부록 E. 안드로이드 SDK 1.6 소개

류광

참고

이 부록은 곧 출간될 시작하세요! 안드로이드 프로그래밍[1](원서는 Android Wireless Application Development[2]의 일부입니다. 원서에는 없던 것으로, 원서가 나온 이후 발표된 안드로이드 SDK 1.6의 내용을 보강하기 위해 2009년 10월 중순에 역자 류광[3]이 안드로이드 개발자 사이트(http://developer.android.com/)의 공식 문서들과 안드로이드 관련 여러 블로그들을 참고해서 작성한 것입니다.

코드명 "도넛(donut)"으로 지칭되던 안드로이드 플랫폼 및 SDK의 차기 릴리스가 2009년 9월 말 1.6으로 공개되었다. 참고로 1.5의 코드명은 컵케이크(cupcake)였다. 도넛이 2009년 말경에 안드로이드 2.0으로 나올 것이라는 예상도 있었으나, 실제로는 좀 더 일찍 1.6으로 등장했다. 번호 차이에서 짐작하겠지만 1.5와 1.6의 차이는 1.1과 1.5의 차이에 비하면, 그리고 1.5와 사람들이 추측하던 2.0의 차이에 비하면 사소한(물론 상대적으로) 편이다. 그래도 몇 가지 주목할 만한 개선점들이 있다. 이 부록에서는 안드로이드 SDK 1.6의 특징을 소개하고 기존 개발 환경과 프로젝트를 1.6에 맞게 갱신하는 방법을 이야기한다.역주

안드로이드 1.6의 새 기능과 그에 따른 SDK 변화

안드로이드 플랫폼 1.6에 여러 가지 새로운 기능들이 추가되었으며, 그에 따라 SDK에도 변화가 생겼다. 다음은 주요 변화들이다. 리눅스 커널이 2.6.29로 업그레이드된 점 등 SDK에 직접적인 영향을 미치지는 않은 변경 사항들은 생략했으니, 그런 사항들에 대해서는 http://developer.android.com/sdk/android-1.6-highlights.html을 보기 바란다.

범용 검색 수단 "Quick Search Box"

플랫폼에 안드로이드 시스템 전체 검색 및 웹 연동 검색 기능을 제공하는 "Quick Search Box" 기능이 추가되었다. 이에 따라 응용프로그램이 자신의 자료를 Quick Search Box를 통해서 사용자에게 제시할 수 있는 수단들이 SDK에 추가되었는데, 특히 `android.app.SearchManager` 클래스에 여러 멤버들이 새로 생겼다. 좀 더 자세한 내용은 이 클래스의 레퍼런스 문서(특히 Exposing Search Suggestions to Quick Search Box 섹션)를 참고하기 바란다.

CDMA 지원

플랫폼이 GSM은 물론 CDMA도 지원하게 되었으며, 이에 의해 `android.telephony.gsm` 패키지가 폐기 대상(deprecated)이 되었다. 이제는 `android.telephony` 패키지가 전반적인 전화통신 기능(GSM과 CDMA 모두)을 담당한다. 특히, `android.telephony.gsm` 패키지의 `SmsManager`, `SmsMessage`, `SmsMessage.SubmitPdu` 클래스가 `android.telephony` 패키지로 올라갔다.

접근성 프레임워크

플랫폼에 접근성(accessibility) 프레임워크가 추가되었다. 예를 들어 다른 창으로의 전환 등의 시각적인 변화를 소리나 진동으로도 사용자에게 알려주는 행동 방식을 좀 더 쉽고 표준적으로 구현할 수 있다. 이를 지원하기 위해 SDK에 `android.view.accessibility` 패키지와 `android.accessibilityservice` 패키지가 추가되었다.

TTS 지원

접근성 프레임워크와도 관련이 있는 기능으로, 텍스트를 음성으로 변환해서 출력하는 TTS 엔진이 플랫폼에 추가되었다. TTS 엔진은 영어(미국, 영국 악센트), 프랑스어, 이탈리아어, 독일어, 스페인어를 기본으로 지원한다. TTS 기능을 위해 `android.speech.tts` 패키지가 SDK에 새로 생겼다.

제스처 입력

사용자의 특정 손동작을 특정 작용에 연관시키는 제스처 기능이 플랫폼에 추가되었으며, 이를 위해 SDK에 android.gesture 패키지가 추가되었다. 또한 제스처 정의를 돕는 도구 GestureBuilder도 추가되었다.

