사용자에게 현재는 보이지 않는 activity(onStop() 메소드를 호출한 경우)를 잡고 있는 프로세스이다. 이런 프로세스들은 사용자에게 어떠한 영향을 주지 않으며, activity life cycle을 정확하게 구현하기 위해서 제공된다. 시스템은 메모리가 부족할 경우 언제든지 이런 프로세스 수행을 멈출 수 있다. 가끔씩 이러한 프로세스들이 많이 수행되고 있을 때, 메모리 부족 시 가장 최근에 사용자에게 보인 것이 가장 마지막으로 종료되도록 LRU(Least Recently Used) 목록에 유지된다.

5) empty process : 어떠한 활성화된 어플리케이션 컴포넌트들도 잡고 있지 않는 프로세스이다. 이러한 프로세스를 유지하고 있는 이유는 다음에 컴포넌트의 어플리케이션이 시작되어야 할 때 시작 시간을 개선하기 위한 캐시로 사용하기 위함이다. 시스템은 캐쉬된 빈 프로세스와 커널 캐쉬 사이의 시스템 전체 리소스의 균형을 맞추기 위해 이런 프로세스를 종료시킨다.

3.3 안드로이드 UI

안드로이드의 UI는 View와 ViewGroup 클래스로 구성한다. Activity가 UI를 화면상에 나타내기 위해서는 안드로이드 플랫폼에서 UI를 표현하는 기본 단위인 View와 ViewGroup이 함께 사용되어야 한다. View 클래스는 화면상에서 직사각형 영역의 레이아웃과 내용을 저장하는 자료구조로 화면에서 정렬, 그리기, 포커스 이동, 키의 동작 제어를 담당하는 메소드를 제공한다. ViewGroup은 자신에게 포함된 View와ViewGroup을 제어하는 기능을 갖고 있으며 이를 통하면 어떠한 복잡한 UI도 만들 수 있다. 안드로이드에서는 모든 UI를 JAVA뿐 아니라 XML로도 디자인할 수 있도록 하여 UI 디자인 효율성을 높이고 있다[7].

(1) View

View는 android.view.View base 클래스의 객체이다. 이것은 화면상에서 특정한 직사각형 영역의 레이아웃과 내용을 저장하는 자료 구조이다. View 객체는 화면상의 특정 영역 내에서 정렬,

레이아웃, 그리기, 포커스 이동, 스크롤링, 키 등을 다룬다.

View 클래스는 위젯(widget : 상호작용하는 화면 요소를 그리도록 구현된 서브 클래스)을 위한 베이스 클래스이다. 위젯은 자신만의 영역과 화면상에 보이는 그래픽 요소를 가지고 있으며, 위젯을 이용하여 UI를 빠르게 만들어낼 수 있다. 위젯에는 Text, EditText, InputMethod, MovementMethod, Button, RadioButton, Checkbox, ScrollView등이 있다.

(2) ViewGroup

Viewgroup은 android.view.Viewgroup 클래스의 객체이다. 이 클래스의 이름이 말해주듯이 자신이 포함하는 하위의 View와 ViewGroup을 제어하는 기능을 가진 Composite 패턴의 특별한 View Object이다. 휴대폰 화면에는 하나의 View만 있는 것이 아니라 여러 개의 뷰가 다양한 형태로 존재한다. Viewgroup은 UI의 구조를 만들어주고, 복잡한 화면 요소들을 만들어 하나의 엔티티로 취급할 수 있도록 한다. Viewgroup 클래스는 레이아웃(layout : 화면 레이아웃의 공통적인 틀을 제공할 수 있도록 구현된 클래스)의 베이스 클래스이다.

(3) Tree-Structured UI

안드로이드 플랫폼에서는 그림 3의 다이어그램처럼 View와 Viewgroup 노드의 트리구조를 사용하여 Activity의 UI를 정의한다. 트리 구조의 UI는 안드로이드에서 제공하는 위젯과 레이아웃 또는 직접 제작한 View를 이용하여 구성된다. 어떻게 구성하느냐에 따라 트리는 간단할 수도 복잡할 수도 있고, 안드로이드에서 제공하는 위젯과 레이아웃 또는 직접 제작한 view를 이용하여 구성할 수 있다.

