시작하기

- 안드로이드 애플리케이션 개발하기
- 모바일 애플리케이션 개발하기
- 할 일 목록 예제
- 안드로이드 개발 도구
- 요 약

여러분만의 안드로이드 애플리케이션을 작성하기 위해 필요한 것은 안드로이드 SDK와 자바 개발 킷JDK, Java Development Kit 한 벌이 전부다. 여러분이 마조히스트masochist가 아닌 이상은 대개 개발을 좀더 쉽게 해주는 자바 IDE(특히, 이클립스가 괜찮다)를 원할 것이다.

안드로이드 SDK와 자바 그리고 이클립스는 윈도우, 맥 OS, 리눅스를 위한 버전이 모두 마련되어 있으므로, 여러분이 선호하는 OS에서 편안하게 안드로이드를 살펴볼 수 있다. 안드로이드 SDK는 이 세 가지 OS 환경 모두를 위한 에뮬레이터를 가지고 있으며, 안드로이드 애플리케이션은 가상 머신 위에서 실행되기 때문에, 어떤 특정 운영체제로부터 오는 개발상의 이점은 없다고 보는 것이 좋다.

안드로이드 코드는 자바 문법으로 작성되며, 코어 안드로이드 라이브러리는 코어 자바 API가 가진 기능의 대부분을 포함한다. 하지만 이들이 실행될 수 있기 전에, 여러분의 프로젝트는 먼저 Dalvik 바이트 코드byte code로 번역된다. 결과적으로 여러분은 자바를 사용한다는 이점과 동시에, 여러분의 애플리케이션이 안드로이드 장치에 최적화된 가상 머신 위에서 실행되는 이점을 얻는다.

안드로이드 SDK에는 안드로이드 라이브러리와 풍부한 문서 그리고 뛰어난 샘플 애플리케이션이 포함되어 있다. 또한 SDK는 여러분의 프로젝트를 실행하기 위한 안드로이드 에뮬레이터와 디버그를 돕는 Dalvik 디버그 모니터링 서비스DDMS, Dalvik Debug Monitoring Service 같이, 애플리케이션을 작성하고 디버그하기 위한 도구 역시 포함하고 있다.

이번 장에서는 안드로이드 SDK를 다운로드받아 개발 환경을 구축하고, 두 가지 새로운 애플리케이션을 완성해, 이를 에뮬레이터와 DDMS로 실행하고 디버그해볼 것이다.

만일 여러분이 모바일 장치에서 동작하는 소프트웨어를 개발해본 경험이 있다면, 모바일 장치의 작은 크기와 제한된 전력 그리고 한정된 메모리가 만들어내는 독특한 몇 가지 설계상의 어려움에 대해 이미 잘 알 것이다. 여러분이 이 분야에 경험이 없다손 치더라도, 데스크톱이나 웹에서는 당연한 것들이 모바일에서는 일부 그럴 수 없다는 사실은 불을 보듯 뻔한 일이다.

이러한 하드웨어의 한계와 더불어 사용자 환경 역시 나름의 어려운 도전거리를 가져다준다. 모바일 장치는 이동하면서 사용될 뿐만 아니라, 집중하기 어려운 산만한 환경에서 사용되는 것이 보통이므로, 여러분의 애플리케이션은 빠르고 반응성이 좋아야 하며 사용하기에도 편리해야 한다.

이렇듯 타고난 하드웨어의 어려움과 환경적인 요인을 극복하도록 돕기 위해, 이번 장에서는 모바일 애플리케이션 작성을 위한 베스트 프랙티스best practices 몇 가지를 살펴본다. 전체 주제를 붙잡고 씨름하려 하기보다는, 좋은 모바일 설계 원칙에 부합하는 측면에서 바라본 안드로이드 SDK 사용에 초점을 맞출 것이다.

안드로이드 애플리케이션 개발하기

안드로이드 SDK는 매력적이면서도 강력한 모바일 애플리케이션을 작성하는 데 필요한 모든 도구와 API를 가지고 있다. 여느 새로운 개발 툴킷과 마찬가지로, 안드로이드의 가장 큰 난제는 바로 안드로이드가 가진 API의 특징과 한계를 배우는 것이다.

만일 여러분이 자바로 개발해본 경험이 있다면, 비록 몇몇 특유의 최적화 기법이 직관적이지 않아 보일지는 모르지만, 지금껏 사용해온 기법과 구문 그리고 문법이 안드로이드에 고스란히 녹아있음을 알게 될 것이다.

자바에는 경험이 없지만 다른 객체지향 언어(C# 같은)를 사용해본 적이 있다면, 쉽게 안드로이드로 넘어올 수 있음이 느껴져야 한다. 안드로이드의 힘은 그가 가진 API로부터 오는 것이지 자바로부터 오는 것이 아니므로, 자바 클래스에 익숙지 않다고 해서 큰 불이익이 되는 것은 아니다.

준비물

안드로이드 애플리케이션은 Dalvik 가상 머신에서 실행되므로, 안드로이드 개발자 도구를 지원하는 플랫폼이라면 어디서든 안드로이드 애플리케이션을 작성할 수 있다. 현재 지원되는 플랫폼은 다음과 같다.

- ✓ 마이크로소프트 윈도우(XP 또는 비스타)
- ✓ 맥 OS 10.4.8 이상(인텔 칩만)
- ✓ 리눅스

안드로이드 애플리케이션을 작성하기 위해서는 다음의 것들을 다운로드받아 설치해야 한다.

- ✓ 안드로이드 SDK
- ✓ 자바 개발 킷(JDK) 5나 6

최신 JDK는 아래 주소의 썬Sun 홈페이지에서 다운로드받을 수 있다.

```
http://java.sun.com/javase/downloads/index.jsp
```

JDK가 이미 설치되어 있다면, 위에 나열된 버전 요구사항을 만족하는지 확인하기 바란다. 자바 런타임 환경JRE, Java Runtime Environment만으로는 부족하다는 사실을 기억하자.

SDK 설치하기

안드로이드 SDK는 완전히 열려있다. API를 다운로드받아 사용하는 데 아무런 비용이 들지 않으며, 여러분이 만든 프로그램을 배포할 때에도 구글은 이에 대해 어떠한 비용도 부과하지 않는다. 여러분의 개발 플랫폼을 위한 SDK 최신 버전은 아래 주소의 안드로이드 개발자 홈페이지에서 다운로드받을 수 있다.

```
http://developer.android.com/sdk
```

달리 표시해두지 않는 한, 이 책을 쓰는 데 사용된 안드로이드 SDK 버전은 1.0 r1이다.[1]

안드로이드 SDK는 API 라이브러리, 개발자 도구, 문서, 그리고 여러 샘플 애플리케이션과 함께 특정 API 기능의 사용을 조명하는 API 데모를 한데 담은 ZIP 파일로 제공된다. 이 압축 파일을 원하는 폴더에 푸는 것으로 SDK 설치가 완료된다(이 폴더의 위치는 나중에 필요하니 적어두자).

안드로이드 SDK가 제공하는 예제와 단계별 지침은 안드로이드 개발자 도구ADT, Android Developer Tool 플러그 인이 설치된 이클립스를 사용하는 개발자를 대상으로 하고 있다. 하지만 반드시 이들을 사용해야 하는 것은 아니다. 여러분의 입맛에 맞는 텍스트 에디터나 자바 IDE를 사용할 수 있으며, SDK에 있는 개발자 도구를 사용해 코드와 샘플 애플리케이션을 컴파일, 테스트, 디버그할 수 있다.

만일 이클립스와 ADT 플러그 인을 여러분의 안드로이드 개발 환경으로 사용할 계획이라면, 다음 절에 설명된 설정 방법을 참고하자. 이번 장 후반부에서는 SDK에 포함된 개발자 도구 역시 자세히 살펴볼 것이므로, 이클립스나 ADT 플러그 인을 사용하지 않고 개발하고자 하는 경우에는 이 부분을 참고하자.

안드로이드 SDK에 포함되어 있는 예제는 완전히 기능하는 안드로이드 애플리케이션으로서, 문서화 또한 잘 되어 있다. 개발 환경설정을 마친 뒤 한번쯤 살펴보는 것도 도움이 될 것이다.

역주 ❶ 이 책을 옮기는 데 사용한 안드로이드 SDK 버전은 2009년 5월에 발표된 1.5 r2이다.

이클립스로 개발하기

안드로이드 개발 환경으로서 이클립스와 ADT 플러그 인의 조합은 몇 가지 커다란 이점을 가져다 준다.

이클립스Eclipse는 특히 자바 개발 진영에서 널리 사용되고 있는 오픈 소스 IDE(Integrated Development Environment통합 개발 환경)다. 이클립스 재단Eclipse foundation 홈페이지를 통해 다운로드받을 수 있으며, 안드로이드가 지원하는 모든 개발 플랫폼(윈도우, 맥 OS, 리눅스)에서 사용할 수 있다.

```
http://www.eclipse.org/downloads
```

위 페이지에 가보면 다운로드받을 수 있는 이클립스의 종류가 매우 다양한데, 이 가운데 안드로이드를 위한 권장 구성은 다음과 같다.