더욱 다양한 화면 크기/밀도 지원

기존의 QVGA와 HVGA 외에 WVGA800, WVGA854를 지원하게 되었다. 화면 밀도(density)가 다양해짐에 따라 Bitmap이나 Canvas, DisplayMetrics 클래스 등에 밀도에 관련된 멤버들이 추가되었다(이를테면 Bitmap.getDensity() 등). 또한, 안드로이드 매니페스트 파일에 응용프로그램이 지원하는 화면 크기들을 지정하기 위한 <supports-screens> 요소가 새로 생겼다. 한편, 화면 종류가 다양해지면서 에뮬레이터 테두리 모습들을 일일이 흉내내기가 힘들어서인지 에뮬레이터 창 자체는 훨씬 단순해졌다(그림 E.1).

그림 E.1. 새로운 에뮬레이터의 모습

추가적인 애니메이션 기능

플랫폼에 새로운 종류의 애니메이션들이 추가됨에 따라 android.view.animation 패키지에 다음과 같은 클래스들이 추가되었다.

- AnticipateInterpolator
- AnticipateOvershootInterpolator
- BounceInterpolator
- OvershootInterpolator

그 외의 SDK 변화

새 기능과는 무관한 여러 변화들이 있다. 몇 가지 주목할 만한 것들을 들자면 다음과 같다.

- android.opengl 패키지에 GLES10, GLES10Ext, GLES11, GLES11Ext 클래스가 추가되었다. 개발자 문서화에 명확한 언급은 없지만 아마도 javax.microedition.khronos.opengles 패키지에 있던 OpenGL ES 구현 클래스들(GL10, GL10Ext, GL11, GL11Ext)이 android.opengl 패키지로 옮겨진 것으로 보인다. 그러나 아직 API가 완전히 정리되지는 않은 것 같다. 예를 들어 android.opengl.GLU의 메서드들은 여전히 khronos.opengles의 형식들을 사용한다.

- 뷰 위젯의 View.OnClickListener를 코드에서는 물론 레이아웃 XML 파일에서도 지정할 수 있게 되었다. 이를 위해 android:onClick 특성이 추가되었다.

- 안드로이드 매니페스트 파일에서 응용프로그램에 꼭 필요한 장치나 기능을 지정하는 데 사용하는 〈uses-feature〉 요소가 생겼다. 또한, 〈uses-sdk〉 요소에는 응용프로그램의 주된 대상 SDK 버전과 응용프로그램이 지원할 수 있는 최대 SDK 버전을 지정하는 targetSdkVersion 특성과 maxSdkVersion 특성이 새로 생겼다(이전에는 응용프로그램의 실행에 필요한 최소 SDK 버전만 지정할 수 있었다).

- 다음과 같은 권한 요청 식별자들이 추가되었다: CHANGE_WIFI_MULTICAST_STATE(WiFi 멀티캐스트 모드 진입), GLOBAL_SEARCH(전역 검색 시스템이 특정 콘텐트 제공자의 자료에 접근), INSTALL_LOCATION_PROVIDER(위치 관리자에 새 위치 제공자를 설치), READ_HISTORY_BOOKMARKS(사용자의 브라우징 내력과 북마크들에 읽기 전용 접근), WRITE_HISTORY_BOOKMARKS(사용자의 브라우징 내력과 북마크들에 쓰기 전용 접근), WRITE_EXTERNAL_STORAGE(외부 저장장치에 쓰기 접근). 1.5에서는 이들이 암묵적으로 허용되지만, 1.6(API 수준 4)부터는 응용프로그램이 이들을 명시적으로 요청해야 한다.

