

초보 개발자를 위한 안드로이드 스튜디오 입문

모바일 생활을 더욱 편리하게 만들 참신한 발상이 떠올랐을 때 개발자에게 가장 필요한 것은 아이디어를 손에 잡히는 앱으로 변환해 주는 개발 도구일 것이다. 구글이 공식 지원하는 통합개발환경 안드로이드 스튜디오는 구글 클라우드 플랫폼과 연계되어 속도와 개발자 편의성 등을 크게 높였고, 풍부한 안드로이드 개발 생태계 조성에 중요한 역할을 하고 있다. 기본 설정부터 실제 작동하는 앱을 빌드하기까지 튜토리얼을 따라가면서 안드로이드 스튜디오에 익숙해지고, 더욱 자유롭게 아이디어를 펼쳐보자.



- ▣ 1부 | 안드로이드 스튜디오 설치와 설정 과정
- ▣ 2부 | 세부 내용 알아보고 앱 코딩 시작하기
- ▣ 3부 | 앱 빌드 후 실행하기



안드로이드 스튜디오 설치와 설정 과정

Jeff Friesen | JavaWorld

최근 몇 년 동안의 추세를 볼 때 현재 모바일 운영체제 시장은 안드로이드가 지배한다고 해도 과언이 아니다. 자바 기반의 안드로이드 운영체제는 새로운 디지털 골드 러쉬를 일으켰고 모바일 앱에서 기회를 얻으려는 프로그래머들이 앞다투어 안드로이드로 모여드는 중이다. Indeed.com에서 구인 검색을 해 보면 안드로이드와 관련된 일자리가 풍부하다는 것을 알 수 있다.

성공적인 안드로이드 개발자가 되기 위해서는 자바, 안드로이드 API(application programming interface), 안드로이드 앱 아키텍처에 대한 이해가 필요하다. 또한, 적절하고 효과적인 개발 환경을 사용하는 것도 필수적이다. 한때 이클립스(Eclipse) 통합개발환경(IDE)과 ADT(Android Development Tool) 플러그인이 안드로이드 앱 개발에 쓰인 대표적인 플랫폼이었고, 현재의 대세는 안드로이드 스튜디오다.

이 강좌는 안드로이드 스튜디오를 처음 접하는 사람을 위한 입문 과정이다. 우선 스튜디오를 간략히 소개한 후 다운로드, 설치, 실행하는 방법을 살펴보고, 그다음 대부분의 시간을 실제 안드로이드 스튜디오를 사용해서 움직이는 모바일 앱을 개발하는 데 할애할 것이다. 1부에서는 안드로이드 스튜디오에서 첫 안드로이드 프로젝트를 시작하고 프로젝트 작업 공간을 살펴본다. 2부에서는 앱을 코딩하면서 안드로이드 스튜디오에서 소스 코드와 리소스를 프로젝트에 입력하는 방법을 배운다. 마지막으로 3부에서는 에뮬레이션 하드웨어 기기와 실제 아마존 킨들 파이어 HD 7" 태블릿을 모두 사용해서 앱을 빌드하고 실행해 볼 것이다.

안드로이드 스튜디오 시작하기

안드로이드 스튜디오는 구글이 공식적으로 지원하는 안드로이드 앱 개발 IDE로, 인텔리J(IntelliJ) IDE를 기반으로 하며 아파치 라이선스 2.0에 따라 무료 배포된다. 2016년 9월 현재 최신 안정화 버전은 2.1.1이며 다음과 같은 기능을 포함한다.

- 모든 안드로이드 기기를 대상으로 개발이 가능한 통합 환경
- 안드로이드 TV 앱과 안드로이드 웨어 앱 빌드 지원
- 일반적인 안드로이드 디자인과 구성 요소를 만들기 위한 템플릿 기반 마법사
- 사용자 인터페이스 구성 요소를 끌어서 놓을 수 있고 여러가지 화면 구성에서 레이아웃을 미리볼 수 있는 레이아웃 편집기
- 안드로이드별 리팩토링 및 퀄 픽스
- 그래들(Gradle) 기반 빌드 지원
- 성능, 사용성, 버전 호환성 및 기타 문제를 잡아내기 위한 린트(Lint) 도구
- 프로가드(ProGuard) 통합 및 앱 서명 기능
- 빠르고 다양한 기능을 갖춘 에뮬레이터
- 새 APK(애플리케이션 패키지 압축 파일)를 빌드하지 않고도 실행 중인 앱에 변경 사항을 푸시하기 위한 즉시 실행(Instant Run) 기능
- 구글 클라우드 플랫폼을 기본 지원하므로 구글 클라우드 메시징 및 앱 엔진과 통합 가능
- C++ 및 NDK(Native Developer's Kit) 지원
- 플러그인을 통해 안드로이드 스튜디오를 확장하기 위한 플러그인 아키텍처

안드로이드 스튜디오 다운로드

구글은 윈도우, 맥OS X, 리눅스 플랫폼용 안드로이드 스튜디오를 제공하며 안드로이드 스튜디오 홈페이지에서 다운로드할 수 있다(다운로드 페이지에는 전통적인 SDK와 안드로이드 스튜디오 명령줄 도구도 있다). 안드로이드 스튜디오를 다운로드하기 전에 사용 중인 플랫폼이 다음 요구 사항 중 하나를 충족하는지 확인해야 한다.

윈도우 운영체제

- 마이크로소프트 윈도우 7/8/10(32비트 또는 64비트)
- 최소 2GB RAM(권장 8GB RAM)
- 최소 2GB 디스크 여유 공간(IDE용 500MB + 안드로이드 SDK와 에뮬레이터 시스템 이미지용 1.5GB). 권장 4GB
- 1280 x 800 이상의 화면 해상도

JDK 8

- 에뮬레이터 가속: 64비트 운영체제와 인텔 VT-x, 인텔 EM64T(인텔 64) 및 실행 비활성화(XD) 비트 기능을 지원하는 인텔 프로세서

맥 OS

- 맥OS X 10.8.5 ~ 10.11.4(엘 캐피탄)
- 최소 2GB RAM(권장 8GB RAM)

- 최소 2GB 디스크 여유 공간(IDE용 500MB + 안드로이드 SDK와 에뮬레이터 시스템 이미지용 1.5GB). 권장 4GB
- 1280 x 800 이상의 화면 해상도
- JDK 6

리눅스 운영체제

- **GNOME 또는 KDE 데스크톱**: 우분투 12.04, 프리사이즈 팡골린(Precise Pangolin, 32비트 애플리케이션을 실행할 수 있는 64비트 배포판)에서 테스트됨
- 32비트 애플리케이션을 실행할 수 있는 64비트 배포판
- GNU C 라이브러리(glibc) 2.11 이상
- 최소 2GB RAM(권장 8GB RAM)
- 최소 2GB 디스크 여유 공간(IDE용 500MB + 안드로이드 SDK와 에뮬레이터 시스템 이미지용 1.5GB). 권장 4GB
- 1280 x 800 이상의 화면 해상도
- JDK 8
- 에뮬레이터 가속: 인텔 VT-x, 인텔 EM64T(인텔 64) 및 실행 비활성화(XD) 비트 기능을 지원하는 인텔 프로세서 또는 AMD 가상화(AMD-V)를 지원하는 AMD 프로세서

사용 중인 운영체제가 안드로이드 2.1.1과 호환됨을 확인했으면 해당되는 안드로이드 스튜디오 배포 파일을 다운로드한다. 안드로이드 스튜디오 다운로드 페이지는 필자의 운영체제가 64비트 윈도우 8.1임을 자동으로 인식하고 **android-studio-bundle-143.2821654-windows.exe** 파일을 다운로드하도록 선택했다.

포함된 설치 프로그램과 안드로이드 SDK

android-studio-bundle-143.2821654-windows.exe에는 설치 프로그램과 안드로이드 SDK가 포함되어 있다. 설치 프로그램과 SDK가 없는 배포 파일을 선택해서 다운로드할 수도 있다.

64비트 윈도우 8.1에 안드로이드 스튜디오 설치하기

android-studio-bundle-143.2821654-windows.exe를 실행해서 설치 프로세스를 시작했다. 가장 먼저 그림 1과 같은 안드로이드 스튜디오 설치 대화 상자가 표시된다.



그림 1 | 안드로이드 스튜디오 설치

'다음(Next)'을 클릭하면 그다음 대화 상자로 진행되는데 여기서 안드로이드 SDK(설치 프로그램에 포함됨)와 안드로이드 가상 기기(AVD) 설치를 거부할 수 있는 옵션이 제공된다.

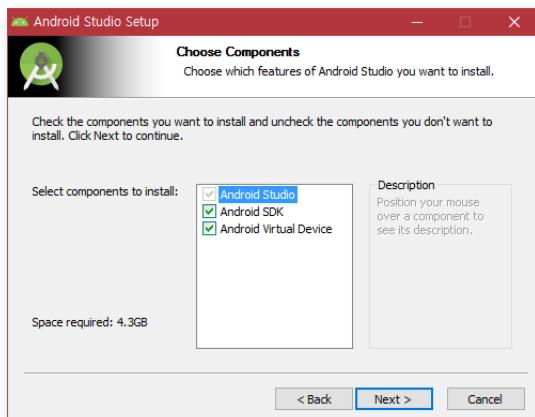


그림 2 | 안드로이드 SDK과 AVD 설치
여부 선택

이번 설치에서는 기본 설정을 그대로 유지했다. '다음(Next)'을 클릭하면 라이선스 동의를 확인하는 대화 상자로 진행된다. 라이선스에 동의하여 설치를 계속 진행한다.