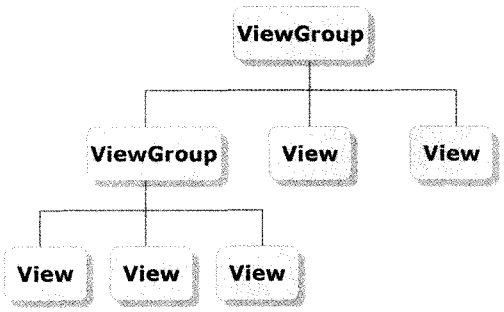


그림 3. 트리구조의 사용자 인터페이스

이러한 트리 구조를 화면상에 그려주기 위해서는 Activity가 setContentView() 메소드를 호출하고 root 노드 객체를 이 메소드에 넘겨주면 된다. Activity가 활성화 되고 포커스를 받으면, 시스템은 Activity에 root 노드를 통해 화면상에 그리는 것을 요청하게 되며, root 노드는 하위의 노드를 그릴 것을 요청한다.

각 Viewgroup은 가용한 공간을 계산하고, 그 하위의 View를 배치하며 각 하위의 View에서 Draw() 메소드를 호출하여 스스로 그릴 수 있도록 한다. 각 하위의 View는 사이즈와 그려질 위치를 부모로부터 요청하며, 부모는 최종적으로 얼마나 크게, 어디에 그려질 수 있는가를 결정한다.

(4) LayoutParams

모든 Viewgroup 클래스는 ViewGroup.LayoutParams를 확장한 클래스를 이용한다. 이 서브클래스는 하위의 View 사이즈와 위치, Viewgorup 클래스의 특성들을 포함하고 있다. 모든 LayoutParmas의 서브클래스는 값을 설정하는 각자의 방법을 가지고 있다. 각 하위 요소들은 반드시 부모의 특징에 맞는 LayoutParams를 정의해야 한다. LayoutParams 클래스는 뷰 객체들이 어떻게 그려져야 하는 지에 대한 정보를 부모 객체에게 알려주는 역할을 한다. 각 객체들의 폭과 높이를 명시할 때 다음 둘 중 하나로 표시된다[8].

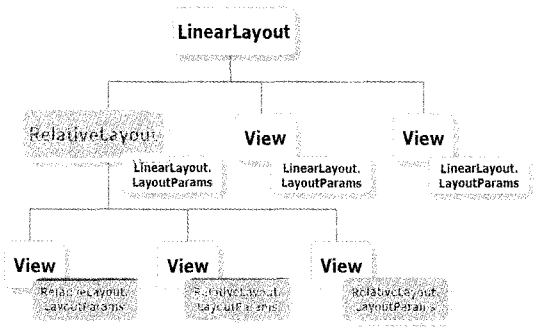


그림 4. 각 View와 ViewGroup은 부모 속성에 맞게 LayoutParams를 정의

- WRAP_CONTENT: 콘텐츠를 표시하는데 필요한 최소한의 크기로 그린다.
- FILL_PARENT: 부모 객체와의 패딩 여백을 제외한 나머지 모든 공간을 차지한다.

모든 Viewgroup은 가로(width)와 세로(height), 마진(margin), 테두리(border)를 가지고 있으며 이를 사용자가 원하는 크기로 바꿀 수 있다. 위 그림에서 루트 노드가 LinearLayout이므로 3개의 자식 노드는 선형으로 배치한다. 자식 노드 중 첫 번째 노드는 다시 RelativeLayout이므로 3개의 자식 노드들은 상대적으로 배치한다.

(5) XML을 이용한 화면 디자인

화면디자인을 하드코딩으로 하는 것은 굉장히 귀찮은 작업이기 때문에 안드로이드는 화면디자인을 위해 XML 문법을 지원한다. 안드로이드는 특정 안드로이드 View의 서브클래스를 나타내는 다양한 화면 구성 요소들을 지원한다. 어플리케이션의 res/layout/ 디렉토리에 XML 파일을 생성하여 디자인 한다.