안드로이드 SDK 1.6 적응을 위한 안드로이드 개발자 문서화 활용법

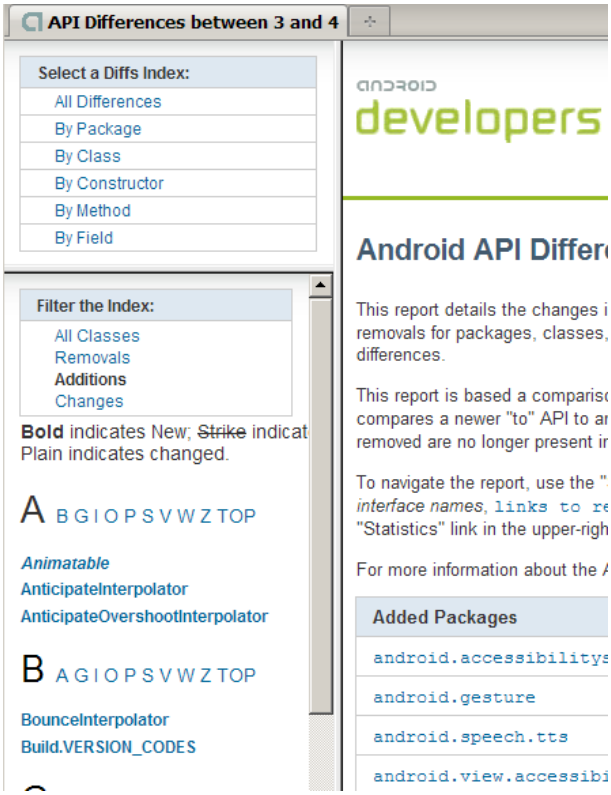
위에서 나열한 것 외에도 크고 작은 여러 변화가 있는데, 무엇보다도 참고해야 할 것은 안드로이드 개발자 사이트의 공식 문서화이다. 우선 다음 두 문서에서 전반적인 사항을 파악할 수 있다.

- Android 1.6 Platform Highlights(http://developer.android.com/sdk/android-1.6-highlights.html)

- Android 1.6 Version Notes(http://developer.android.com/sdk/android-1.6.html)

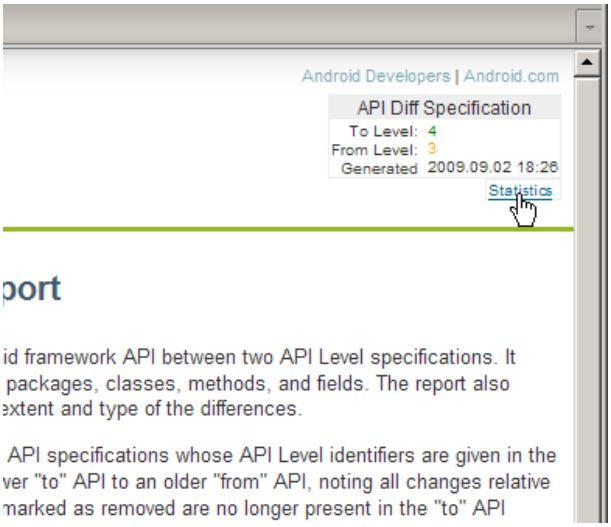
다음으로, API의 세세한 변경 사항은 Android API Differences Report(http://developer.android.com/sdk/api_diff/4/changes.html)에서 볼 수 있다. 특히 이 페이지의 왼쪽 내비게이션 영역에서 단위(패키지, 클래스 등)별 변경 사항을 일목요연하게 알아볼 수 있다. 그림 E.2는 추가된 클래스들만 나열되게 한 모습이다.

그림 E.2. Android API Differences Report 페이지의 왼쪽 상단 모습. 새로 추가된 클래스들만 나열했다.



또한, Android API Differences Report 페이지의 오른쪽 상단 링크(그림 E.3)에서 진입할 수 있는 API Change Statistics 페이지도 유용하다. 이 페이지에서는 변경된 패키지들과 클래스 및 인터페이스들을 변경 비율 순으로 나열해준다. 또한 각 클래스나 패키지 이름을 클릭하면 바뀐 클래스들이나 멤버들이 나타난다. 기존 프로젝트에 쓰이는 패키지나 클래스들에서 무엇이 얼마나 변했는지를 파악하는 데 도움이 될 것이다.

그림 E.3. Android API Differences Report 페이지의 오른쪽 상단 모습. API Change Statistics 페이지로 가는 링크가 있다.



개별 레퍼런스 페이지에서도 API의 변화를 파악할 수 있다. 레퍼런스 페이지 오른쪽 상단에 보면 Filter By API Level: 이라는 이름표가 붙은 드롭다운 목록이 있는데, 여기서 3(SDK 1.5에 해당)을 선택하면 1.6에만 해당하는 사항들이 흐리게 표시된다. 예를 들어 `android.telephony` 패키지에 대한 레퍼런스 페이지에서 3을 선택하면 `SmsManager` 클래스 등이 흐리게 나타난다.