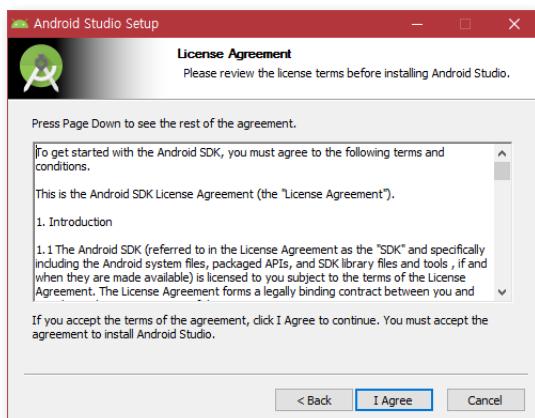


그림 3 | 라이선스에 동의해서 설치 진행

다음 대화 상자에서는 안드로이드 스튜디오와 안드로이드 SDK의 설치 위치를 변경할 수 있다.

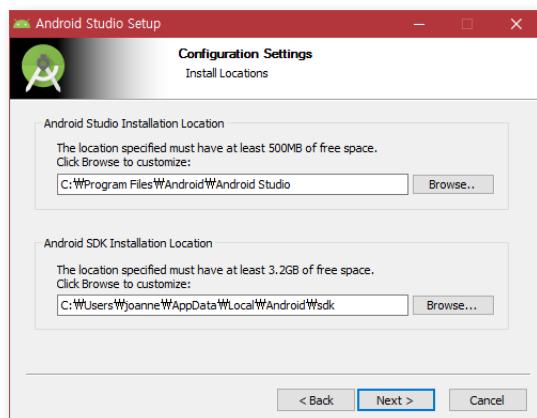


그림 4 | 안드로이드 스튜디오와 안드로이드 SDK 설치 위치 설정

위치를 변경하거나 기본 위치를 그대로 두고 ‘다음(Next)’을 클릭한다. 설치 프로그램은 기본적으로 프로그램 실행을 위한 바로 가기를 생성하기를 원할 경우 바로 가기 생성을 거부할 수 있다. 가급적 바로 가기를 생성하고, ‘설치(Install)’ 버튼을 클릭해서 설치를 시작한다.

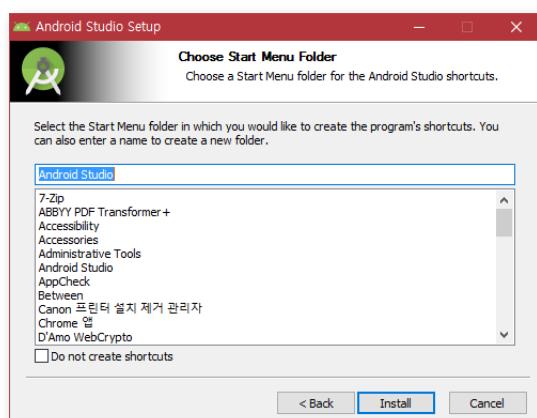


그림 5 | 안드로이드 스튜디오 바로 가기 생성

이후 대화 상자에는 안드로이드 스튜디오와 안드로이드 SDK의 설치 진행률이 표시된다. ‘자세히 보기>Show Details’ 버튼을 클릭하면 설치 진행 과정에 관한 세부 정보를 볼 수 있다.

설치가 완료되었다는 메시지가 표시되면 ‘다음(Next)’을 클릭한다. 그림 6과 같은 대화 상자가 표시된다.



그림 6 | 소프트웨어를 실행하려면 안드로이드 스튜디오 시작(Start Android Studio) 확인란이 선택된 상태 그대로 둔다.

설치를 완료하려면 안드로이드 스튜디오 시작(Start Android Studio) 확인란이 선택된 상태로 마침(Finish)을 클릭한다.

안드로이드 스튜디오 실행

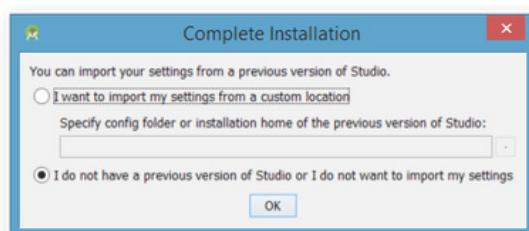
안드로이드 스튜디오를 실행하면 다음과 같은 시작 화면이 표시된다.



그림 7 | 안드로이드 스튜디오 시작 화면

처음 실행하면 프로그램 구성을 위한 몇 가지 대화 상자가 등장한다. 첫 번째 대화 상자에서는 이전에 설치했던 안드로이드 스튜디오 버전에서 설정을 가져올지 여부를 묻는다.

그림 8 | 설정 가져오기



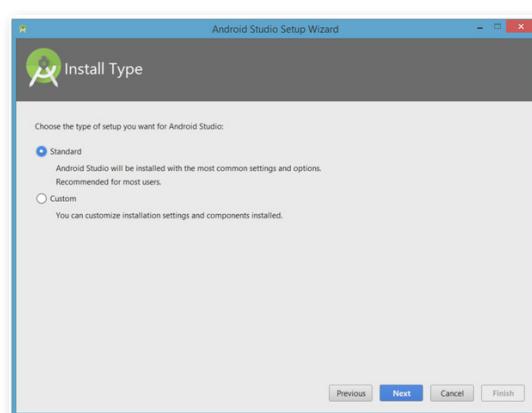
기존에 설치한 버전이 없으면, 기본 설정 그대로 두고 ‘확인(OK)’을 클릭한다. 다시 시작 화면이 표시되고 이후 안드로이드 스튜디오 설정 마법사 상자가 나타난다.

그림 9 | 안드로이드 SDK 및 개발 환경 설정 확인



‘다음(Next)’을 클릭하고 설정 마법사의 안내에 따라 SDK 구성 요소의 설치 유형을 선택한다. 일단 기본 표준 설정을 그대로 유지하는 것이 좋다.

그림 10 | 설치 유형 선택



‘다음(Next)’을 클릭하고 설정을 확인한 다음 ‘마침(Finish)’을 선택해서 계속 진행한다.

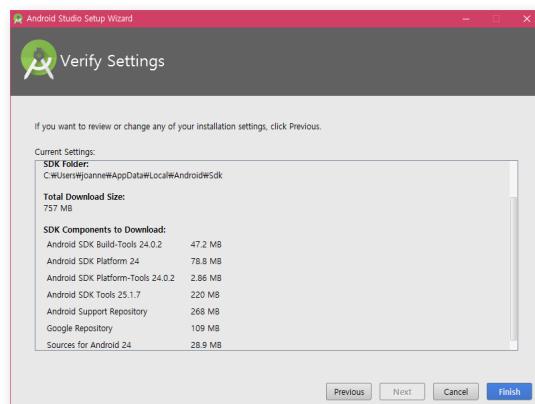


그림 11 | 설정 검토

마법사가 다양한 구성 요소를 다운로드해서 압축을 푼다. 다운로드하는 압축 파일과 내용에 대한 추가 정보를 보려면 ‘자세히 보기>Show Details’를 클릭한다.

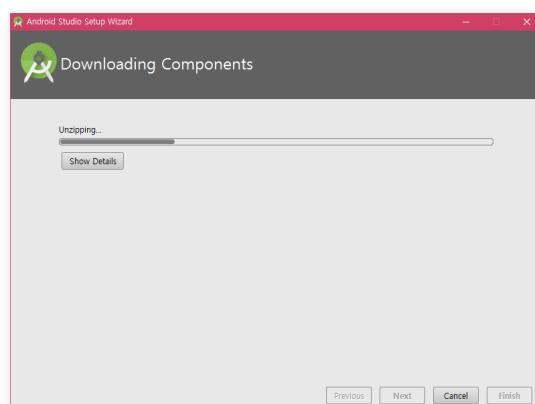


그림 12 | 마법사가 안드로이드 스튜디오 구성 요소를 다운로드하고 압축을 푼다.

인텔 기반 컴퓨터가 아니라면, 구성 요소가 완전히 다운로드되고 압축이 해제된 후 그림 13과 같은 경고 메시지가 나타날 수 있다.

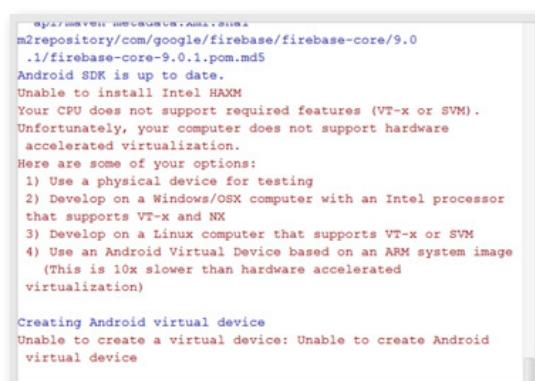


그림 13 | 인텔 기반 하드웨어 가속을 사용할 수 없음

이 경우 선택할 수 있는 방법은 느린 에뮬레이터를 감수하고 그냥 사용하거나 안드로이드 기기로 개발 속도를 높이는 것이다. 후자에 대해서는 가이드 후반에서 구체적으로 알아본다.