- ✓ 이클립스 3.3, 3.4(Ganymede)❷
  - ✓ 이클립스 JDT 플러그 인
  - ✓ WST

WST와 JDT 플러그 인은 대부분의 이클립스 IDE 패키지에 포함되어 있다.

다운로드받은 압축 파일을 원하는 폴더에 푸는 것으로 이클립스 설치가 완료된다. 설치를 완료했으면 `Eclipse` 실행파일을 실행하자. 이클립스를 처음 실행하는 경우에는, 안드로이드 개발을 위한 새로운 워크스페이스workspace 하나를 만들자.

이클립스 플러그 인 사용하기

ADT 이클립스 플러그 인은 에뮬레이터와 .class-to-.dex 변환기 등의 개발자 도구를 IDE에 직접 통합함으로써 안드로이드 개발을 단순화한다. 반드시 ADT 플러그 인을 사용해야 하는 것은 아니지만, ADT 플러그 인은 여러분의 애플리케이션을 보다 빠르고 손쉽게 만들어 테스트하고 디버그할 수 있게 해준다.

ADT 플러그 인은 다음과 같은 것들을 이클립스에 통합한다.

역주 ❷ 이 책은 이클립스 3.3(Europa)을 기준으로 설명한다.

- ✓ 새로운 프로젝트 생성을 단순화하며 기본적인 애플리케이션 템플릿을 가지고 있는 안드로이드 프로젝트 마법사Android Project Wizard
- ✓ XML 리소스의 생성, 편집, 검증을 돕는 폼 기반의 매니페스트manifest와 레이아웃layout 그리고 리소스 에디터
- ✓ 안드로이드 프로젝트의 자동화된 빌드, 안드로이드 실행파일(.dex)로의 변환, 패키지 파일(.apk)로의 패키징, 그리고 Dalvik 가상 머신에 패키지 설치
- ✓ 에뮬레이터의 겉모양과 네트워크 연결 설정을 제어하고, 걸려오는 전화와 SMS 메시지를 시뮬레이션할 수 있는 기능을 갖춘 안드로이드 에뮬레이터
- ✓ 포트 포워딩port forwarding, 스택, 힙, 스레드 뷰, 프로세스 세부사항, 화면 캡쳐 기능을 가진 Dalvik 디버그 모니터링 서비스DDMS, Dalvik Debug Monitoring Service
- ✓ 폴더 트리를 탐색하고 파일 전송을 가능케 하는 장치나 에뮬레이터의 파일시스템 접근
- ✓ 브레이크포인트breakpoints를 설정하고 호출 스택을 볼 수 있게끔 해주는 런타임 디버깅
- ✓ 안드로이드/Dalvik의 모든 로그와 콘솔 출력

그림 2-1은 ADT 플러그 인이 설치된 이클립스에서 DDMS 퍼스펙티브perspective를 보인 모습이다.

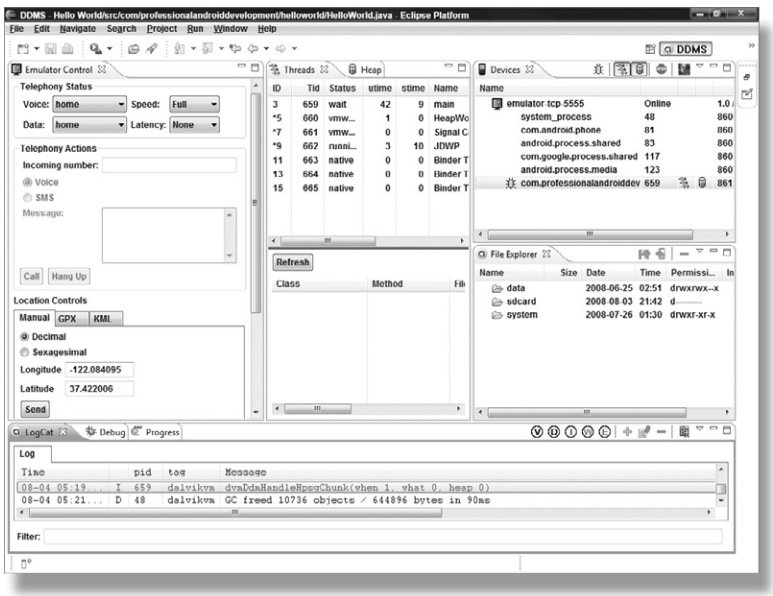

그림 2-1

ADT 플러그 인 설치하기

아래의 단계를 따라 개발자 도구 플러그 인을 설치하자.

1. 이클립스에서 Help → Software Updates → Find and Install...을 선택한다.
2. 다이얼로그 박스가 화면에 나타나면 Search for new features to install설치할 새로운 기능 검색을 선택한다.
3. New Remote Site새 원격 사이트를 선택한 뒤, 그림 2-2와 같이 다이얼로그 박스 안에 아래의 주소를 입력한다.

   `https://dl-ssl.google.com/android/eclipse/`❸
4. 입력한 새로운 사이트를 체크하고 Finish를 클릭한다.
5. 이제 이클립스가 ADT 플러그 인을 다운로드할 것이다. 다운로드가 끝나면 나타나는 Search Results검색 결과 다이얼로그 박스에서 Android Plugin → Developer Tools를 선택하고 Next를 클릭한다.

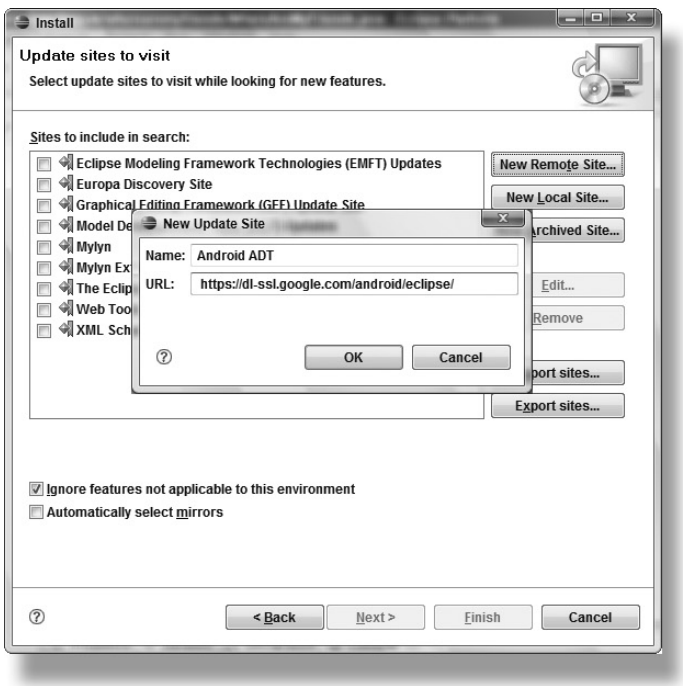

그림 2-2

역주 ❸ 만일 위 주소가 동작하지 않는다면 https를 http로 바꿔 입력해보자.

6. 라이선스 계약 조건을 살펴보고 승인한 뒤 Next와 Finish를 차례로 클릭한다. ADT 플러그 인은 서명되어 있지 않으므로 설치 진행 전 이에 대한 메시지가 표시될 것이다.
7. 작업이 완료되면 이클립스를 재시작하고 ADT 설정을 업데이트해야 한다. 이클립스를 재시작한 뒤 Window → Preferences... (맥 OS의 경우에는 Eclipse → Preferences)를 선택한다.
8. 그런 다음 왼쪽 패널에서 Android를 선택한다.
9. 그림 2-3에서 보이는 것처럼, Browse...를 클릭해 압축을 풀어놓은 안드로이드 SDK 폴더를 찾은 다음 Apply와 OK를 클릭한다.

새로운 버전의 SDK를 다운로드받아 이를 다른 위치에 두는 경우에는, 이 설정을 업데이트해 어떤 ADT가 빌드에 사용되어야 하는지를 SDK에 반영할 필요가 있다.