안드로이드 SDK와 Eclipse 개발 환경 업데이트

안드로이드 SDK 1.6을 설치하고 Eclipse 개발 환경을 SDK 1.6에 맞게 갱신하는 방법을 살펴보자.

안드로이드 SDK 1.6 설치

안드로이드 SDK 1.6 설치는 원칙적으로 안드로이드 SDK 1.5와 무관하다. 즉, 안드로이드 SDK 1.5가 설치된 곳에 새로운 파일들을 덮어 씌워서 SDK 1.6을 설치하는 것이 아니라 1.5와는 개별적인 디렉터리에 새로이 SDK를 설치하는 것임을 주의하기 바란다. 아마도 1.6 이후 버전은 같은 SDK 디렉터리의 platforms 디렉터리에 새로운 버전을 추가하는 식으로 업데이트가 진행될 가능성이 크지만(아래 "AVD 만들기" 참고), 어쨌든 지금은 그냥 새로운 SDK를 따로 설치한다고 생각하면 된다. SDK의 설치 자체는 아주 간단하며, 사실 제3장에 나온 과정과 완전히 동일하다. 1. 안드로이드 개발자 사이트의 다운로드 페이지에서 압축 파일을 내려 받아서 원하는 디렉터리에 푼다. SDK 1.6 r1의 경우 다운로드 페이지는 http://developer.android.com/sdk/1.6_r1/index.html이다. 2. 안드로이드 SDK를 설치한 디렉터리의 `tools` 디렉터리를 시스템의 환경 변수 PATH에 추가한다. 운영체제별 구체적인 방법은 제3장에 나와 있다. PATH에서 안드로이드 SDK 1.5의 `tools` 디렉터리는 제거하는 것이 안전하다(적어도 1.6의 것이 더 먼저 나오게는 해야 한다).

ADT 플러그인과 DDMS 갱신 및 Eclipse 설정 변경

SDK와 함께 Eclipse용 ADT 플러그인과 DDMS도 0.9.3으로 판올림되었다. 이 번역서가 나온 후에 제3장에 나온 과정을 따라 ADT 플러그인과 DDMS을 처음 설치했다면 이미 0.9.3(또는 그 이상)이 설치되어 있는 것이다. 아래의 내용은 그렇지 않은 독자에게만 해당된다. 한 가지 중요한 점은, ADT 0.9.3은 더 이상 Eclipse 3.3을 더 이상 지원하지 않는다는 점이다(적어도 공식적으로는). 혹시 3.3 Europa를 사용하고 있다면 이 기회에 3.4(Ganymede)나 3.5(Galileo)를 설치하기 바란다.

ADT(그리고 DDMS)를 갱신하는 방법은 http://developer.android.com/sdk/1.6_r1/upgrading.html#UpdateAdt에 잘 나와 있다. 요약하자면 다음과 같다.

Eclipse 3.4 (Ganymede):

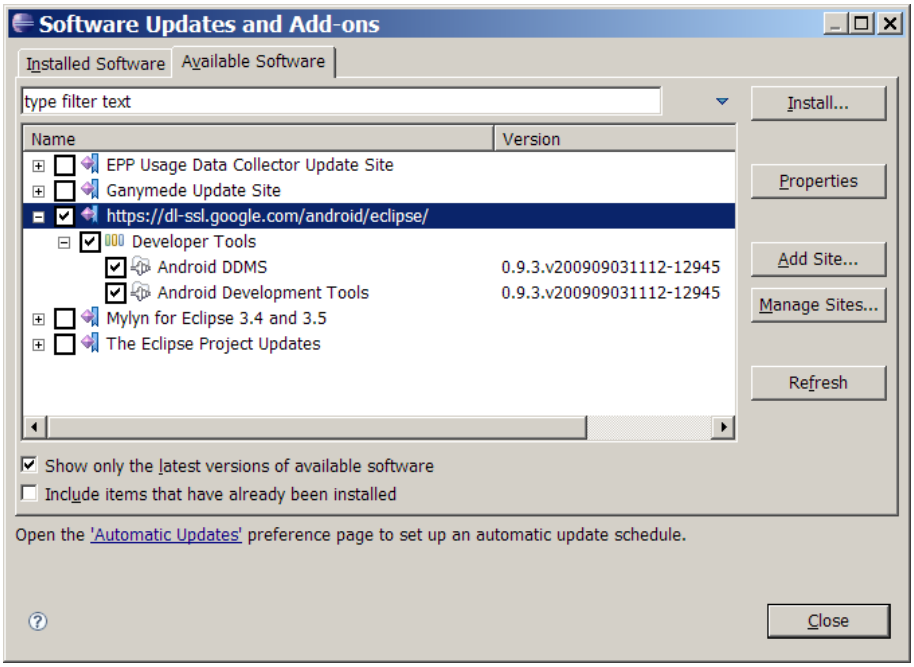
1. Eclipse의 주 메뉴에서 Help - Software Update를 선택하고 Available Software 탭을 선택한다.