마지막으로 ‘마침(Finish)’을 클릭해서 마법사를 완료한다. 이제 그림 14와 같은 안드로이드 스튜디오 시작 대화 상자가 표시된다.



그림 14 | 안드로이드 스튜디오 시작

이제 이 대화 상자에서 새로운 안드로이드 스튜디오 프로젝트를 시작하거나 기존 프로젝트 작업을 이어갈 수 있다. 바탕 화면의 안드로이드 스튜디오 바로 가기를 두 번 클릭해서 언제든지 이 대화 상자에 접근할 수 있다.

안드로이드 스튜디오 모바일 앱 처음 만들기

안드로이드 스튜디오를 가장 빠르게 학습하는 방법은 안드로이드 스튜디오를 사용해 앱을 개발하는 것이다. 이 강좌에서는 “Hello, World” 애플리케이션을 약간 변형해서 “Welcome to Android” 메시지를 표시하는 간단한 모바일 앱부터 시작할 것이다.

이어질 단계에서는 새 안드로이드 스튜디오 프로젝트를 시작해서 프로젝트 작업 공간을 익힌다. 이 작업 공간에는 2부에서 앱을 코딩하는 데 사용할 프로젝트 편집기가 포함된다.

새 프로젝트 시작하기

지금까지의 설정 단계에 따랐다면 안드로이드 스튜디오 대화 상자가 아직 열려 있는 상태일 것이다. 여기서 ‘새 안드로이드 스튜디오 프로젝트 시작(Start a new Android Studio project)’을 클릭한다. 그림 15와 같은 ‘새 프로젝트 만들기(Create New Project)’ 대화 상자가 표시된다.

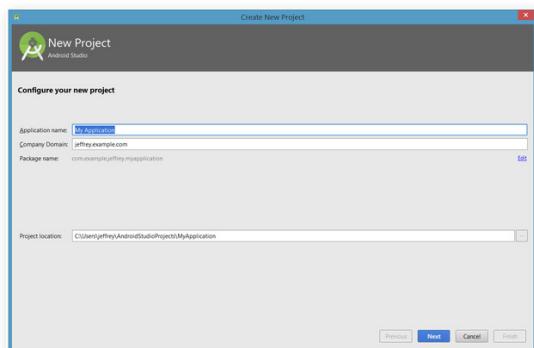


그림 15 | 새 프로젝트 만들기

애플리케이션 이름으로 W2A(Welcome to Android의 약어)를, 회사 도메인 이름으로 javajeff.ca를 입력한다. 프로젝트 위치가 C:\Users\jeffrey\AndroidStudioProjects\W2A로 설정될 것이다. ‘다음(Next)’을 클릭해서 대상 기기를 선택한다.

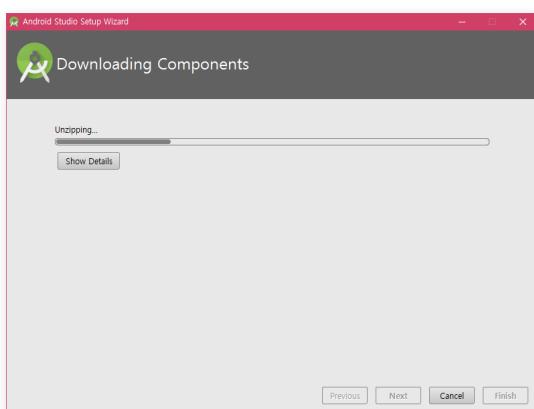


그림 16 | 대상 기기 범주 선택

안드로이드 스튜디오에서 만드는 모든 앱에 대한 폼 팩터, 즉 대상 기기의 범주를 선택할 수 있다. ‘전화기 및 태블릿(Phone and Tablet)’ 메뉴에서 필자의 아마존 킨들 파이어 HD 태블릿이 지원하는 기본 API 15: Android 4.0.3(IceCreamSandwich) 최소 SDK 설정을 그대로 유지하려고 했지만, 현재 안드로이드 스튜디오는 이 API 수준을 지원하지 않으므로(SDK 관리자에 4.0.3 시스템 이미지를 추가해도 마찬가지) API 14: Android 4.0(IceCreamSandwich)으로 변경했다. 역시 필자의 태블릿에서 지원하는 API다.

‘다음(Next)’을 선택하면 앱의 기본 동작 템플릿을 선택할 수 있다. 여기서는 ‘빈 동작(Empty Activity)’ 템플릿을 선택하고 ‘다음(Next)’을 클릭한다.

How To | 초보 개발자를 위한 안드로이드 스튜디오 입문

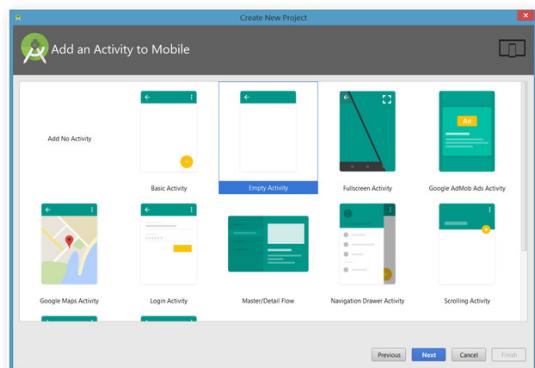


그림 17 | 동작 템플릿 지정

다음 순서는 동작 맞춤 설정이다.

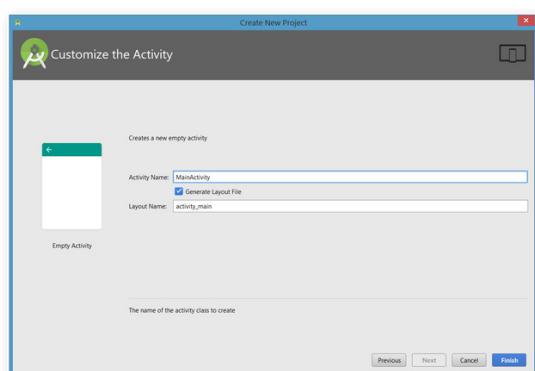


그림 18 | 동작 맞춤 설정

동작 이름으로 W2A를, 레이아웃 이름으로 main을 입력한 후 ‘마침(Finish)’을 클릭해서 단계를 마친다. 프로젝트를 만드는 중이라는 메시지가 표시된 후 프로젝트 작업 공간이 열린다.

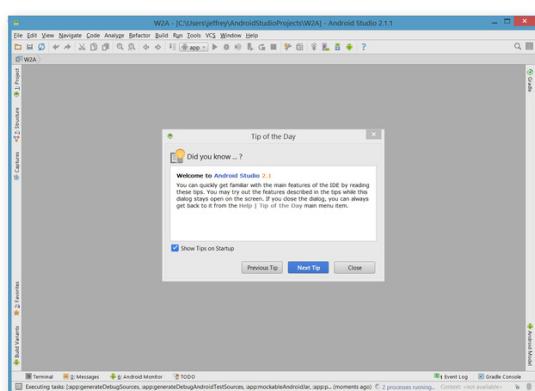


그림 19 | 안드로이드 스튜디오 작업 공간

프로젝트 작업 공간은 메뉴 표시줄, 도구 모음, 작업 영역, 별도의 창(예: 그래들 콘솔 창)으로 열리는 추가 구성 요소, 상태 표시줄로 구성된다. ‘오늘의 팁(Tip of the Day)’ 대화 상자도 표시되는데 필요 없다면 비활성화할 수 있다.

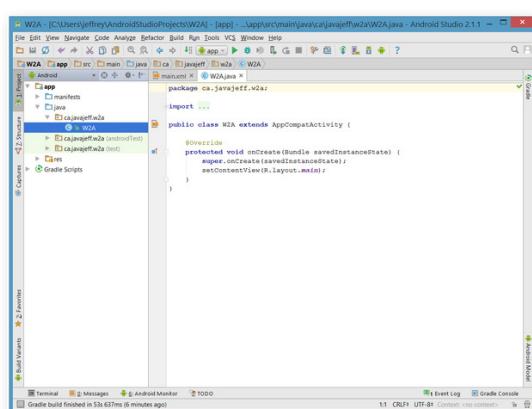
메뉴 및 도구 모음에서 AVD 관리자 또는 SDK 관리자 접근

전통적인 AVD 관리자나 SDK 관리자에 접근하려면 ‘도구(Tools)’ 메뉴에서 ‘안드로이드(Android)’를 선택하고 팝업 메뉴에서 ‘AVD 관리자(AVD Manager)’나 ‘SDK 관리자(SDK Manager)’를 선택한다(또는 각각의 도구 모음 아이콘 클릭).

프로젝트 및 편집기 창

프로젝트 작업 공간으로 들어가면 W2A가 현재 프로젝트로 나타나지만, 프로젝트 세부 정보는 바로 표시되지 않는다. 잠깐 기다리면 두 개의 새 창에 프로젝트 세부 정보가 표시된다.