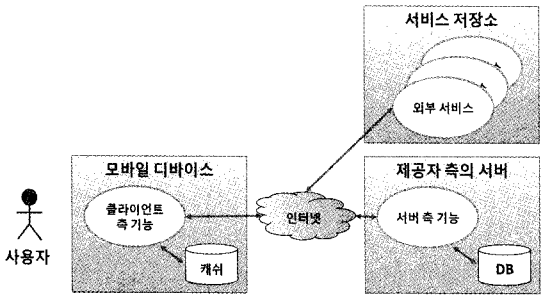
각각의 XML 파일들은 하나의 android.view.View 요소를 정의하며, 이것은 간단한 비쥬얼 요소일 수도 있고, 자식 객체들의 집합을 포함하는 레이아웃일 수 있다. 안드로이드가 어플리케이션을 컴파일할 때 Activity.onCreate() 구현 내에서 setContnetView(R.layout.layout_ file_name) 호출을 통해 로드할 수 있는 각각의 android.view.View 리소스를 컴파일 한다. 그리고 각 XML 파일은 안드로이드 GUI 클래스와 일치하는 태그들로 구성된다. 이 태그들은 각 클래스들의 메소드와 대부분 일치하는 속성들을 가지고 있다. 예를 들면 EditText는 EditText.setText와 일치하는 text 속성을 가지고 있다.

각 클래스와 메소드 이름, 요소와 속성이름이 비슷하더라도 항상 1:1로 정확하게 일치하지 않는다. 안드로이드는 XML에 나타나있는 순서대로 각 요소들을 그린다. 그러므로 각 요소들이 겹칠 경우 XML 파일의 마지막 요소는 항상 최상위에 그려지고, 각 XML 파일은 View 또는 ViewGroup 객체를 최상위로 하는 트리구조로 컴파일되며, 반드시 단일 루트 태그를 포함한다.

4. 서비스 기반 모바일 어플리케이션(SMA)

안드로이드 SMA는 모바일 디바이스의 장점을 부각시키고 단점을 보완하기 위하여 제안된 응용형태이다. SMA는 사용자가 필요로 하는 기능의 일부는 서버 측에 배포하고, 모바일 디바이스에 설치된 클라이언트 어플리케이션과의 네트워크를 통한 상호작용을 통하여 전체 기능을 실행한다. 아래 그림은 서비스 기반 모바일 어플리케이션의 전체 구조를 보여준다. 사용자가 필요로 하는 전체 기능 및 데이터베이스는 모바일 디바이스에 설치된 클라이언트 측과 서비스 제공자 측에 설치된 서버 측에 분리되어 있다. 전체 어플리케이션 기능 중 비교적 적은 자원을 필요로 하는 간단한 기능은 클라이언트 측에서 실행이 되며, 복잡한 계산 및 데이터 조작을 요구하는 기능은 서버 측에서 실행이 된다. 더욱이 공통적이고 재

사용 가능한 기능들은 서비스 형태로 제공하며, 이는 클라우드 컴퓨팅 서비스와 유사하다. 서비스는 모바일 어플리케이션 제공자가 직접 개발하여 제공하거나, 구글 Map 서비스와 같이 제3의 서비스 제공자로부터 구독하여 사용할 수 있다.



SMA는 다양하고 풍부한 네트워크를 이용하여 독립형 모바일 어플리케이션의 기능 제약성을 보완할 수 있다. 즉, 성능 좋은 서버에서 제공되는 서비스를 사용함으로써, 모바일 디바이스의 특징인 부족한 컴퓨팅 자원을 확장해서 복잡한 기능의 어플리케이션을 사용할 수 있게 된다. 그러나 기능 중 일부는 서비스 측에서 실행되기 때문에, 네트워크 안정성이 보장이 되어야만 사용자가 기능 호출에 대한 응답을 받을 수 있다. 그리고 모든 기능 호출은 반드시 네트워크 통신을 필요로 하기 때문에 독립형 모바일 어플리케이션만큼 빠른 시간 내에 응답을 받지는 못한다.