2. 트리에서 https://dl-ssl.google.com/android/eclipse/ 항목의 체크상자를 체크한다. 트리를 펼쳐서 ADT와 DDMS가 모두 체크되었는지 확인하는 것이 좋겠다(그림 E.4).

3. Install 버튼을 클릭하면 설치 과정이 시작된다. 지시에 따라 설치 과정을 진행하면 된다.

4. . 설치가 끝나면 Eclipse를 재시동한다.

그림 E.4. Eclipse용 ADT 플러그인과 DDMS 갱신.



Eclipse 3.5 (Galileo):

1. Eclipse의 주 메뉴에서 Help - Check for Updates를 선택한다.

2. Available Updates 대화상자의 목록에서 Android DDMS와 Android Development Tools를 찾아서 해당 체크상자들을 체크한다(만일 Android DDMS와 Android Development Tools가 목록에 없다면, Help - Install New Software에서 원격 사이트 https://dl-ssl.google.com/android/eclipse/를 적절히 추가한 후 다시 시도하기 바란다).

3. Next 버튼을 클릭하면 설치 과정이 시작된다. 지시에 따라 설치 과정을 진행하면 된다.

4. 설치가 끝나면 Eclipse를 재시동한다.

마지막으로 할 일은 새 SDK의 위치를 Eclipse에게 알려주는 것이다. 역시 제3장에 나온 것과 다를 바 없다. (이 과정은 Ganymede와 Galileo가 동일하다.)

1. Eclipse의 주 메뉴에서 Window(Mac OS X의 경우에는 Eclipse)를 선택하고 Preference를 선택한다.

2. Preference 대화상자의 왼쪽 트리 뷰에서 Android를 선택하고, 오른쪽의 SDK Location에 앞에서 설치한 안드로이드 SDK의 경로를 설정한다.

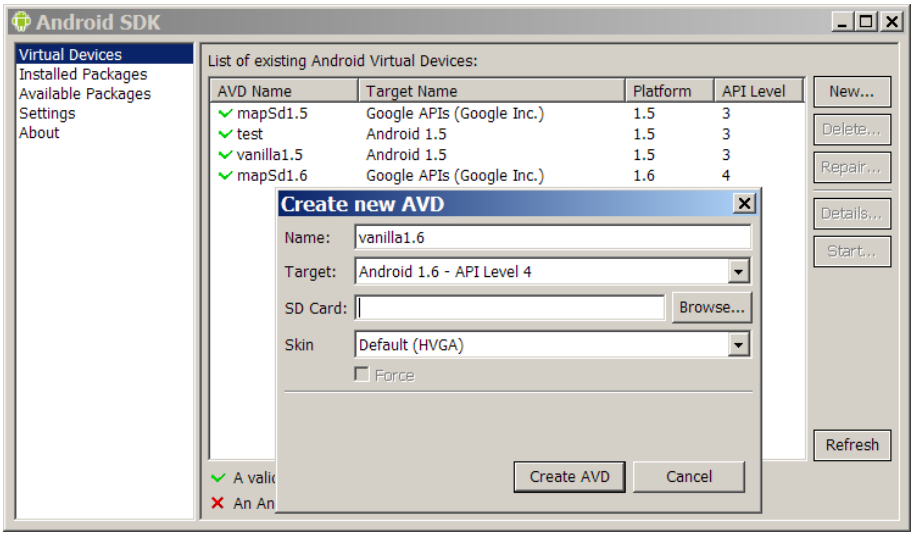
AVD 만들기

안드로이드 1.6의 새 기능을 에뮬레이터에서 시험해 보려면 1.6 시스템 이미지를 포함하는 AVD를 만들어야 한다. AVD를 만드는 방법은 1.5에서와 별로 달라진 것이 없다. 단지 대상 ID들과 스킨 종류가 다를 뿐이므로 부록 A의 내용이 여전히 유효하다.