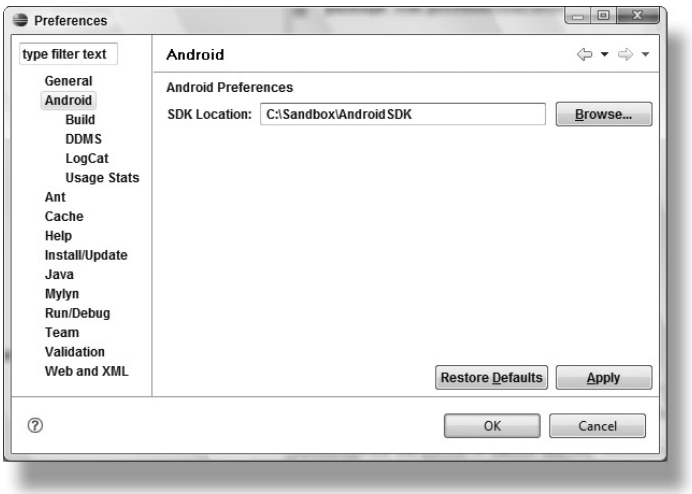

그림 2-3

플러그 인 업데이트하기

안드로이드 SDK가 발전해 나감에 따라 ADT 플러그 인의 업데이트 또한 잦아질 것이다. 여러분이 설치한 플러그 인을 업데이트하는 방법은 대부분 아래와 같이 간단히 이뤄진다.

1. Help → Software Updates → Find and Install...을 클릭한다.
2. Search for updates of the currently installed features현재 설치된 기능의 업데이트 검색를 선택하고 Finish...를 클릭한다.

3. 적용할 수 있는 ADT 업데이트가 존재하면 화면에 나타날 것이다. 이 가운데 원하는 것을 선택하고 Install을 클릭한다.

때로는 이러한 동적 업데이트 메커니즘을 사용할 수 없을 만큼 플러그 인 업그레이드가 중요한 경우도 있다. 이럴 때는 앞 절에서 설명한 대로 새로운 버전을 설치하기에 앞서 이전 플러그 인을 완전히 제거해야만 할 것이다.[4]

첫 번째 안드로이드 액티비티 만들기

SDK를 다운로드받았고 이클립스와 플러그 인도 설치했다. 이제 안드로이드 프로그래밍을 시작할 준비가 됐다. 먼저 새로운 프로젝트 하나를 생성한 다음, 이클립스의 실행 및 디버그 구성을 설정하는 것으로 시작하자.

새로운 안드로이드 프로젝트 시작하기

새 안드로이드 프로젝트 마법사를 통해 새로운 안드로이드 프로젝트를 만드는 방법은 아래와 같다.

1. File → New → Project를 선택한다.
2. Android 폴더에서 애플리케이션 타입을 Android Project로 선택하고 Finish를 클릭한다.
3. 화면에 나타나는 다이얼로그(그림 2-4)에 새로운 프로젝트에 관한 세부사항을 입력한다. "Project name"에는 프로젝트 파일 이름을, "Package name"에는 프로젝트의 패키지를, "Activity name"에는 초기 액티비티 클래스 이름을, "Application name"에는 맘에 드는 애플리케이션 이름을 입력하자.
4. 세부사항을 입력하고 난 뒤 Finish를 클릭한다.

역주 ❹ 안드로이드 SDK 1.5 r1이 발표되면서 그에 따라 ADT 플러그 인도 업데이트되었다. 안드로이드 SDK 1.5 r1을 사용하고자 한다면, 기존에 설치된 ADT 플러그 인을 제거한 뒤 "ADT 플러그 인 설치하기"에 나열된 절차를 따라 ADT 플러그 인을 새로운 버전으로 다시 설치하면 된다.

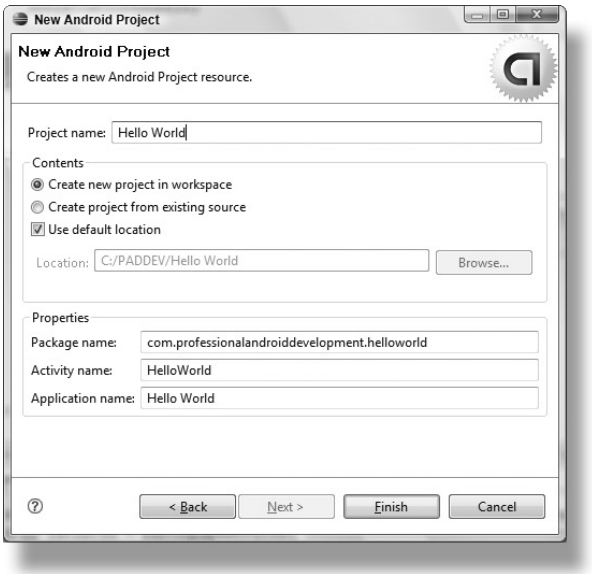

그림 2-4

이렇게 하고 나면 ADT 플러그 인은 Activity를 확장하는 새로운 클래스가 포함된 프로젝트 하나를 생성한다. 이 기본 템플릿은 아무것도 없이 텅 비어있지 않고 "Hello World."를 구현한다. 이 프로젝트를 수정하기에 앞서 이를 가지고 실행 및 디버그 시작 구성을 만들어보자.

시작 구성 만들기

시작 구성launch configurations은 애플리케이션의 실행 및 디버깅을 위한 런타임 옵션을 지정하도록 해준다. 시작 구성을 통해 지정할 수 있는 사항에는 아래와 같은 것들이 있다.

- ✓ 띄울 프로젝트와 액티비티
- ✓ 사용할 에뮬레이터 옵션
- ✓ 입/출력 설정(콘솔 기본값 포함)

여러분은 실행 모드와 디버그 모드에 대해 서로 다른 시작 구성을 지정할 수 있다. 다음 절차는 안드로이드 애플리케이션의 시작 구성 생성 방법을 보여준다.

1. Run → Open Run Dialog... (또는 Run → Open Debug Dialog...)를 선택한다.
2. 타입 목록에서 Android Application을 오른쪽 마우스 버튼으로 클릭하고 New를 선택한다.
3. 이 구성의 이름을 입력한다. 프로젝트마다 복수의 구성을 만들 수 있으므로, 해당 구성이 어떤 설정을 담고 있는지를 쉽게 알 수 있도록 적당한 이름을 붙이자.
4. 이제 시작 옵션을 고른다. 첫 번째 탭(Android)에서는 애플리케이션을 실행(또는 디버그)할 때 시작시킬 프로젝트와 액티비티를 선택할 수 있다. 그림 2-5는 앞서 여러분이 만든 프로젝트를 위한 설정사항을 보여준다.

그림 2-5

5. Target 탭으로 이동해 에뮬레이터를 구성한다. 이 탭에는 화면 크기와 장치 스킨 그리고 네트워크 연결 설정을 고를 수 있는 옵션이 있다. 또한 에뮬레이터상에 있는 사용자 데이터를 임의로 지울 수 있으며, 부트 애니메이션을 활성화하거나 비활성화할 수 있다. 필요하다면 명령행command-line 텍스트박스를 사용해 추가적인 에뮬레이터 시작 옵션을 지정할 수도 있다.
6. 마지막으로, Common 탭에서 기타 추가 속성을 설정한다.
7. Apply를 클릭하면, 여러분의 시작 구성이 저장된다.

안드로이드 애플리케이션 실행 및 디버깅

이렇게 해서 여러분의 첫 번째 프로젝트와 이를 위한 실행 및 디버그 구성을 만들었다. 그럼 Hello World 프로젝트를 실행하고 디버깅하여 설치와 구성을 테스트해보자.

Run 메뉴에서 Run이나 Debug를 선택해 가장 최근에 선택된 구성을 띄우거나, Open Run Dialog... 또는 Open Debug Dialog...를 선택해 사용할 구성을 선택한다.

ADT 플러그 인을 사용하고 있는 경우, 애플리케이션을 실행하거나 디버깅하는 과정은 다음과 같다.

- ✓ 현재 프로젝트를 컴파일한 뒤 이를 안드로이드 실행이미지(.dex)로 변환한다.
- ✓ 실행이미지와 외부 리소스를 안드로이드 패키지(.apk)로 묶는다.
- ✓ 에뮬레이터를 시작시킨다(이미 실행 중인 상태가 아닐 경우).
- ✓ 애플리케이션을 에뮬레이터에 설치한다.
- ✓ 애플리케이션을 시작시킨다.

디버깅 중인 경우에는 브레이크포인트를 설정해 코드를 디버그할 수 있는 이클립스 디버거debugger가 부착된다.