그림 20 | 프로젝트 및 편집기 창



프로젝트 창은 트리 형태로 구성되며 최상위 가지는 앱(App)과 그래들 스크립트(Gradle Scripts)다. 앱 가지는 다시 매니페스트(manifests), 자바(java), res 하위 가지로 나뉜다.

- 매니페스트에는 안드로이드 앱의 구조를 설명하는 XML 파일인 **AndroidManifest.xml**이 저장된다. 이 파일은 권한 설정(해당되는 경우)과 기타 앱에 대한 세부 정보도 기록한다.
- 자바에는 패키지 계층에 따라 앱의 자바 소스 파일이 포함된다. 이 예제에서는 **ca.javajeff.w2a**다.
- res에는 앱의 리소스 파일이 저장되며 각 파일은 드로어블(drawable), 레이아웃(layout), 미니맵(minimap), 값(values) 하위 가지로 분류된다.
 - 드로어블: 앱의 아트워크를 저장하기 위한 위치로 처음에는 비어 있음
 - 레이아웃: 앱의 레이아웃 파일이 포함된 위치. 처음에는 **main.xml**(주 동작의 레이아웃 파일)이 여기에 저장됨
 - 미니맵: 다양한 해상도의 런처 화면 아이콘이 포함된 다양한 **ic_launcher.png** 파일의 위치
 - 값: **colors.xml**, **dimens.xml**, **strings.xml**, **styles.xml**이 포함된 위치

그래들 스크립트 가지에는 그래들 기반 빌드 시스템에 사용되는 다양한 **.gradle**(예: build.gradle) 및 **.properties**(예: local.properties) 파일이 나타난다.

가지 이름과 디렉터리/파일 이름

각 가지/하위 가지는 디렉터리 이름 또는 파일 이름에 해당한다. 예를 들어, res는 res 디렉터리에 해당하며 strings.xml은 strings.xml 파일에 해당한다.

1부 맷음말

지금까지 안드로이드 스튜디오를 설치 및 구성하고 첫 안드로이드 스튜디오 모바일 앱을 위한 프로젝트를 만들었다. 이제 안드로이드 애플리케이션을 빌드 할 준비가 된 것이다. 안드로이드 스튜디오에서 애플리케이션 빌드란 모바일 앱을 위한 자바 소스 코드와 리소스 파일로 새 프로젝트의 내용을 채우는 것을 의미한다. 각 기능이 동작하는 첫 안드로이드 모바일 앱을 만들 준비가 됐다면 2부로 넘어가자. **ITWORLD**



ITWORLD

테크놀로지 및 비즈니스 의사 결정을 위한 최적의 미디어 파트너



기업 IT 책임자를 위한 글로벌 IT 트렌드와 깊이 있는 정보

ITWorld의 주 독자층인 기업 IT 책임자들이 원하는 정보는 보다 효과적으로 IT 환경을 구축하고 IT 서비스를 제공하여 기업의 비즈니스 경쟁력을 높일 수 있는 실질적인 정보입니다.

ITWorld는 단편적인 뉴스를 전달하는 데 그치지 않고 업계 전문가들의 분석과 실제 사용자들의 평가를 기반으로 한 깊이 있는 정보를 전달하는 데 주력하고 있습니다. 이를 위해 다양한 설문조사와 사례 분석을 진행하고 있으며, 실무에 활용할 수 있고 자료로서의 가치가 있는 내용과 형식을 지향하고 있습니다.

특히 IDG의 글로벌 네트워크를 통해 확보된 방대한 정보와 전세계 IT 리더들의 경험 및 의견을 통해 글로벌 IT의 표준 패러다임을 제시하고자 합니다.

2

세부 내용 알아보고 앱 코딩 시작하기

Jeff Friesen | JavaWorld

초 보자용 안드로이드 스튜디오 입문 강좌 1부에서는 개발 환경에 안드로이드 스튜디오를 설치하고 프로젝트 작업 공간을 살펴봤다. 이번 2부에서는 첫 앱을 코딩해볼 것이다. 코딩할 애니메이션 모바일 앱은 하나의 동작으로 구성되며 구글 안드로이드 로봇 캐릭터와 이 캐릭터를 움직이기 위한 버튼이 있다. 버튼을 클릭하면 캐릭터의 색이 녹색에서 빨간색, 파란색으로 차차 바뀌었다가 다시 녹색으로 돌아온다. 딱히 쓸모가 있는 앱은 아니지만, 앱을 작성하는 과정에서 안드로이드 스튜디오 사용법을 익힐 수 있다. 3부에서는 앱을 컴파일해서 안드로이드 기기 에뮬레이터와 7" HD 킨들 파이어 태블릿에서 실행해본다.

이 강좌의 예제 애플리케이션을 위한 자바 소스를 다운로드한다.

예제 앱 다운로드

(by Jeff Friesen for JavaWorld)

프로젝트 편집기 시작하기

1부의 마지막 부분에서 안드로이드 스튜디오 프로젝트 작업 공간을 소개했다. 프로젝트 작업 공간에는 안드로이드 스튜디오에서 모바일 앱 코드를 작성하고 리소스를 지정하는 작업을 수행하는 영역인 프로젝트 편집기가 포함된다. 그림 21에서 프로젝트 편집기를 볼 수 있다.

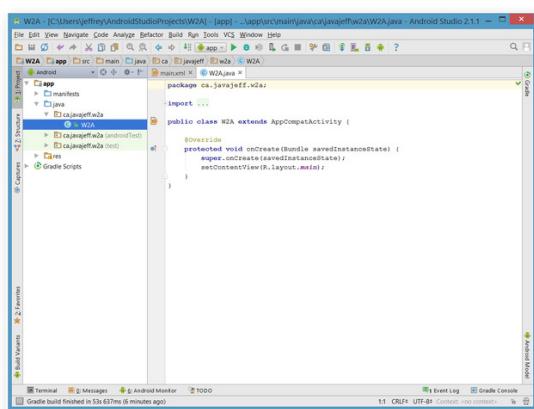


그림 21 | 안드로이드 스튜디오 프로젝트 편집기

탭을 사용해서 자바 소스 파일, XML 리소스 파일 및 기타 파일을 살펴볼 수 있다. 프로젝트 창에서 파일 이름을 두 번 클릭하면 프로젝트 파일용 탭을 추가 할 수 있다.

그림 21에서 볼 수 있듯이 현재 프로젝트 편집기에는 W2A.java(앱의 주 동작을 위한 골격 자바 소스 코드)와 main.xml(앱의 주 동작을 위한 기본 XML 기반 레이아웃), 두 개의 탭이 있다. 현재 열려 있는 탭은 W2A.java이다.

W2A 예제 앱

W2A 앱은 안드로이드 로봇 캐릭터와 버튼을 표시하는 하나의 주 동작으로 구성된다. 사용자가 버튼을 누르면 로봇의 색이 변경되는 애니메이션이 작동한다. 이 단원에서는 동작의 소스 코드와 리소스를 살펴본다.

W2A.java 세부 내용과 코딩

동작의 소스 코드는 목록 1의 **W2A.java** 파일에 저장된다.

목록 1. W2A.java

```
package ca.javajeff.w2a;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
public class W2A extends Activity
{
    AnimationDrawable androidAnimation;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView androidImage = (ImageView) findViewById(R.id.android);
        androidImage.setBackgroundResource(R.drawable.android_animate);
        androidAnimation = (AnimationDrawable) androidImage.getBackground();
        final Button btnAnimate = (Button) findViewById(R.id.animate);
        View.OnClickListener ocl;
        ocl = new View.OnClickListener()
```

```
{  
    @Override  
    public void onClick(View v)  
    {  
        androidAnimation.stop();  
        androidAnimation.start();  
    }  
};  
btnAnimate.setOnClickListener(ocl);  
}  
}
```

W2A.java 파일의 시작 부분은 패키지 문으로, 이를 통해 **W2A** 클래스를 저장하는 패키지(**ca.javajeff.w2a**)의 이름을 지정한다. 그 뒤로 다양한 안드로이드 API 유형을 가져오는 일련의 가져오기 문이 나오고, 그다음으로 **android.app.Activity**를 확장하는 **W2A** 클래스 기술 코드가 있다.

W2A는 먼저 **android.graphics.drawable.AnimationDrawable** 형식의 **androidAnimation** 인스턴스 필드를 선언한다. **AnimationDrawable** 형식의 객체는 프레임별 애니메이션을 기술하는데, 여기서 현재 드로어블이 애니메이션 시퀀스의 다음 드로어블로 대체된다.

애니메이션드로어블(AnimationDrawable)이란?

드로어블은 예를 들어 이미지와 같이 그릴 수 있는 것을 나타낸다. **AnimationDrawable**은 추상적 **android.graphics.drawable.Drawable** 클래스를 간접적으로 확장하는데, 일반적인 드로어블 추상화라 할 수 있다.

onCreate() 메소드

앱의 모든 작업은 **W2A**의 오버라이딩 **onCreate(Bundle)** 메소드에서 이루어진다. 다른 메소드는 필요 없기 때문에 앱은 복잡하지 않고 단순하다.

onCreate(Bundle)은 먼저 모든 오버라이딩 동작 메소드에서 따라야 하는 규칙, 즉 같은 이름의 슈퍼클래스 메소드를 호출한다.