5. 안드로이드 마켓

안드로이드 마켓은 구글이 운영하고 있고, 안드로이드 어플리케이션을 다운로드할 수 있는 서비스이다. 애플의 앱스토어와 다르게 안드로이드 마켓은 구글의 정책에 따라 사용자가 구입한 지 24시간 내에 다운로드한 어플리케이션의 환불을 요구하면 구매금액의 전액을 환불해준다. 콘텐츠 유통 방식은 이동통신사의 검수 과정이 없고, 누구나 콘텐츠를 업로드할 수 있다. 검수 과정이 없거나 또는 간략화 되면 질 낮은 어플리케이션이 유통될 수 있는 단점도 있으나 많은 콘텐츠가 사용자들에게 다가갈 수 있다는 장점도 있다[9].

시장 조사 업체 스트래티지 어낼리틱스(SA)는 최근 발간한 보고서에서 안드로이드가 올해 휴대폰 제조업체, 개발자, 이동통신 서비스 업체들의 지원을 등에 업고 무려 900%까지 성장할 것으로 전망했다. 그래서 2012년경에는 아이폰을 추월할 것으로 전망하고 있다. 이는 안드로이드의 개방성에 기인한다. 애플의 OS X와 그 위에서 동작하는 아이폰은 애플에서만 제조할 수 있어 다른 업체들이 참여할 기회가 크지 않다. 모든 수익이 애플에게만 돌아가는 구조이어서 다른 업체들이 안드로이드 진영에 참여할 가능성이 크다.

아무튼 모바일 콘텐츠 개발자들에게는 유료로 어플리케이션을 판매할 수 있는 또 다른 공간이 생기는 것이고, 이에 또 다른 기회가 펼쳐지고 있다. 안드로이드 마켓이 활성화 되면 앱스토어에서 서비스되고 있는 많은 어플리케이션들이 안드로이드 마켓에서도 판매될 것으로 예상된다. 모바일 콘텐츠 시장은 애플의 앱스토어와 구글의 안드로이드 마켓이 큰 흐름을 만들어 나가고 있는 것이 사실이다. 표 1은 두 시장을 비교하고 있다.

아이폰 SDK는 애플에 개발 업체로 등록해야 SDK를 다운로드 받을 수 있는 반면 안드로이드 SDK는 누구나 다운로드 받을 수 있다. 두 시장 모두 개발자는 판매 금액의 약 70%를 수익으로 가져가고, 나머지 30%는 애플의 경우 수수료 명목으로 애플이 징수하고 있으며, 안드로이드 마켓에서는 이동통신사가 30%를 수취하게 된다. 구글은 별도로 이익을 취하지 않고 있다. 이익을 추구하는 기업 입장에서는 용납되지 않는 일이다. 대

표 1. 앱스토어와 안드로이드 마켓의 비교

비교대상	애플의 앱스토어	구글의 안드로이드마켓
서비스 시기	2008년 7월 11일	2008년 8월 28일
운영주체	애플	자율적으로
어플리케이션 등록	애플의 허락 하에	자율적으로
SDK 다운로드	등록해야만 가능	누구나 가능
수익분배	CP와 애플이7:3	CP와 이통사 또는 솔루션업체가7:3
폰출시시기	아이폰 2007년6월2일	구글폰 2008년10월22일

신 구글은 수익을 창출할 수 있는 방법을 다른 곳에서 찾고 있다. 구글은 안드로이드 마켓을 활성화시켜 모바일 광고 시장을 통해 수익을 얻겠다는 전략을 가지고 있다.

두 시장의 가장 큰 차이점은 애플의 앱스토어는 애플의 통제 하에 운영되고 있고 안드로이드 마켓은 사용자에게 열려있어 자율적으로 운영된다는 것이다. 따라서 애플의 게시 조건을 만족시키지 못하면 그 어플리케이션은 앱스토어에서 판매될 수 없다.