모든 것이 올바르게 동작한다면, 그림 2-6과 같이 에뮬레이터에서 실행 중인 새로운 액티비티 하나를 보게 될 것이다.❺

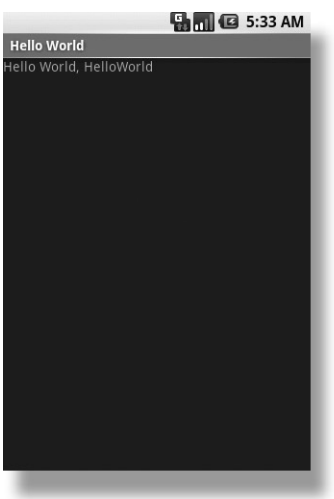

그림 2-6

역주 ❺ 안드로이드 SDK 1.5 r1를 사용하고 있다면, 그림 2-6과 같은 결과 대신 "Failed to find an AVD compatible with target 'Android 1.5'. Launch aborted."라는 에러를 만나게 될 것이다. 이는 실제 장치를 보다 잘 모델링할 수 있도록 하기 위해, 안드로이드 SDK 1.5 r1을 시작으로 안드로이드 가상 장치AVD, Android Virtual Devices라는 개념이 도입됐기 때문이다. 명령행(윈도우의 경우에는 명령 프롬프트를 실행한다)에서 아래의 명령을 사용해 AVD를 하나 만들도록 하자. AVD에 대한 보다 자세한 내용은 http://developer.android.com/guide/developing/tools/avd.html을 참고하자.

android create avd —name TestAVD_1.5 —target 2

헬로 월드 이해하기

잘 동작하는 것을 확인했으니, 한발 뒤로 물러나 여러분의 첫 번째 안드로이드 애플리케이션의 실제 모습을 살펴보자.

Activity는 애플리케이션의 비주얼한 구성요소와 인터랙티브한 구성요소를 위한 기반 클래스로서, 전통적인 데스크톱 개발에서의 폼Form과 얼추 같다고 할 수 있다. 아래에 있는 코드는 Activity에 기반을 둔 클래스의 골격을 보여준다. Activity를 확장하고 있으며, onCreate 메서드를 재정의하고 있음을 눈 여겨 보자.

```
package com.paad.helloworld;

import android.app.Activity;
import android.os.Bundle;

public class HelloWorld extends Activity {
    /** 액티비티가 처음 생성될 때 호출된다. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
    }
}
```

이 템플릿에는 비주얼 인터페이스의 레이아웃이 빠져 있다. 비주얼한 구성요소를 안드로이드에서는 뷰views라 하는데, 이는 전통적인 데스크톱 개발에서의 컨트롤controls과 유사하다.

마법사가 생성한 Hello World 템플릿에는, 아래에 강조된 것처럼 onCreate 메서드가 setContentView를 호출하도록 재정의되어 있다. setContentView는 레이아웃 리소스를 부풀려 유저 인터페이스를 배치한다.

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
}
```

안드로이드 프로젝트의 리소스는 프로젝트 계층의 res라는 폴더 안에 저장되는데, 이 폴더는 drawable, layout, values라는 하위 폴더를 포함하고 있다. ADT 플러그 인은 이들 XML 리소스를 해석하여, R이라는 변수를 통해 설계 시점에 이들을 접근할 수 있도록 하는데, 이에 대해서는 3장에서 설명한다.

아래에 있는 코드는 안드로이드 프로젝트 템플릿이 생성한 main.xml 파일에 정의된 UI 레이아웃을 보여준다.

```
<?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">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, HelloWorld"
    />
</LinearLayout>
```

UI를 XML로 정의한 뒤 이를 부풀리는 방식은 UI 디자인에서 애플리케이션 로직을 깔끔하게 분리해내기 때문에 사용자 인터페이스 구현 시 선호된다.

코드 내에서 UI 엘리먼트에 접근하려면, 해당하는 UI 엘리먼트의 XML 정의에 식별자 identifier 속성을 추가해야 한다. 이렇게 하면 이름이 주어진 각 아이템의 레퍼런스를 리턴하는 findViewById 메서드를 사용할 수 있다. 아래에 있는 XML 코드는 Hello World 템플릿의 TextView 위젯에 추가된 ID 속성을 보여준다.

```
<TextView
    android:id="@+id/myTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Hello World, HelloWorld"
/>
```

그리고 아래에 있는 코드는 이를 코드에서 접근하는 방법을 보여준다.

```
TextView myTextView = (TextView)findViewById(R.id.myTextView);
```

필요하다면 여러분의 레이아웃을 다음와 같이 직접 코드로 생성할 수도 있다(하지만 그리 좋은 방법은 아니다).

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);

    LinearLayout.LayoutParams lp;
    lp = new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
                                       LayoutParams.FILL_PARENT);

    LinearLayout.LayoutParams textViewLP;
    textViewLP = new LinearLayout.LayoutParams(LayoutParams.FILL_PARENT,
                                               LayoutParams.WRAP_CONTENT);

    LinearLayout ll = new LinearLayout(this);
    ll.setOrientation(LinearLayout.VERTICAL);
    TextView myTextView = new TextView(this);
    myTextView.setText("Hello World, HelloWorld");
    ll.addView(myTextView, textViewLP);
    this.addContentView(ll, lp);
}
```

코드에서 사용 가능한 모든 프로퍼티는 XML 레이아웃 내에서 속성으로 설정될 수 있다. 이렇게 하면 레이아웃 디자인과 개별적인 UI 엘리먼트를 보다 손쉽게 바꿀 수 있을 뿐만 아니라, 비주얼 디자인을 애플리케이션과 분리함으로써 코드를 보다 간결하게 유지할 수 있다.

안드로이드 웹 사이트(`http://developer.android.com/guide/index.html`)에는 여러분이 안드로이드 개발자로서 사용하게 될 많은 기능과 좋은 습관을 설명하는 훌륭한 단계별 지침 몇 가지가 포함되어 있다. 따라하기 쉽게 구성되어 있으며, 안드로이드 애플리케이션이 어떻게 하면 서로 잘 맞을 수 있을지에 대한 좋은 아이디어를 담고 있으므로 한번쯤 살펴보길 권한다.

안드로이드 애플리케이션의 종류

여러분이 만드는 안드로이드 애플리케이션의 대부분은 아래에 나열된 범주 중 하나에 속할 것이다.

- ✓ **포그라운드 액티비티** foreground activity 포그라운드에 있을 때만 쓸모 있는 애플리케이션으로서, 화면에 보이지 않을 때는 사실상 일시 중단된다. 게임과 맵 매시업mashups이 이 부류에 속하는 대표적인 예다.

- ✓ **백그라운드 서비스** background service 구성이 변경되고 있는 경우를 제외한 대부분의 시간을 화면에 보이지 않은 채로 실행되는 상호작용이 제한된 애플리케이션. 전화 차단 애플리케이션이나 SMS 자동응답기가 이 부류에 속한다.
- ✓ **인터미턴트 액티비티** intermittent activity 드문드문 상호작용이 있긴 하지만 대부분의 작업을 백그라운드에서 수행한다. 이런 류의 애플리케이션을 실행하면 보통 소리소문 없이 동작하다가 적절한 때에 사용자에게 통지한다. 미디어 플레이어가 대표적인 예다.

복잡한 애플리케이션은 하나의 범주로 분류해 넣기 어려우며, 이 세 가지가 가진 요소를 모두 포함한다. 애플리케이션을 만들 때는 용도를 고려하여 그에 맞게 설계할 필요가 있다. 그럼 위에 설명된 애플리케이션 종류 각각에 대한 몇 가지 설계 고려사항에 대해 자세히 살펴보자.

포그라운드 액티비티

포그라운드 애플리케이션을 만들 때는 액티비티가 포그라운드와 백그라운드 사이를 매끄럽게 넘나들 수 있도록 액티비티 수명 주기(3장에서 설명한다)를 신중히 고려할 필요가 있다.

자신의 수명 주기를 제어할 수 없는 애플리케이션과, 서비스를 제외한 기타 백그라운드로 전환된 애플리케이션은 안드로이드의 리소스 관리를 통해 정리되는 후보 일순위다. 이는 액티비티가 화면에서 사라질 때 애플리케이션 상태를 저장해두고, 포그라운드로 돌아올 때 전과 똑 같은 상태로 나타날 필요가 있음을 의미한다.

깔끔하고 직관적인 사용자 경험user experience을 선보이는 것 역시 포그라운드 액티비티에게는 중요하다.

단정하면서도 매력적인 포그라운드 액티비티를 만드는 방법에 대해서는 3장에서 보다 자세히 배울 것이다.

백그라운드 서비스

이 부류에 속한 애플리케이션은 매우 드문 사용자 입력을 가지고 백그라운드에서 조용히 실행된다. 사용자의 상호작용보다는 대개 하드웨어나 시스템 또는 다른 애플리케이션이 만들어낸 메시지나 액션에 귀 기울인다.

화면에 전혀 보이지 않는 형태의 서비스를 만들 수 있긴 하지만, 실제로는 사용자가 제어할 수 있는 몇 가지 종류의 기능을 제공하는 형태가 더 낫다. 최소한 서비스가 현재 실행 중인지 정도는 사용자가 확인할 수 있어야 하며, 필요에 따라 구성을 바꾼다거나 일시 중지

또는 종료할 수 있게끔 해야 한다.

백그라운드 애플리케이션의 발전소인 서비스에 대해서는 8장에서 자세히 다룬다.