그런 다음 **setContentView(R.layout.main)**을 실행해서 앱의 사용자 인터페이스를 설정한다. **R.layout.main**은 별도의 파일에 위치하는 애플리케이션 리소스의 식별자(ID)다. 이 ID를 해석하자면 다음과 같다.

- **R**은 앱이 빌드될 때 생성되는 클래스의 이름이다. 이 클래스의 이름이 **R**인 이유는 클래

스의 내용이 레이아웃, 이미지, 문자열, 색상 등 다양한 종류의 애플리케이션 리소스를 지정하기 때문이다.

- **layout**은 **R**내에 중첩된 클래스 이름이다. ID가 이 클래스에 저장되는 애플리케이션 리소스는 특정 레이아웃 리소스를 기술한다. 각 애플리케이션 리소스 종류는 비슷한 방식으로 명명된 중첩된 클래스와 연결된다. 예를 들어 **string**은 문자열 리소스를 지정한다.
- **main**은 **layout** 내에 선언되는 **int** 기반 상수의 이름이다. 이 리소스 ID는 주 레이아웃 리소스를 지정한다. 구체적으로 보면 **main**은 주 동작의 레이아웃 정보를 저장하는 **main.xml** 파일을 가리킨다. **main**은 **W2A**의 유일한 레이아웃 리소스다.

R.layout.main을 **Activity**의 **void setContentView(int layoutResID)** 메소드에 전달하면 안드로이드는 **main.xml**에 저장된 레이아웃 정보를 사용하여 사용자 인터페이스 화면을 생성한다. 이때 내부적으로 안드로이드는 **main.xml**에 기술된 사용자 인터페이스 구성 요소를 생성하고 이를 **main.xml**의 레이아웃 데이터에 지정된 대로 기기 화면에 배치한다.

화면은 뷰(사용자 인터페이스 구성 요소의 추상화)와 뷰 그룹(관련 사용자 인터페이스 구성 요소를 그룹화하는 뷰)을 기반으로 한다. 뷰는 **android.view.View** 클래스를 하위 클래스로 두는 클래스의 인스턴스이며 AWT/Swing 구성 요소와 유사하다. 뷰 그룹은 추상 **android.view.ViewGroup** 클래스를 하위 클래스로 두는 클래스의 인스턴스이며 AWT/Swing 컨테이너와 유사하다. 안드로이드는 특정 뷰(예: 버튼 또는 스피너)를 위젯으로 지칭한다.

계속해서 **onCreate(Bundle)**은 **ImageView androidImage = (ImageView) findViewById(R.id.android);**를 실행한다. 이 문은 먼저 **View**의 **View findViewById(int id)** 메소드를 호출해 **main.xml**에 선언되고 **android**로 식별되는 **android.widget.ImageView** 요소를 찾는다. 이 문은 **ImageView**를 인스턴스화하고 **main.xml** 파일에 선언된 값으로 초기화한다. 그런 다음 이 객체의 참조를 로컬 변수 **androidImage**에 저장한다.

이미지뷰와 애니메이션드로어블

다음으로, **androidImage.setBackgroundDrawable(R.drawable.android_animate);** 문은 **ImageView**의 상속된(**View**에서 상속) **void setBackgroundDrawableMethod(int resID)** 메소드를 호출하고 뷰의 배경을 **resID**로 식별되는 리소스로 설정한다. **R.drawable.android_animate** 인수는 **android_animate.xml**(뒷부분에서 설명)이라는 XML 파일을 지정하는데, 이 파일은 애니메이션에 대한 정보를 저장하고 **res**의 **drawable** 하위 디렉토리에 저장된다. **setBackgroundDrawable()** 호출은 **androidImage** 뷰를 **android_animate.xml**에 기술된 이미지 시퀀스(이 뷰에 그려짐)에 연결한다. 이 메소드 호출의 결과로 첫 이미지가 그려진다.

ImageView는 앱에서 **AnimationDrawable** 메소드를 호출하여 드로어블 시퀀스를 애니메이션으로 처리할 수 있게 해준다. 이를 위해 먼저 앱은 **ImageView**의

AnimationDrawable을 획득해야 한다. 그 뒤의 **androidAnimation = (AnimationDrawable) androidImage.getBackground();** 할당 문이 **ImageView**의 상속된(**View**에서 상속) **Drawable getBackground()** 메소드를 호출함으로써 이 작업을 수행한다. 이 메소드는 주어진 **ImageView**에 대한 **AnimationDrawable**을 반환하는데, 이는 이후 **androidAnimation** 필드에 할당된다. **AnimationDrawable** 인스턴스는 애니메이션을 시작하고 멈추는 데 사용된다. 이 프로세스에 대해서는 잠시 후 설명할 것이다.

마지막으로, **onCreate(Bundle)**은 ‘Animate’ 버튼을 생성한다. **findViewById(int)**를 호출하여 **main.xml**에서 버튼 정보를 획득한 다음 **android.widget.Button** 클래스를 인스턴스화한다.

그 후 **View** 클래스의 중첩된 **onClickListener** 인터페이스를 사용하여 리스너(listener) 객체를 생성한다. 이 객체의 **void onClick(View v)** 메소드는 사용자가 버튼을 클릭할 때마다 호출된다. 리스너는 **View**의 **void setOnClickListener(AdapterView.OnClickListener listener)** 메소드를 호출하여 **Button** 객체에 등록된다.

Animate의 클릭 리스너는 애니메이션을 멈춘 다음 시작하기 위해 **androidAnimation.stop(), androidAnimation.start()**를 차례로 호출한다. 이후 ‘Animate’ 버튼 클릭 시 새 애니메이션이 시작될 수 있도록 **stop()** 메소드가 **start()** 전에 호출된다.

애플리케이션 코드를 업데이트하고 저장하기

계속하기 전에 W2A.java 탭의 골격 코드를 목록 2의 코드로 대체한다. Ctrl+S를 누르거나 파일(File) 메뉴에서 저장(Save)을 선택해서 이 창의 내용을 저장한다.

main.xml 세부 내용과 코딩

앱의 주 동작은 **main.xml** 파일에 저장되며 목록 2에 나와 있는 XML 기반 레이아웃과 연결된다.

목록 2. main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:background="#ffffffff">
    <ImageView android:id="@+id/android"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```