그림 E.5는 명령 프롬프트에서 SDK 디렉터리의 `tools` 디렉터리로 가서 아무 인수 없이 `android`를 실행하거나 Eclipse 안에서 주 메뉴의 Windows - Android SDK and AVD Manager를 선택했을 때 나타나는 AVD 및 SDK 관리자를 이용해서 새 AVD를 만드는 모습이다.

참고로, SDK 1.5에서도 이와 비슷한 AVD 관리 도구가 있었으나, SDK 1.6에서는 SDK 관리 기능이 추가되었다. 이후의 새 SDK는 따로 압축 파일을 내려 받는 대신 이 관리자를 통해서 플랫폼 버전별 SDK들을 좀 더 간편하게 설치 또는 갱신할 수 있을 것이다.

그림 E.5. 새로운 AVD 관리자



기존 프로젝트를 1.6에 맞게 이전하기

일반적으로 응용프로그램의 SDK 버전 간 이전은 상당히 까다로운 일일 수 있다. 여기에서는 1.5용 프로젝트를 1.6에 맞게 갱신하는 데 관련된 아주 기본적인 사항들만 이야기한다.

코드에 손을 대기 전에 먼저 할 일은 1.6으로의 이전이 꼭 필요한가를 따지는 것이다. 구체적인 기준은 프로젝트나 개발자/개발사 사정에 따라 다를 것이다. 또한, 이전 과정에서 코드를 크게 뜯어고쳐야 할 수도 있으므로 소스 코드 관리 시스템 상에서 새로운 가지나 태그로 코드를 분기해 두는 것도 빼먹지 말아야 한다.

관련 준비가 끝났다고 할 때, 첫 번째로 할 일은 프로젝트의 빌드 대상을 변경하는 것이다. 프로젝트 속성 페이지(패키지 탐색기에서 프로젝트를 오른쪽 클릭 후 Properties를 선택)의 왼쪽 목록에서 Android를 선택하고, Project Build Target에서 Android 1.6 또는 Google APIs 1.6을 선택한다. 패키지 탐색기의 프로젝트 트리에 Android 1.5 대신 Android 1.6이라는 문구가 나타나면 제대로 된 것이다.

다음으로는 디버그 구성(Debug Configuration)과 실행 구성(Run Configuration)이 1.6용 AVD를 사용하게 해야 한다. 해당 구성 대화상자의 Target 탭에서 1.6용으로 만든 새 AVD를 선택하면 된다. 디버그 구성 또는 실행 구성을 만드는 방법은 제3장에서 이야기했다.

여기까지 마쳤다면 프로젝트를 빌드해 본다. 편집창 아래의 Problems 탭에 여러 가지 경고(Warning)들이 나와 있을 것이며, 경우에 따라서는 빨간 색 오류(Error) 메시지도 있을 수 있다. 오류들은 반드시 해결해야 한다. 경고들은 무시할 수도 있으나 프로젝트의 건강을 위해서는 해결해 주는 것이 바람직하다. 구체적인 방법은 프로젝트에 따라 다르겠지만, 이 부록 앞부분에 나온 SDK 변경 사항들을 출발점으로 삼고 안드로이드 개발자 사이트의 온라인 문서화에서 구체적인 사항들을 짚어 나간다면 크게 어려운 일은 아닐 것이다. 본문 제13장의 Telephony 예제를 예로 들어 보겠다.

사례: 제13장의 Telephony 예제

"안드로이드 SDK와 Eclipse 개발 환경 업데이트" 절에 나온 것처럼 SDK와 Eclipse 개발 환경을 갱신했으며, Telephony 프로젝트를 열고 빌드 대상과 실행 구성 등을 방금 전에 말한 것처럼 갱신했다고 하겠다. 이제 프로젝트를 빌드해 보자(이를테면 패키지 탐색기에서 Telephony를 선택한 후 오른쪽 클릭 메뉴에서 Build Project 선택해서). 패키지 탐색기의 Telephony 프로젝트에 노란 경고 아이콘이 나타날 것이며, 편집창 아래의 Problems 탭을 보면 20개가 넘는 경고들이 보일 것이다.