인터미턴트 액티비티

때로는 사용자의 입력에 반응하면서도 활성화된 포그라운드 액티비티가 아닐 때 역시 여전히 쓸모 있는 애플리케이션을 만들고자 할 수 있다. 이런 애플리케이션은 보통 화면에 보이지 않는 백그라운드 서비스와 이를 제어하기 위한 화면에 보이는 컨트롤러 액티비티가 결합한 형태를 띤다.

이 부류에 속한 애플리케이션은 사용자와 상호작용할 때 자신의 상태를 인식할 필요가 있다. 이는 자신이 화면에 나타날 때 액티비티 UI를 업데이트하는 것과, 백그라운드에 있을 때 사용자가 업데이트된 상태를 유지하도록 알림notifications을 전송하는 것을 의미하는 것일 수 있는데, 이에 대해서는 8장의 알림과 서비스 절에서 자세히 다룬다.

모바일 애플리케이션 개발하기

안드로이드가 모바일 소프트웨어 개발을 단순화하는 많은 일을 하고 있지만, 이러한 약정 이면에 있는 갖가지 이유를 이해하는 것은 여전히 중요하다. 모바일 및 임베디드 장치를 위한 소프트웨어를 작성할 때, 특히 안드로이드 애플리케이션을 개발할 때 감안해야 하는 많은 요인이 존재한다.

이번 장에서는 효율적인 안드로이드 코드를 작성하기 위한 몇 가지 기법과 베스트 프랙티스를 배울 것이다. 이후 예제에서는 안드로이드의 새로운 개념이나 기능을 소개할 때 명확함과 간결함을 위해서 이따금씩 효율성을 절충하기도 하는데, 이 예제들은 무언가를 하는 가장 간단한(혹은 가장 이해하기 쉬운) 방법을 보여주도록 의도된 것이므로, 이를 통해 제시되는 방법이 항상 최선은 아님을 염두에 둬야 한다. "내가 '바담 풍' 해도 너는 '바람 풍' 해라"라는 옛 속담을 기억하기 바란다.

하드웨어를 고려한 설계

작고 휴대 가능한 모바일 장치는 흥미진진한 소프트웨어 개발 기회를 제공한다. 이들이 가진 제한된 화면 크기를 비롯한 적은 메모리와 저장공간 그리고 프로세서 속도는 흥미가 훨씬 덜하지만, 대신 몇 가지 독특한 도전과제를 선사한다.

데스크톱이나 노트북 컴퓨터와 비교해볼 때 모바일 장치는 다음과 같은 특징을 갖는다.

- ✓ 느린 처리 속도
- ✓ 제한된 RAM
- ✓ 제한된 영구 저장공간
- ✓ 저해상도의 작은 화면
- ✓ 비싼 데이터 전송 비용
- ✓ 다소 신뢰성이 떨어지는 데이터 연결
- ✓ 제한된 배터리 수명

새로운 애플리케이션을 만들 때는 이러한 제한사항을 마음에 새기는 것이 중요하다.

효율적일 것

임베디드 장치, 특히 모바일 장치 제조사는 프로세서 속도의 잠재적인 개선보다는 작은 크기와 긴 배터리 수명에 더 가치를 둔다. 개발자의 입장에서 볼 때, 이는 곧 무어의 법칙으로 인해 주어지던 유리한 출발점을 잃는다는 것을 의미한다. 해마다 보게 되는 데스크톱과 서버 하드웨어의 성능 향상은 모바일의 경우 대개 프로세서의 큰 속도 향상보다는 좋은 전력효율로 재 표현된다.

실제로 이는 여러분이 항상 코드를 빠르고 반응성이 좋게끔 최적화해야 함과 동시에, 여러분의 소프트웨어가 사용되는 동안 하드웨어 향상의 이득을 볼 수 없다고 가정할 필요가 있음을 의미한다.

코드의 효율성에 관한 부분은 소프트웨어 공학에서도 굉장히 큰 주제에 속하므로, 섣불리 여기서 다루려 하지는 않겠다. 비록 이번 장에서는 안드로이드에 특화된 몇 가지 효율성에 관한 팁만을 다루지만, 효율성이라는 것은 모바일 장치처럼 리소스가 한정된 환경에서 특히 중요하다는 사실을 기억하자.

제한된 저장 공간을 예측할 것

플래시 메모리와 반도체 디스크SSD, Solid-State Disk ❻의 발전은 모바일 장치가 가진 저장 능

역주 ❻ 반도체 기억 소자를 사용한 기억 장치. 플래시 메모리로만 만들어져 고속 데이터 입출력이 가능하고, 기존의 하드 디스크 드라이브HDD, Hard Disk Drive와 같은 기계적인 동작이 없어 발열, 소음, 전력 소모 등이 적으며, 외부 충격에 의한 데이터 손실이 없고 소형 경량화가 쉬운 장점이 있어, 기존의 HDD를 대체할 차세대 대용량 기억 장치로 주목받고 있다.

력의 극적인 증대를 이끌어왔다(늘어난 저장 공간은 금새 다시 MP3로 가득 채워지는 경향이 있긴 하지만). 하지만 실제로 대부분의 장치가 여전히 여러분의 애플리케이션에 상대적으로 제한된 저장공간을 제공한다. 애플리케이션의 컴파일된 크기도 중요하지만, 애플리케이션이 시스템 리소스를 품위 있게 사용하도록 만드는 것이 보다 중요하다.

여러분은 애플리케이션이 데이터를 저장하는 방법에 대해 신중히 고려해야 한다. 안드로이드 데이터베이스와 콘텐트 공급자를 사용하면 대량의 데이터를 편리하게 저장, 재사용, 공유할 수 있는데, 이에 대해서는 6장에서 설명한다. 환경설정이나 상태 설정과 같은 작은 데이터 저장을 위해서도 안드로이드는 최적화된 프레임워크를 제공하는데, 이 역시 6장에서 살펴본다.

물론 이들 메커니즘을 사용하지 않고 직접 파일시스템에 기록하고자 하거나 그럴 필요가 있는 경우에는 그렇게 할 수 있는데, 이런 상황에서는 항상 파일을 구조화하는 방법에 대해 고려해야 하며, 여러분이 선택한 방법이 과연 효율적인 해결책인지를 확실히 해야 한다.

리소스를 품위 있게 사용하는 방법 가운데 하나는, 사용하고 난 리소스를 여러분이 직접 해제하는 것이다. 캐싱caching 같은 기법은 반복되는 네트워크 조회를 제한하는 데 유용하지만, 더 이상 필요치 않을 때는 파일시스템상에 파일이나 데이터베이스에 레코드를 남겨두지 않는다.

작은 화면을 위해 디자인할 것

휴대폰의 작은 크기와 휴대할 수 있다는 특징은 좋은 인터페이스를 만드는 데 걸림돌이 되는데, 이는 사용자가 점점 더 멋지고 정보가 풍부한 그래픽 사용자 경험을 요구할 때 특히 더 그렇다.

사용자는 (작은) 화면을 종종 흘끗 보기만 한다는 사실을 염두에 두고 애플리케이션을 작성하자. 컨트롤 개수를 줄이고 가장 중요한 정보를 맨 앞 한가운데 둠으로써 애플리케이션을 직관적이고 사용하기 쉽게 만들자.

여러분이 4장에서 만들게 될 것과 같은 그래픽 컨트롤은 빽빽한 정보를 이해하기 쉽게 전하는 훌륭한 수단이다. 버튼과 텍스트 입력 박스로 가득 찬 화면 대신, 색상과 모양 그리고 그래픽스를 사용해 정보를 나타내자.

만일 터치 스크린을 지원할 계획이라면(생각해보지 않았다면 지금 계획에 넣도록 하자), 터치 입력이 여러분의 인터페이스 디자인에 어떠한 영향을 미치게 될지 고려할 필요가 있을 것이다. 스타일러스stylus[7]의 시대는 지났다. 이제는 모든 것을 손가락으로 하는 것이 대

역주 ❼ 펜 모양으로 된 위치지정 입력 도구.

세이므로, 화면을 손가락으로 조작할 수 있게끔 뷰를 충분히 크게 만들자. 터치 스크린 상호작용에 대한 보다 많은 정보는 11장에서 다룬다.

물론, 휴대폰의 해상도와 화면 크기가 점점 커지는 추세이므로, 작은 화면을 위해 디자인하되 UI의 크기가 조절될 수 있도록 하는 것이 현명하다.

느린 속도와 긴 지연 시간을 예측할 것

5장에서는 애플리케이션에서 인터넷 리소스를 사용하는 방법을 배울 것이다. 풍부한 온라인 정보를 애플리케이션에 통합하는 능력은 믿을 수 없으리만큼 강력하다.

불행히도 모바일 웹은 우리가 바라는 만큼 빠르거나 신뢰적이지 않으며 쉽게 사용 가능하지도 않으므로, 인터넷기반 애플리케이션을 개발할 때는 네트워크 연결이 느리고, 간헐적이며, 요금 또한 비싸다고 가정하는 편이 최선이다. 이러한 상황은 3G 무제한 정액 요금제 및 사방에 설치된 Wi-Fi와 함께 변해가고 있지만, 최악의 경우를 대비해 설계하게 되면 언제나 높은 기준의 사용자 경험을 전해줄 수가 있다.