        android:layout_marginBottom="10dip"/>
<Button android:id="@+id/animate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/animate"/>
</LinearLayout>
```

XML 선언 이후 목록 2는 포함된 위젯(중첩된 레이아웃 포함)을 화면에서 가로 또는 세로로 배치하기 위해 레이아웃(포함된 뷰를 일정한 방식으로 안드로이드 기기의 화면에 배치하는 뷰 그룹)을 지정하는 **LinearLayout** 요소를 선언한다.

<LinearLayout> 태그는 이 선형 레이아웃을 제어하기 위한 여러 가지 특성을 지정한다. 이러한 특성에는 다음이 포함된다.

- **orientation**은 선형 레이아웃을 가로 또는 세로로 지정한다. 포함된 위젯은 가로 또는 세로로 배치되며 기본 방향은 가로다. 이 특성에는 “**horizontal**”과 “**vertical**”, 두 가지 값만 할당할 수 있다.
- **layout_width**는 레이아웃의 너비를 지정한다. 사용 가능한 값은 “**fill_parent**”(부모만큼의 너비)와 “**wrap_content**”(내용을 담을 만큼의 너비)이다. (참고로 **fill_parent**는 안드로이드 2.2에서 **match_parent**로 바뀌었지만, 지금까지 지원되며 널리 사용된다.)
- **layout_height**는 레이아웃의 높이를 지정한다. 사용 가능한 값은 “**fill_parent**”(부모만큼의 높이)와 “**wrap_content**”(내용을 담을 만큼의 높이)이다.
- **gravity**는 화면에 대한 레이아웃의 상대적인 위치를 지정한다. 예를 들어 “**center**”는 레이아웃이 화면에서 가로, 세로로 중앙에 위치하도록 지정한다.
- **background**는 배경 이미지를 그라데이션이나 단색으로 지정한다. 강좌에서는 단순하게 하기 위해 16진수 색 식별자로 흰색 단색 배경을 지정했다(#ffffff). (색은 일반적으로 **colors.xml**에 저장되며 이 파일에서 참조된다.)

LinearLayout 요소는 **ImageView** 및 **Button** 요소를 캡슐화한다. 이러한 각 요소는 **id** 특성을 지정하는데, 이 특성은 요소를 지정하여 코드에서 참조할 수 있도록 한다. 이 특성에 할당되는 리소스 식별자(@으로 시작하는 특수 구문)는 앞에 **@+id**가 붙는다. 예를 들어 **@+id/android**는 **ImageView** 요소를 **android**로 지정한다. 이 요소는 코드에서 **R.id.android**를 지정하여 참조된다.

여기에서는 내용 배치를 결정하기 위한 **layout_width** 및 **layout_height** 특성도 지정한다. 요소가 실물 크기로 표시되도록 각 특성에는 **wrap_content**가 할당된다.

ImageView는 **layout_marginBottom** 특성을 지정하여 세로 방향으로 아래 버튼까지의 공백 구분자를 지정한다. 공백은 **10 dip**(밀도 독립적 픽셀)로 지정된다. dip은 앱이 화면 밀도에 대해 독립적으로 레이아웃 크기/위치를 표현하는 데 사용

밀도 독립적 픽셀이란?

밀도 독립적 픽셀은 안드로이드가 전제하는 기본 밀도인 160dpi 화면에서 하나의 물리적 픽셀에 해당한다. 런타임에 안드로이드는 사용 중인 화면의 실제 밀도를 기준으로 필요한 dip 단위의 스케일링을 알아서 처리한다. dip 단위는 픽셀 = dip * (밀도 / 160) 공식에 따라 화면 밀도로 변환된다. 예를 들어 240dpi 화면에서 1dp은 1.5개의 물리적 픽셀과 같다. 구글은 다양한 기기 화면에서 사용자 인터페이스가 적절히 표시되도록 하기 위해 앱의 사용자 인터페이스를 dip 단위로 정의할 것을 권장한다.

할 수 있는 가상 픽셀이다.

새 레이아웃 선택 및 저장

1부에서 앱을 설정할 때 ‘빈 동작’ 템플릿을 선택했지만, 이 템플릿이 제공하는 XML 레이아웃은 이 강좌에서 소개하는 튜토리얼 앱에 적합하지 않다. 새 레이아웃을 선택하려면 먼저 main.xml 탭을 클릭한다. 그러면 사용하기 편리한 레이아웃 편집기가 표시된다.

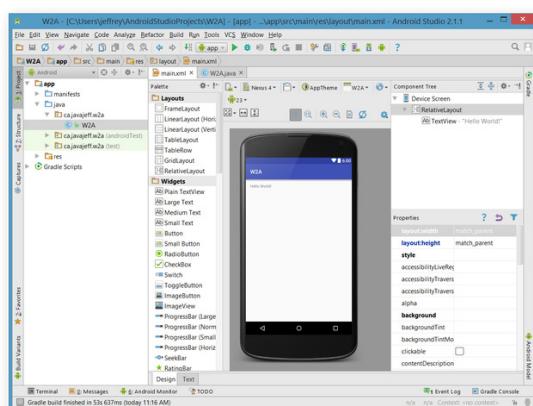


그림 22 | 안드로이드 스튜디오의 레이아웃 편집기

계속 진행하기 전에 main.xml 탭 맨 아래의 ‘텍스트(Text)’ 탭(기본적으로는 ‘디자인(Design)’ 탭이 표시됨)을 클릭한다. 템플릿 XML을 목록 2의 코드로 교체한 다음 창 내용을 저장한다.

strings.xml의 세부 내용과 코딩

W2A는 다른 위치에서 참조되는 문자열 데이터를 **strings.xml**을 사용하여 저장한다. 목록 2로 돌아가서 보면 <Button> 태그에 **android:text="@string/animate"** 특성이 포함된 것을 볼 수 있다. 이 특성은 **strings.xml**에 저장된 **animate** 문자열 리소스를 참조하는 **@string/animate**를 통해 버튼의 텍스트를 참조한다. 목록 3에는 이 파일의 내용이 나와 있다.

목록 3. strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">W2A</string>
    <string name="animate">Animate</string>
</resources>
```

목록 3에는 **animate**와 함께 **app_name**으로 지정되는 문자열 리소스가 나와 있다. 이 리소스 ID는 앱 이름을 지정하며 앱의 **AndroidManifest.xml** 파일에서 참조된다(일반적으로 애플리케이션 요소 시작 태그의 **label** 특성에서 참조됨).

모바일 앱 국제화

문자열을 **strings.xml**에 직접 저장하고 다른 곳에서 이러한 리소스를 참조하는 것은 소스 파일 및 기타 리소스 파일에 문자열을 하드코딩하는 것보다 좋은 방법이다. 이 방법을 사용하면 해외 시장에 맞춰 앱을 더 쉽게 수정하고 앱의 수익 가능성도 더 높일 수 있다.

strings.xml 저장

프로젝트 창에서 res 하위 가지의 values 하위 가지에는 strings.xml 하위 가지가 포함된다. 이 하위 가지를 두 번 클릭해서 strings.xml 탭을 표시한 후, 그 내용을 목록 3의 내용으로 대체하고 변경 사항을 저장한다.

animate.xml의 세부 내용과 코딩

마지막으로, W2A는 드로어를 항목의 애니메이션을 저장하는 **android_animate.xml**을 사용한다. 목록 4에는 이 파일의 내용이 나와 있다.