두 회사가 사용한 용어에서도 미세하게 차이를 느낄 수 있다. 애플은 상점이라는 뜻인 Store란 용어를 사용하고 있고 구글은 장터의 의미를 갖는 Market이란 용어를 사용하고 있다. 마켓이라는 용어를 선택한 구글이 좀 더 열린 시장이라고 생각된다. 안드로이드 마켓이 좀 더 개방적이라고 생각되는 사례는 또 있다. 애플에서는 인터넷 전화(VoIP)를 이동통신사와의 관계 때문에 허락하지 않고 있다. 반면 구글에서는 인터넷 전화를 이용할 수 있는 어플리케이션이 등록되어 있는데 이 어플리케이션을 이용하면 비싼 전화 대신에 값싼 인터넷 전화를 이용할 수 있다.

5. 결 론

IDC에 따르면 2010년 1분기 세계 휴대폰 시장 출하 대수는 294,900천대로 시장이 위축되었던 2009년 1분기의 242,400천대에 비해 21.7% 증가했고, 이는 스마트폰에 대한 수요 증가로 분석하고 있고, 특히 구글폰이 2010년을 기점으로 폭발적으로 증가하고 있는 점으로 보아 안드로이드 플랫폼을 조사 분석하는 것이 중요하다. 안드로이드 플랫폼에서 사용하는 '어플리케이션 프레임워크', 어플리케이션을 거래하는 온라인 공간 '안드로이드 마켓'을 분석하는 것은 의미 있다고 판단한다.

안드로이드 플랫폼은 아주 단순하게 생각하면 PC 위에 돌아가는 Windows와 같은 운영체제라고 생각해도 된다. 안드로이드는 리눅스 커널 위에서 동작하며, 포괄적 라이브러리 세트, 풍부한 멀티미디어 사용자 인터페이스, 폰 어플리케이션 등을 제공한다. 안드로이드는 개발자들이 자바 언어로 어플리케이션을 작성 할 수 있으며, 컴파일된 바이트코드를 구동할 수 있는 런타임 라이브러리를 제공한다.

구글은 OHA 정신에 입각하여 안드로이드 출시 후 안드로이드 소스를 공개 하였다. 또한 SDK를 공개하여 안드로이드 환경에서 동작할 수 있는 소프트웨어를 개발할 수 있게 하였다. 마지막으로 이것을 유통시킬 수 있는 안드로이드 마켓 또한 오픈하여 다양한 소프트웨어를 공급할 수 있게 하였다.

이러한 변화는 소비자가 콘텐츠를 개발하여 수

익을 올리는 즉, 소비자가 바로 생산자가 되는 프로슈머(prosumer)로서의 앱스토어 시장이 본격적으로 열리고 있는 점이다. 안드로이드 폰의 급속한 보급 확대는 대기업과 하드웨어 중심이 아닌 1인 기업, 벤처, 소프트웨어 중심의 시장 형성을 전개시키고 있다.

참 고 문 헌

[1] http://developer.android.com/guide/basics/what-is-android.html.
[2] http://www.openhandsetalliance.com/oha_overview.html.
[3] Tergujeff, R., Haajanen, J., Leppanen, J., and Toivonen, S., "Mobile SOA: Service Orientation on Lightweight Mobile Devices," In Proceedings of 2007 IEEE International Conference on Web Services(ICWS 2007), pp. 1224-1225, 2007.
[4] 안드로이드 SDK, http://developer.android.com/sdk/index.html.
[5] http://developer.android.com/guide/topics/fundamentals.html.
[6] http://www.kandroid.org/.
[7] http://developer.android.com/guide/topics/ui/index.html.
[8] http://developer.android.com/guide/topics/ui/declaring-layout.html.
[9] 안드로이드마켓, http://www.android.com/market/.

김 평 중

- 1985년 충남대학교 계산통계학과(학사)
- 1995년 한국과학기술원 전산학과(석사)
- 2000년 충남대학교 컴퓨터과학과(박사)
- 2004년~2005년 Wright State Univ. Post-Doc.
- 1995년 전자계산기조직응용기술사
- 2008년 재난관리지도사
- 1987년~1988년 포항종합제철 전산기술직원
- 1988년~1998년 한국전자통신연구원 선임연구원
- 1998년~현재 충북도립대학 컴퓨터정보과 교수
- 관심분야 : 이동에이전트, 네트워크멀티미디어, 재난정보통신