이는 또한 여러분의 애플리케이션이 끊긴(혹은 아직 찾지 못한) 데이터 연결을 다룰 수 있어야 함을 의미한다.

안드로이드 에뮬레이터는 이클립스 시작 구성 설정 시 네트워크 연결 속도와 지연 시간을 제어할 수 있도록 해준다. 그림 2-7은 차선次善으로 사용될 EDGE 연결을 시뮬레이션하기 위한 에뮬레이터의 네트워크 연결 속도와 지연 시간 설정을 보여준다.

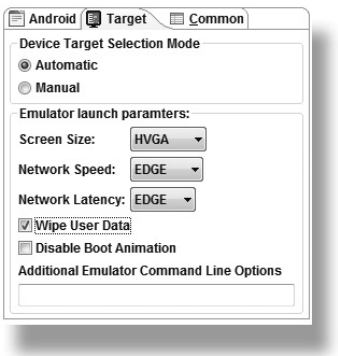

그림 2-7

사용 가능한 네트워크는 무엇이고 속도와 지연 시간은 어떻게 되는지에 관계 없이 좋은 반응성을 유지하도록 테스트하자. 어떤 상황에서는 사용할 수 있는 네트워크에 따라 애플리케이션 기능을 제한하거나 캐시된 데이터에 대한 네트워크 조회를 줄이는 편이 더 낫다

는 것을 발견할지도 모른다. 실행 중에 사용 가능한 네트워크 연결의 종류와 속도를 감지하는 방법에 대해서는 10장에서 다룬다.

비용은 얼마나?

모바일 장치를 하나쯤 가지고 있다면 너무나 잘 알겠지만, 모바일 장치가 가진 강력한 기능 중 일부는 사실상 비용을 지불해야 사용할 수 있다. SMS, GPS, 그리고 데이터 전송 같은 서비스를 사용하려면 서비스 공급자에게 추가 요금을 내야 한다.

애플리케이션에 있는 기능과 연관된 모든 비용을 최소화하는 것과, 사용자가 자신이 수행하는 동작에 비용이 부과될 수 있음을 아는 것이 왜 중요한지는 명백하다.

바깥 세상과의 상호작용을 수반하는 모든 동작에는 그에 따른 비용이 존재한다고 가정하는 것이 좋다. 아래의 방법을 통해 상호작용 비용을 최소화하자.

- ✓ 가능한 한 데이터 전송을 작게 하기
- ✓ 데이터와 GPS 결과를 캐싱해 중복 또는 반복 참조 제거하기
- ✓ 액티비티가 포그라운드에서 화면에 보이지 않을 때, 데이터 전송 및 GPS 업데이트가 UI를 업데이트하는 데에만 사용되고 있다면 이를 모두 중단하기
- ✓ 데이터 전송(그리고 위치 조회)을 위한 재생/갱신 비율을 실질적인 최저수준으로 유지하기
- ✓ 큰 업데이트나 전송은 8장에 소개된 것처럼 알람을 이용해 "한산할 때" 수행하도록 계획하기

가끔은 비용이 적게 드는 낮은 품질 옵션을 사용하는 것이 최선의 해결책이기도 하다.

7장에 설명된 위치기반 서비스를 사용할 때는 관련된 비용이 존재하는지에 따라 위치 공급자location provider를 선택할 수 있다. 위치기반 애플리케이션을 만들 때는 사용자가 저렴한 요금과 높은 정확도 사이를 선택할 수 있도록 배려하자.

어떤 상황에서는 비용을 정의하기가 어렵거나 사용자마다 상이하기도 하다. 서비스 요금은 서비스 공급자와 사용자 계획에 따라 천차만별이다. 무제한 무료 데이터 전송을 사용하는 사람들이 있는가 하면, 무료 SMS를 사용하는 사람들도 있다.

저렴해 보이는 특정 서비스 하나를 강요하기보다는 이를 사용자가 직접 고를 수 있도록 하자. 예컨대, 인터넷을 통해 데이터를 다운로드받을 경우, 어떤 네트워크든 사용 가능한 것 하나를 골라 사용할 것인지 아니면 Wi-Fi를 통해 연결됐을 때만 전송하도록 제한할 것인지를 사용자에게 물을 수 있다.

사용자 환경 고려하기

사용자가 자신의 폰에 있는 기능 중 여러분의 애플리케이션을 가장 중요한 것으로 여기리라 생각하고 있다면 큰 오산이다.

일반적으로 휴대폰의 첫째가는 주요한 기능은 전화이고, 둘째는 SMS와 이메일 커뮤니케이터이며, 셋째는 카메라이고, 넷째는 MP3 플레이어다. 여러분이 작성하는 애플리케이션은 필시 "유용한 모바일 도구"라는 다섯 번째 부류에 속할 것이다.

그렇다고 이게 나쁘다는 말은 아니다. 구글 맵과 웹 브라우저 등 훌륭한 다른 애플리케이션 역시 사정은 마찬가지다. 이런 점을 감안한다면 사용자의 휴대폰 사용 모델은 달라지게 될 것이다. 어떤 사람들은 자신의 휴대폰을 절대 음악 감상용으로 사용하지 않으며, 또 일부 휴대폰은 카메라를 가지고 있지 않지만, 장치에 없어서는 안 될 만큼 흔히 내제되어 있는 멀티태스킹multitasking 원리는 사용성 설계를 위한 중요한 고려사항이다.

사용자가 여러분의 애플리케이션을 언제 그리고 어떻게 사용할지 고려하는 것 또한 중요하다. 사람들은 기차를 타거나 길을 걸을 때 심지어 운전 중에도 항상 휴대폰을 사용한다. 여러분은 사람들이 자신의 폰을 적절히 사용하도록 만들 수는 없지만, 여러분의 애플리케이션을 필요 이상으로 혼란스럽지 않게 만들 수는 있다.

소프트웨어 설계의 입장에서 볼 때 이는 무엇을 의미할까? 애플리케이션을 만들 때는 아래와 같은 내용을 염두에 두자.

- ✓ **상황에 따라 효율적으로 동작해야 한다.** 액티비티가 포그라운드에 있지 않을 때는 일시 중단된 상태로 시작하도록 하자. 안드로이드는 액티비티가 일시 중단되거나 다시 복구되어 실행을 계속할 경우 이벤트 핸들러를 호출하므로, 애플리케이션이 화면에 보이지 않을 때 여러분은 UI 업데이트와 네트워크 조회를 잠시 멈출 수 있다(애플리케이션이 화면에 보이지 않을 때 UI를 업데이트하는 것은 별 의미가 없다). 백그라운드에서 업데이트나 처리를 계속해야 할 필요가 있는 경우를 위해, 안드로이드는 UI 오버헤드 없이 백그라운드에서 실행되도록 설계된 서비스 클래스를 제공한다.
- ✓ **백그라운드에서 포그라운드로 매끄럽게 전환되어야 한다.** 모바일 장치의 멀티태스킹 특성으로 인해, 여러분의 애플리케이션은 십중팔구 규칙적으로 백그라운드로 전환되었다가 빠져나올 가능성이 높다. 이렇게 될 경우에는 빠르고 매끄럽게 "되살아나는 것"이 중요하다. 안드로이드의 비결정적 프로세스 관리nondeterministic process management는 여러분의 애플리케이션이 백그라운드에 있을 경우 리소스 회수를 위해 종료될 수 있음을 의미한다. 이것이 사용자의 눈에 보여서는 안 된다. 이는 사용자가 여러분의 애플리케이션을 재시작하는 것과 다시 복구되어 실행을 계속하는 것 간의 차이를 알아

차리지 못하도록 애플리케이션 상태를 저장하고 업데이트된 내용을 대기열에 넣음으로써 보장할 수 있다. 원상태로 되돌아오는 전환은 사용자가 마지막으로 봤던 동일한 UI와 애플리케이션 상태를 보이도록 매끄러워야 한다.