목록 4. android_animate.xml

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/android0" android:duration="500" />
    <item android:drawable="@drawable/android1" android:duration="500" />
    <item android:drawable="@drawable/android2" android:duration="500" />
    <item android:drawable="@drawable/android0" android:duration="500" />
</animation-list>
```

목록 4의 시작 부분에는 드로어블 시퀀스를 기술하는 **animation-list** 요소가 있다. 이 요소의 **oneshot** 특성은 애니메이션이 루프를 순환할지(이 특성에 “**false**”가 할당된 경우) 한 번만 수행할지(“**true**”가 할당되는 경우)를 결정한다. **oneshot**에 “**true**”가 할당되는 경우 **start()** 메소드 전에 **AnimationDrawable()**의 **stop()** 메소드를 호출하여 다른 원샷(oneshot) 애니메이션 시퀀스를 생성해야 한다.

animation-list 요소 내에는 **item** 요소가 중첩되어 있다. 각 **item** 요소는 **drawable** 특성을 통해 애니메이션 시퀀스에서 하나의 드로어블을 지정한다. **@drawable/** **androidx** 리소스 참조(x의 범위는 0 ~ 2)는 이름이 **android**로 시작하는 이미지 파일을 지정한다. **duration** 특성은 다음 **item** 요소의 드로어블을 표시하기 전에 경과해야 할 밀리초를 지정한다.

animate.xml 저장

프로젝트 창에서 res 하위 가지의 ‘드로어블’ 하위 가지를 마우스 오른쪽 버튼으로 클릭하면 팝업 메뉴가 표시된다. 그림 23에서 이 팝업 메뉴를 볼 수 있다.

팝업 메뉴에서 ‘새로 만들기(New)’, ‘파일(File)’을 차례로 선택한다. ‘새 파일

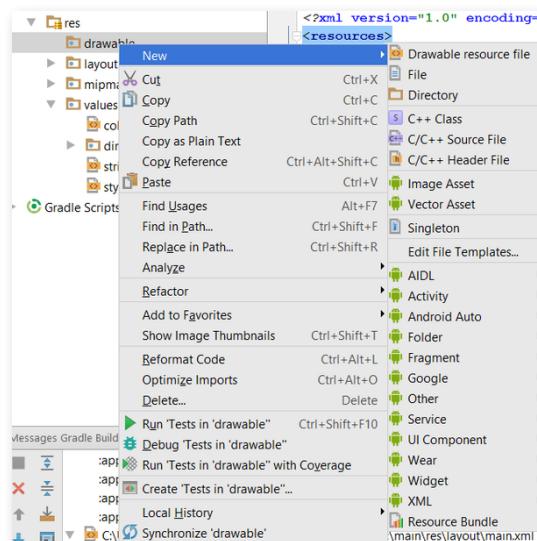


그림 23 | android_animate.xml을 새 드로어블 리소스로 추가

(New File)’ 대화 상자가 표시된다(그림 24 참조).

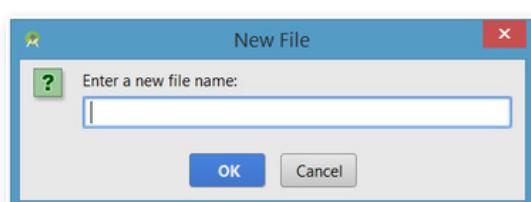


그림 24 | android_animate.xml 입력

'새 파일 이름 입력(Enter a new file name)' 텍스트 필드에 android_animate.xml을 입력하고 '확인(OK)'을 클릭한다(드로어를 하위 가지 아래에 android_animate.xml 하위 가지가 표시됨). 그런 다음 android_animate.xml 탭의 내용을 목록 4의 코드로 대체하고 변경 사항을 저장한다.

또, 연결된 코드 소스에서 **android0.png**, **android1.png**, **android2.png** 파일(목록 4에 참조됨)을 drawable 가지에 복사해야 한다. 윈도우에서 작업하는 경우 윈도우 탐색기에서 이 파일들을 선택하고 이 가지에 붙여 넣는다(가지 이름을 마우스 오른쪽 버튼으로 클릭하고 '붙여넣기(Paste)' 선택).

2부 맷음말

이제 W2A의 아키텍처를 이해했고 안드로이드 스튜디오에서 간단한 애니메이션 모바일 앱을 코딩하는 방법을 익혔다. 다음은 앱을 빌드해서 실행할 차례다. [ITWORLD](#)

**IT 트렌드 종합 정보센터
IDG Tech Library**

IDG Tech Library는 IDG 글로벌 네트워크를 통해 축적된 전문 정보를 재구성하여 최신 기술의 기본 개념부터 현황, 전략 및 도입 가이드까지 다양한 프리미엄 IT 정보를 제공합니다. Computer World, Info World, CIO, Network World 등의 세계적 IT 유명 매체의 심도 깊은 정보를 무료로 만나보세요

IDG Deep Dive, Tech Focus, Summary, World Update 등의 다양한 콘텐츠를 제공 받을 수 있습니다.

IDG
INTERNATIONAL DATA GROUP

한국IDG(주) 서울시 중구 봉래동 1가 108번지 창회빌딩 4층 100-161 Tel : 02-558-6950 Fax : 02-558-6955
www.itworld.co.kr www.twitter.com/ITWorldKR www.facebook.com/Itworld.Korea

3

앱 빌드 후 실행하기

Jeff Friesen | JavaWorld

2부에서 안드로이드 스튜디오를 사용해서 첫 애니메이션 모바일 앱을 만들었다. 3부에서는 안드로이드 스튜디오를 사용한 모바일 애플리케이션 개발 입문의 마지막 단계를 익힌다. 먼저 오픈 소스 빌드 자동화 도구인 그레들(Gradle)을 사용해서 앱의 애플리케이션 패키지(APK) 파일을 빌드한다. 그다음 안드로이드 기기 애뮬레이터 또는 킨들 파이어 HD 태블릿에서 앱을 설치하고 실행하기 위한 방법을 알아본다.

앱 빌드하기

안드로이드 스튜디오 프로젝트에 소스 코드와 리소스 파일을 로드했으면 이제 처음으로 앱을 빌드할 준비가 됐다. 프로젝트 창에서 ‘빌드(Build)’ 메뉴를 볼 수 있는데, 이 메뉴를 사용해 그레들에 연결하고 W2A를 빌드할 것이다.

프로젝트 창의 ‘그레들(Gradle)’ 아래에서 ‘프로젝트 리빌드(Project Rebuild)’를 선택한다. 그레들이 빌드 프로세스를 시작한다. 이 과정에 약 1분이 소요된다. 문제 없이 진행되면 잠시 후 콘솔 창에 ‘BUILD SUCCESSFUL’이라는 메시지가 표시된다.

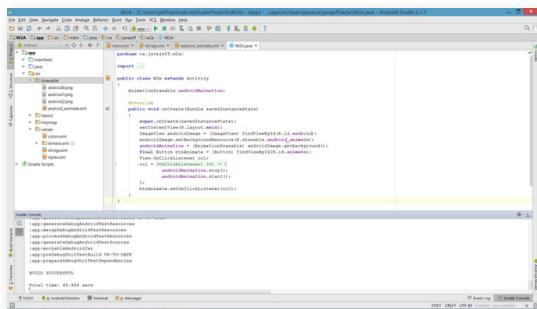


그림 25 | 빌드가 완료되면 그레들에 ‘BUILD SUCCESSFUL’ 메시지가 표시됨

‘프로젝트 리빌드(Rebuild Project)’를 선택하지 않고 직접 앱을 실행할 수도 있다. 이 경우 안드로이드 스튜디오가 그레들에 프로젝트 리빌드를 지시한다. 리빌드가 성공적으로 완료되면 안드로이드 스튜디오가 앱의 APK 파일(실

행 코드, 리소스 및 기타 정보가 포함된 파일)이 실제 또는 에뮬레이션 안드로이드 기기에 설치되었는지 확인한 다음 앱을 실행한다(참고로 ‘즉시 실행(Instant Run)’을 사용하면 앱을 재시작할 필요가 없는 경우도 있다).

그래들 활용하기

구글은 그래들 빌드 시스템을 더 효과적으로 사용하기 위한 다양한 팁을 제공한다. 예를 들어 빌드 서버 성능을 높이기 위한 팁이 있다.

부가적인 작업

안드로이드 스튜디오의 ‘빌드(Build)’ 메뉴에서는 기본 빌드 기능 외에 여러 가지 유용한 빌드 작업을 수행할 수 있다. ‘APK 빌드(Build APK)’ 및 서명된 APK 생성(Generate Signed APK)’ 메뉴 항목을 예로 들 수 있다. 이 메뉴를 사용하면 앱 애플리케이션 패키지 파일의 서명된 버전을 빌드하거나 디버깅을 위한 버전을 빌드할 수 있다.

앱 실행하기

프로젝트 창의 ‘실행(Run)’ 메뉴에는 앱을 실행하고 디버깅하기 위한 메뉴 항목이 있다. 이 단원에서는 에뮬레이션 기기와 아마존 킨들 파이어 HD 태블릿에서 W2A를 실행하는 방법을 알아본다.

에뮬레이션 기기에서 W2A 실행하기

‘실행(Run)’ 메뉴의 ‘앱 실행(Run app)’ 메뉴 항목을 선택해서 W2A 또는 다른 앱을 실행할 수 있다. 또는 도구 모음의 녹색 삼각형 버튼을 클릭해도 된다. 어느 방법을 사용하든 안드로이드 스튜디오에 ‘배포 대상 선택(Select Deployment Target)’ 대화 상자가 표시된다.

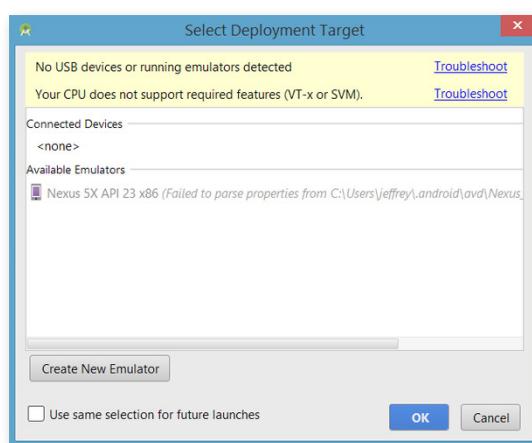


그림 26 | ‘배포 대상 선택(Select Deployment Target)’ 대화 상자