안드로이드 매니페스트 파일 갱신

첫 번째 경고는 프로젝트의 최소 SDK 버전이 프로젝트의 대상 API 수준보다 낮다는 것이다. 예를 들어 1.6에만 있는 클래스를 사용하는 응용프로그램을 안드로이드 1.5 플랫폼을 운영하는 기기에 설치해서 실행하려 하면 문제가 생길 것이므로 이런 경고가 나오는 것이다. 해결책은 프로젝트의 빌드 대상을 다시 API 수준 3(SDK 1.5)으로 되돌리거나, 안드로이드 매니페스트 파일의 최소 SDK 버전을 4(SDK 1.6)로 수정하는 것이다. 여기에서는 후자의 방법을 택하기로 한다.

본문 제3장에서 배운 것처럼, 프로젝트의 AndroidManifest.xml 파일을 열고 Manifest 탭의 Manifest Extras에서 Uses SDK를 선택한 후 Min SDK version을 3에서 4로 수정한다. 그 부분을 보면 이전과는 달리 Target SDK version과 Max SDK version이라는 필드가 새로 생겼음을 알 수 있다. 전자는 이 프로젝트가 주된 대상으로 삼는 SDK 버전(API 수준)이다. 역시 4로 설정하면 된다. 후자는 이 프로젝트가 지원하는 최대 SDK 버전(역시 API 수준)인데, 최대 SDK를 제한할 필요가 있는 경우에만 설정하면 된다. 이제 AndroidManifest.xml 파일을 저장하고 다시 프로젝트를 빌드하면 첫 번째 경고가 사라진다.

SDK 변화에 따른 소스 코드 수정

이제 Problems 탭의 나머지 경고들을 보자. 나머지 경고들에는 모두 "deprecated"는 단어가 있는데, 이는 해당 형식이나 메서드가 폐기 대상이 되었으며 따라서 이후의 SDK에서는 사라질 수 있음을 뜻한다. 경고 메시지를 더블 클릭하면 소스 편집창에서 경고가 발생한 줄이 선택된다. "The type SmsMessage is deprecated"를 더블 클릭해 보면 SmsManager가 폐기 대상임을 시각적으로도 확인할 수 있다(취소선이 그어져 있다). 폐기 대상인 이 클래스를 다른 클래스로 대체해야 경고가 없어질 것이다.

어떤 클래스로 대체해야 할까? 웹 브라우저로 안드로이드 개발자 사이트의 문서화를 검색해서 android.telephony.gsm.SmsManager의 레퍼런스 페이지를 보면 "This class is deprecated.Replaced by android.telephony.SmsManager that supports both GSM and CDMA."라는 문구가 나온다. 답이 나온 셈이다. Problems 탭의 Resource 열을 클릭해 보면 경고가 있는 파일이 MakeCall.java와 SMSSender.java임을 알 수 있다. 두 Java 파일 모두에서

```
import android.telephony.gsm.SmsManager;
```

를

```
import android.telephony.SmsManager;
```

로 고치고 프로젝트를 빌드해 보자. 경고들이 많이 사라졌을 것이다.

나머지 경고들은 SmsMessage 에 대한 것들이다. 역시 같은 방식으로 개발자 문서화를 살펴보면 android.telephony.gsm.SmsMessage가 폐기 대상이 되었고 대신 android.telephony.gsm.SmsMessage를 사용해야 함을 알 수 있다. 방금 전처럼 두 Java 파일에서

```
import android.telephony.gsm.SmsMessage;
```

를

```
import android.telephony.SmsMessage;
```

로 고치고 프로젝트를 빌드해 보자. Problems 탭에 경고가 모두 사라지고 패키지 탐색기의 Telephony 노드의 경고 아이콘도 사라졌을 것이다.

SmsMessage와 SmsManager가 내부적인 변화 없이 패키지 위치만 바뀌었기 때문에 비교적 쉽게 끝났지만, 만일 멤버들에 변화가 있었다면 좀 더 세밀한 수정이 필요했을 것이다. 또한, 이상의 과정은 단지 빌드 경고들을 없애기 위한 "구문론적" 처리였을 뿐이다. 이전된 프로젝트가 실제로 제대로 작동하는지는 에뮬레이터와 실제 기기에서 구체적으로 시험해 보아야 한다.