- ✓ **품위 있어야 한다.** 여러분의 애플리케이션은 절대로 사용자가 현재 사용하고 있는 액티비티의 포커스를 빼앗거나 방해해서는 안 된다. 애플리케이션이 포그라운드에 있지 않을 때는, 뭔가를 알리거나 상기시키는 방법으로 사용자의 주의를 끌기보다는 알림Notifications과 토스트Toasts(8장에서 자세히 다룬다)를 사용하자. 모바일 장치가 사용자에게 뭔가를 알리는 방법에는 여러 가지가 있다. 예컨대, 전화가 걸려올 경우 폰은 벨소리를 울리고, 읽지 않은 메시지가 있을 경우에는 LED를 깜박이며, 신규 음성메일이 있을 때는 상태 바에 작은 "메일" 아이콘을 나타낸다. 알림 메커니즘을 사용하면 위에 나열한 모든 기법뿐만 아니라 그 이상의 많은 것이 가능하다.
- ✓ **일관된 사용자 인터페이스를 보여줘야 한다.** 여러분의 애플리케이션은 사용자가 늘 사용하는 여러 애플리케이션 가운데 하나일 가능성이 있으므로, 편리한 UI를 선보이는 것이 중요하다. 사용자가 애플리케이션을 띄울 때마다 매번 이를 이해하고 다시 배우도록 강요하지 말자. 애플리케이션은 사용이 간단하고, 쉽고, 분명해야 하는데, 제한된 화면 공간과 산만한 사용자 환경 속에서는 특히 더 그렇다.
- ✓ **반응성이 좋아야 한다.** 반응성은 모바일 장치에서 가장 중요한 설계 고려사항 가운데 하나다. 분명 여러분은 소프트웨어의 일부가 "얼어버려" 꼼짝달싹 못하는 좌절을 경험해봤을 것이다. 모바일이 가지는 다기능이라는 특징은 더 성가시다. 느리고 신뢰성이 떨어지는 데이터 연결로 인해 생길 수 있는 지연 가능성을 염두에 두고, 여러분의 액티비티가 좋은 반응성을 유지하도록 애플리케이션이 작업자 스레드worker threads와 백그라운드 서비스를 사용하는 게 중요하며, 다른 애플리케이션이 적시에 반응할 수 없도록 방해하는 것들을 중단하는 것이 보다 중요하다.

안드로이드 애플리케이션 개발하기

지금까지는 안드로이드에 대해 구체적으로 다루지 않았다. 앞서 살펴본 설계 고려사항은 모바일 애플리케이션 개발 시 보편적으로 적용되는 중요한 내용이다. 안드로이드는 이러한 일반적인 지침과 더불어 추가적인 몇 가지 특별한 고려사항을 가지고 있다.

우선 잠시 시간을 내어 `http://code.google.com/android/toolbox/philosophy.html`에 있는 구글의 안드로이드 설계 철학을 읽어보면 많은 도움이 될 것이다.

안드로이드 설계 철학은 애플리케이션이 다음의 사항을 만족할 것을 요구한다.

- ✓ 빠를 것
- ✓ 반응성이 좋을 것
- ✓ 안전할 것
- ✓ 심리스seamless할 것

빠르고 효율적일 것

리소스가 한정된 환경에서 빠르다는 것은 효율적임을 뜻한다. 여러분이 이미 알고 있는 효율적인 코드 작성에 관한 많은 것들이 안드로이드에도 유효하겠지만, 임베디드 시스템의 한계와 Dalvik VM의 사용은 이를 그저 당연한 것으로 여길 수만은 없음을 의미한다.

가장 현명한 조언은 소스를 들여다보라는 것이다. 안드로이드 팀은 안드로이드를 위한 효율적인 코드 작성에 관한 몇 가지 구체적인 지침을 발표했다. 이 제안을 여기에 다시 고쳐 옮겨 반복하기보다는, 여러분이 직접 `http://code.google.com/android/toolbox/performance.html`을 방문해 살펴볼 것을 권한다.

이들 성능 제안 가운데, 가령 내부 세터setters 및 게터getters 사용을 피하라든가, 인터페이스보다는 가상virtual을 선호하라는 등 몇 가지 사항이 기존에 알고 있던 설계 원칙과 모순됨을 발견할지 모른다. 임베디드 장치와 같이 리소스가 한정된 시스템을 위한 소프트웨어를 작성할 때는 이따금씩 정형화된 설계 원칙과 높은 효율성 요구 간에 절충이 존재하기 마련이다.

효율적인 안드로이드 코드 작성을 위한 핵심 가운데 하나는, 데스크톱과 서버 환경에서의 가정들을 임베디드 장치로 끌어들이지 않는 것이다.

대부분의 데스크톱 및 서버 장비의 경우 2GB에서 4GB의 메모리를 갖는 것이 보통임에 반해, 최신 스마트폰이라 할지라도 32MB RAM을 갖는 것은 행운이다. 여러분은 이처럼 한정된 메모리를 효율적으로 사용하기 위해 특별 관리가 필요하다. 이는 스택과 힙의 사용 방식, 객체 생성 제한하기, 그리고 변수의 유효범위가 메모리 사용에 어떠한 영향을 미치는지에 관해 생각하는 것을 뜻한다.

반응성이 좋을 것

안드로이드는 반응성을 매우 심각하게 받아들인다.

안드로이드는 액티비티 관리자와 윈도우 관리자를 통해 반응성을 관리한다. 만일 이 두 서비스 중 하나가 반응이 느린 애플리케이션을 감지하게 되면, 그 서비스는 그림 2-8과 같이 애플리케이션의 반응이 느림을 알리는 AUR(Application unresponsive) 메시지를 내보일 것이다.

그림 2-8

이 경보는 모달modal로서, 포커스를 빼앗으며, 사용자가 버튼을 누르거나 여러분의 애플리케이션이 반응을 보이기 시작할 때까지 사라지지 않을 것이다. 이는 절대로 사용자에게 마주하도록 하고 싶지 않은 최후의 것이나 다름없다.

안드로이드는 반응성을 결정하기 위해 다음 두 가지 조건을 감시한다.

- ✓ 애플리케이션은 키 누름이나 화면 터치 같은 모든 사용자 액션에 대해 반드시 5초 이내로 반응해야 한다.
- ✓ 브로드캐스트 수신자Broadcast Receiver는 자신의 `onReceive` 핸들러에서 반드시 10초 이내에 리턴해야 한다.

느린 반응성을 유발하는 가장 유력한 범인은 네트워크 조회, 복잡한 처리(게임의 이동 계산 같은), 그리고 파일 I/O다.

이들 동작이 위 반응성 조건을 초과하지 않도록 보장하는 방법에는 여러 가지가 있는데, 이 가운데 서비스와 작업자 스레드를 사용하는 방법은 8장에서 다룬다.

사용성의 관점에서 볼 때 AUR 다이얼로그는 최후의 수단이다. 넉넉해 보이는 이 5초 제한은 최악의 시나리오이지 목표로 삼아야 할 기준이 아니다. 사용자는 키 누름과 동작 간에 0.5초 이상 넘어가는 모든 버벅거림을 알아차릴 것이다. 다행히도, 여러분이 이미 작성하고 있는 효율적인 코드의 부수 효과는 더 빠르고 반응성 좋은 애플리케이션일 것이다.

안전한 애플리케이션 개발하기

안드로이드 애플리케이션은 하드웨어를 직접 접근하며, 자유롭게 배포될 수 있고, 또 열린 소통이 오고 가는 오픈 소스 플랫폼 위에 빌드되므로, 보안이 큰 관심사인 것은 어찌 보면 당연한 일이다.

사용자가 설치하는 애플리케이션과 여기에 부여하는 권한에 대한 책임은 대부분의 경우 사용자 자신이 질 것이다. 안드로이드 보안 모델은 애플리케이션이 어떠한 서비스와 기능을 사용하기 전에 그에 대한 권한을 먼저 요청하도록 함으로써 해당 서비스와 기능에 대한 접근을 제한한다. 그러면 설치하는 동안 사용자는 애플리케이션이 요청된 권한을 부여받아야 하는지를 결정한다. 안드로이드의 보안 모델에 관해서는 11장과 `http://code.google.com/android/devel/security.html`에서 보다 많이 배울 수 있다.

이것만으로는 부족하다. 여러분은 안전 그 자체에 관심을 가지고 애플리케이션을 만들어야 할 뿐만 아니라, 애플리케이션이 장치를 위협하기 위한 목적으로 약탈될 수 없도록 보장해야 할 필요가 있다. 장치 보안 유지를 위해 사용할 수 있는 기법에는 여러 가지가 있는데, 이들에 대해서는 그 안에 수반된 기술들을 배워나가면서 보다 자세히 다룰 것이다. 여러분은 특히 아래의 내용을 염두에 둬야 한다.

- ✓ 여러분이 생성하는 모든 서비스나 전송하는 모든 브로드캐스트에 대해 권한 요청을 고려하자.
- ✓ 여러분의 애플리케이션이 인터넷이나 SMS 메시지 또는 인스턴트 메시징(IM) 같은 외부 소스로부터 입력을 받아들일 때는 각별히 주의하자. 애플리케이션 메시징을 위한 IM과 SMS 사용에 대한 보다 자세한 사항은 9장에서 찾아볼 수 있다.
- ✓ 여러분의 애플리케이션이 하위 수준의 하드웨어에 대한 접근을 노출할지 모를 때는 신중을 기하자.