필자의 경우 컴퓨터의 프로세서가 인텔이 아니고 아직 아마존 킨들 파이어 HD 태블릿을 설정하지도 않은 상태였으므로 에뮬레이터를 사용해 앱을 실행하는 방법 외의 옵션이 없었다. 대화 상자에서 확인된 에뮬레이터는 하나인데, 안드로이드 스튜디오에서 이 에뮬레이터의 속성을 인식하지 못한 탓에 에뮬레이터를 사용할 수 없었다. 문제를 해결하기 위해 ‘새 에뮬레이터 만들기(Create New Emulator)’ 버튼을 클릭하자 다양한 에뮬레이터가 포함된 ‘가상 기기 구성(Virtual Device Configuration)’ 대화 상자가 표시됐다. 그림 27은 넥서스 5x(Nexus 5x)가 선택된 화면이다.

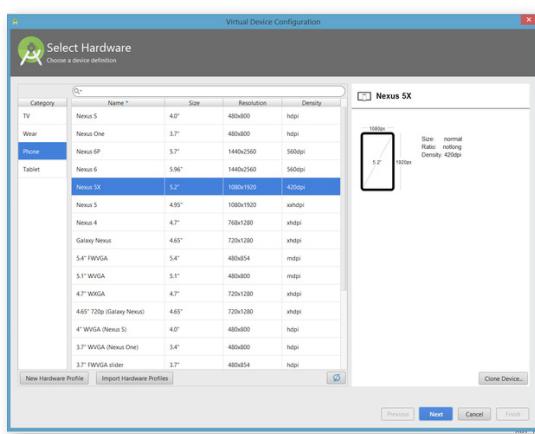


그림 27 | ‘새 에뮬레이터(New Emulator)’
창에서 선택할 수 있는 기기 에뮬레이터

필자는 넥서스 4(Nexus 4)를 선택하고 ‘다음(Next)’을 클릭했다. 이후 표시되는 시스템 이미지(System Image) 대화 상자에서는 시스템 이미지를, ‘다른 이미지(Other Images)’ 탭을 클릭한 후에는 IceCreamSandwich(구글 API 제외)를 선택했다.

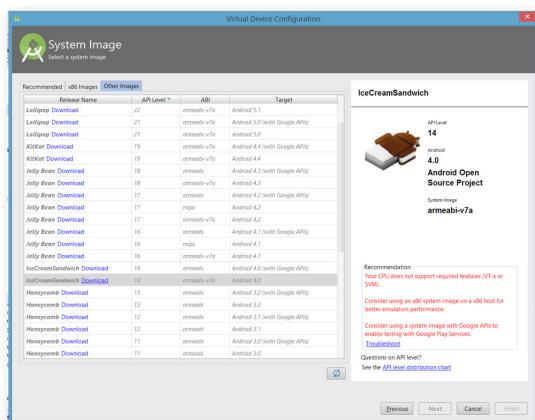


그림 28 | 에뮬레이터를 사용해서 인텔
이외의 플랫폼에서 앱 실행

에뮬레이터 설정을 마치고 나면 다운로드(Download) 링크를 클릭해서 기본적으로 설치되지 않은 이미지를 다운로드해야 한다. 라이선스 계약에 동의하고 다음(Next)을 클릭한다. 구성 요소 설치 프로그램에서 구성 요소 다운로드

를 시작한다.

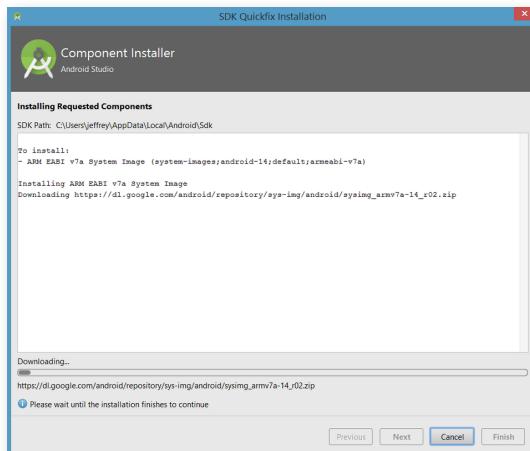


그림 29 | 설치가 완료되고 '마침(Finish)' 버튼이 밝게 표시되면 버튼을 클릭한다.

목록에서 'IceCreamSandwich'와 '다음(Next)'을 차례로 선택한다. 이제 AVD를 구성할 수 있다.

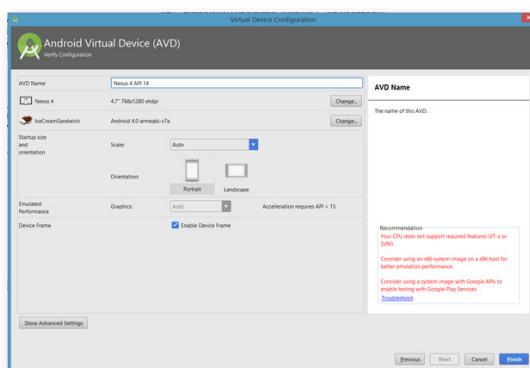


그림 30 | AVD를 적절히 구성하고 '마침(Finish)' 클릭



그림 31 | 에뮬레이션 넥서스 4 기기의 프레임 창

'실행(Run)' 메뉴에서 '앱 실행(Run app)'을 선택하거나 도구 모음의 녹색 삼각형 아이콘을 선택한다. '배포 대상 선택(Select Deployment Target)' 대화상자가 다시 표시되는데, 이번에는 넥서스 4 API 14(Nexus 4 API 14) 항목을 선택하고 '확인(OK)'을 클릭한다.

에뮬레이터에서 시작 화면이 먼저 표시되고 잠시 후 넥서스 4 기기의 프레임 창으로 바뀐다.

에뮬레이터 초기화에는 꽤 오랜 시간이 걸릴 수 있다. 필자의 경우 10분 이상 경과하자 안드로이드 스튜디오가 작업을 중단하고 다음과 같은 오류 메시지를 표시했다.

Error while waiting for device: Timed out after 300 seconds waiting for emulator to come online.

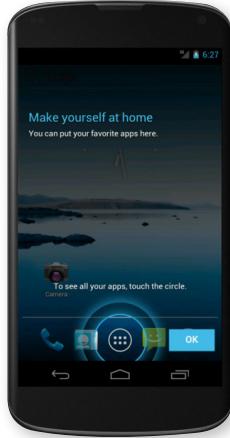


그림 32 | 에뮬레이션 네서스
4 기기의 프레임 창에
표시되는 시작 화면



그림 33 | 에뮬레이션 네서스
4 기기의 앱 런처 화면



그림 34 | W2A 아이콘이 앱
런처 화면의 오른쪽 하단에
표시됨

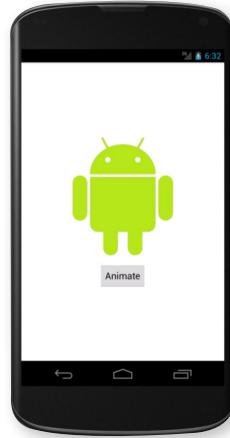


그림 35 | 버튼을 클릭하면
이미지 애니메이션이
시작된다.

이 오류가 발생할 때는 커피 한 잔을 마시고 책도 좀 읽은 다음 돌아오면 된다. 아마 그때쯤이면 에뮬레이션 기기의 시작 화면이 나타나 있을 것이다.

시작 화면을 닫고 런처 화면의 맨 위로 이동한다. W2A 앱 아이콘은 아직 이 화면에 표시되지 않는다.

앱을 다시 한 번 실행한다. 문제가 없다면 앱을 설치 중이라는 메시지가 표시된다. 또한, 앱 런처 화면에서 W2A 앱의 기본 안드로이드 아이콘을 볼 수 있다.

W2A가 자동으로 실행되므로 아이콘을 클릭할 필요는 없다. 녹색 안드로이드 로고와 Animate 버튼을 볼 수 있다.

아마존 킨들 파이어 HD 태블릿에서 앱 실행하기

느린 에뮬레이터를 통해 앱을 실행하다 보면 답답하다. 해결 방법은 지니모션(Genymotion) 안드로이드 에뮬레이터와 같이 속도가 더 빠른 에뮬레이션 소프트웨어를 사용하거나 실제 안드로이드 기기를 사용하는 것이다.

몇 년 전 필자는 안드로이드 4.0.3(API 레벨 15)을 실행하는 2012년형 1세대 아마존 킨들 파이어 HD 7인치 태블릿을 구매했다. 지금은 최신 안드로이드 API가 필요 없는 안드로이드 앱을 실행하는 용도로 사용 중이다.

킨들 파이어 태블릿을 안드로이드 스튜디오와 함께 사용하는 방법을 알아보

지니모션과 안드로이드 스튜디오

안드로이드 스튜디오에서 지니모션을 사용하는 방법에 대해 알아보고 싶다면 프란체스코 아졸라의 자바 코드 블로그 게시물인 “안드로이드 대안 에뮬레이터: 지니모션과 안드로이드 스튜디오 함께 사용하기(Android Studio with android alternative emulator:Genymotion)”를 참고하자.



그림 36 | 오른쪽 하단에 위치한 W2A 앱 아이콘



그림 37 | 'Animate'를 클릭하면 애니메이션이 실행된다.

던 중 아마존에서 제공하는 유용한 가이드 “파이어 태블릿을 위한 개발 환경 설정하기(Setting Up Your Development Environment for Fire Tablets)”와 “테스트를 위해 파이어 태블릿 연결하기(Connecting Your Fire Tablet for Testing)”를 찾았다. 여기서는 킨들 파이어 기기를 안드로이드 스튜디오에 연결하는 과정을 간단히 요약만 하고 넘어간다. 더욱 자세한 정보가 필요하다면 아마존 가이드를 참조하자.

먼저 필자 같은 윈도우 사용자들은 킨들 파이어 태블릿을 개발 컴퓨터에 연결할 때 설치되는 비 ADB(Android Debug Bridge) 드라이버를 먼저 제거해야 한다(아직 ADB를 활성화하면 안 됨). 그런 다음 아마존의 USB 드라이버를 설치한다.

그다음 킨들 파이어 USB 드라이버를 다운로드한다. 다운로드되는 ZIP 압축 파일에는 **KindleDrivers.exe** 애플리케이션이 포함되어 있다.

KindleDrivers.exe를 실행하고 표시되는 안내에 따른다. 필자의 경우 **C:\Program Files (x86)\Amazon.com\Fire_Devices\Drivers** 디렉터리가 생성되고 그 안에 필요한 드라이버 파일이 설치됐다.

드라이버를 설치한 후 태블릿에서 ADB를 활성화해야 한다. 이후에는 태블릿을 개발용 컴퓨터에 연결한다. 태블릿을 안드로이드 스튜디오에 연결하는 방법에 대한 추가 지침은 아마존 가이드에 나와 있다.

모든 준비가 끝나면 안드로이드 스튜디오를 시작하고 W2A 프로젝트를 로드하고 W2A 앱을 실행한다. 이번에는 ‘배포 대상 선택>Select Deployment Target’ 대화 상자의 연결된 기기(Connected Devices) 부분에 ‘아마존 KFTT(Amazon KFTT)’ 항목이 표시될 것이다. 이 항목을 선택하고 ‘확인(OK)’을 클릭한다. 그러면 그레들이 앱 빌드를 시작한다.

찾을 수 없는 다양한 API 유형에 대한 여러 가지 오류 메시지가 뜨기는 했지만, 필자의 기기에 APK가 성공적으로 설치되었고 자동으로 앱이 실행됐다.

그림 36에서 앱 런처 화면의 W2A 앱 아이콘을 볼 수 있다.

그림 37은 실행 중인 W2A 앱의 모습이다.

3부 맷음말

첫 안드로이드 스튜디오 애플리케이션을 작성하고 빌드하고 실행까지 했으니, 그다음엔 무엇을 해야 할지 궁금할 것이다. 일단은 배운 내용으로 이것저것 실험을 해볼 것을 권한다. 3편의 입문 강좌에서 제공된 예제와 소스 코드를 사용해서 직접 새 프로젝트를 개발해 보는 것이 좋다. 안드로이드 스튜디오의 여러 특성과 기본 기능을 익히는 동안에는 프로젝트를 간결하게 유지하는 것이 좋지만, 더 나아가 실험을 통해 스스로 과제를 내고 해결해 보는 것도 실력 향상에 도움이 될 것이다. ITWORLD