이 책에 있는 많은 예제는 명확함과 간결함을 이유로 보안에 대해 상당히 완화된 접근법을 취한다. 여러분만의 애플리케이션을 만드는 경우, 특히 배포 계획을 가지고 있는 경우에는 이 부분을 간과해서는 안 된다. 안드로이드 보안에 대한 자세한 사항은 11장에서 찾아볼 수 있다.

심리스 사용자 경험 보장하기

다소 애매모호하긴 하지만 심리스 사용자 경험에 대한 아이디어는 중요한 개념이다. 심리스seamless란 무슨 뜻일까? 심리스의 목표는 애플리케이션이 즉각 시작, 종료, 전이되고, 눈에 띄는 지연이나 거슬리는 전이가 없는 일관된 사용자 경험이다.

모바일 장치의 속도와 반응성이 사용자 경험을 저하해서는 안 된다. 안드로이드의 프로세스 관리는 말없는 자객刺客처럼 필요에 따라 리소스 회수를 위해 백그라운드 애플리케이션을 종료시킨다. 여러분의 애플리케이션은 이점을 기억해, 재시작되든 아니면 다시 복구되어 실행을 계속하든 관계 없이 항상 일관된 인터페이스를 보여줘야 한다.

일반적으로 서로 다른 개발자에 의해 작성된 여러 서드파티 애플리케이션을 실행하는 안

드로이드 장치에서는 이들 애플리케이션 간의 매끄러운 상호작용이 특히나 중요하다.

사용성에 대해서는 일관적이면서도 직관적인 접근법을 사용하자. 물론 기존과는 다른 파격적인 애플리케이션을 만들 수도 있지만, 이 역시 폭넓은 안드로이드 환경과 깔끔하게 통합돼야 한다.

애플리케이션이 화면에 보이지 않을 때는 세션 간에 데이터를 지속하고, 프로세서 사이클이나 네트워크 대역폭 혹은 배터리 수명을 닳게 하는 작업을 일시 중단하자. 액티비티가 화면에 보이지 않는 동안에도 애플리케이션이 계속 실행되어야 할 필요가 있는 처리를 가지고 있다면 서비스를 이용하되 이러한 구현 결정을 사용자에게 숨기자.

애플리케이션이 뒤에 있다가 앞으로 오거나 재시작될 때는 가장 마지막에 보였던 상태로 매끄럽게 되돌아와야 한다. 사용자의 입장에서 각 애플리케이션은 사용될 채비를 갖추고 화면 밖에서 얌전히 앉아있어야 한다.

또한 여러분은 알림 사용을 위한 베스트 프랙티스 가이드라인을 따라야 하며, 애플리케이션 간의 일관성 유지를 위해 일반화된 UI 요소와 테마를 사용해야 한다.

이 외에도 심리스 사용자 경험을 보장하기 위해 사용할 수 있는 다른 많은 기법들이 존재하는데, 안드로이드에서 할 수 있는 보다 많은 가능성을 발견해 나가면서 그 가운데 일부를 이어지는 장들을 통해 만나보게 될 것이다.

할 일 목록 예제

이번 예제에서는 안드로이드 애플리케이션을 완전히 밑바닥에서부터 새로 만들어볼 것이다. 이 예제는 새로운 프로젝트를 시작하는 데 수반되는 기본적인 단계들을 설명하기 위한 것으로서, 안드로이드가 제공하는 뷰 컨트롤을 사용해 간단한 할 일 목록to-do list 애플리케이션을 만든다.

이 예제에서 일어나는 일을 다 이해하지 못한다 할지라도 너무 걱정하지 말자. ArrayAdapter, ListView, KeyListener 등 이 애플리케이션을 만드는 데 사용된 기능 가운데 일부는 이후의 장들을 통해 자세히 설명될 때까지 제대로 알려주지 않을 것이다. 또한 여러분이 안드로이드에 대해 더 많은 것을 배워나감에 따라 이 예제로 되돌아와서 새로운 기능을 추가할 것이다.

1. 먼저 새로운 안드로이드 프로젝트 하나를 생성한다. 이클립스에서 File → New → Project...를 선택한 다음, 그림 2-9에서 보이는 것처럼 Android를 선택하고 Next를 클릭한다.

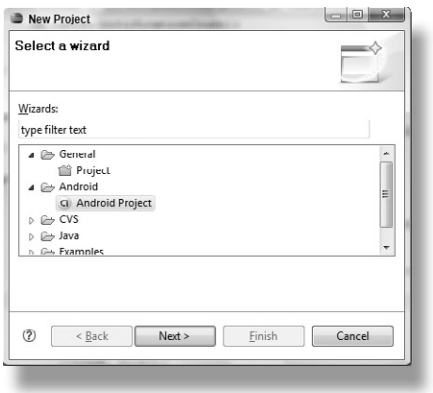

그림 2-9

2. 화면에 나타나는 다이얼로그 박스(그림 2-10)에 새로운 프로젝트에 관한 세부사항을 입력한다. "Application name"에는 맘에 드는 여러분의 애플리케이션 이름을 입력하고, "Activity name"에는 여러분의 액티비티 하위클래스 이름을 입력하자. Finish를 클릭해 입력된 세부정보를 바탕으로 새로운 프로젝트를 생성하자.

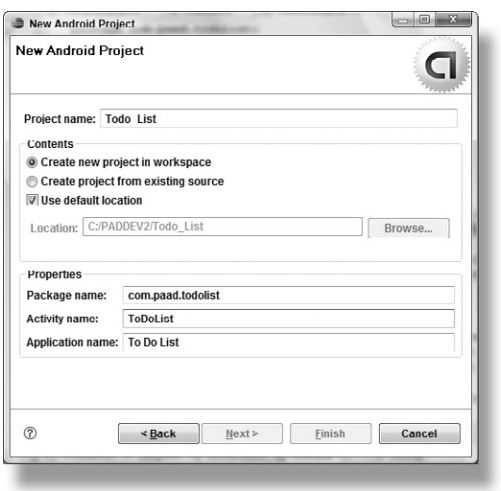

그림 2-10

3. Run → Open Debug Dialog...와 Run → Open Run Dialog...를 통해 디버그와 실행 각각을 위한 새로운 구성을 만들고, Todo_List 프로젝트를 지정함으로써 디버그 및 실행 구성을 설정하자. 시작 동작 Launch Default Activity로 놔두거나, 그림 2-11에서 보이는 것처럼 새로운 ToDoList 액티비티를 띄우도록 명시적으로 설정할 수 있다.

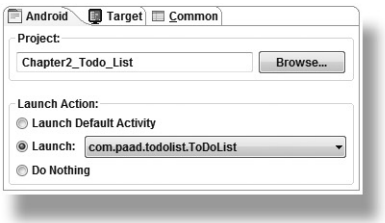

그림 2-11

4. 이제 여러분이 사용자에게 보여주고 싶은 것과 사용자가 수행할 필요가 있을 동작들을 결정한다. 이를 위한 사용자 인터페이스는 가능한 한 직관적으로 설계하자.

이 예제에서는 사용자에게 해야 할 일들의 목록과 새로운 할 일을 추가하기 위한 텍스트 입력 박스 하나를 내보이고자 한다. 리스트와 텍스트 입력 컨트롤(뷰)은 모두 안드로이드 라이브러리를 통해 사용 가능하다. 안드로이드에서 사용할 수 있는 뷰와 새로운 뷰를 생성하는 방법에 대한 보다 자세한 내용은 4장에서 배울 것이다.

UI를 배치하는 데에는 레이아웃 리소스 파일을 사용하는 방법이 선호된다. 그림 2-12와 같이 프로젝트의 res/layout 폴더에 있는 main.xml 레이아웃 파일을 열자.

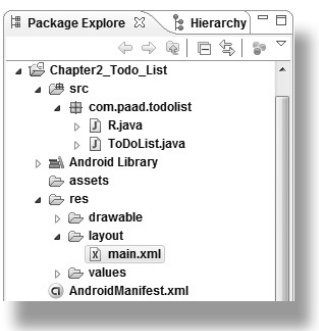

그림 2-12

5. 이 메인 레이아웃을 수정하여 LinearLayout에 ListView와 EditText를 포함시킨다. 여러분이 코드에서 이 EditText와 ListView의 레퍼런스를 얻을 수 있도록, 이 둘 모

두에 컨트롤 ID를 주는 것이 중요하다.

```
<?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">
    <EditText
        android:id="@+id/myEditText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="새 해야 할 일"
    />
    <ListView
        android:id="@+id/myListView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

6. 정의된 사용자 인터페이스와 함께, 여러분의 프로젝트에 있는 소스 폴더에서 ToDoList.java 액티비티 클래스를 연다. 이번 예제에서는 onCreate 메서드를 재정의하는 것으로 모든 변경이 끝난다. 먼저 setContentView로 여러분의 UI를 부풀린 다음, findViewById를 사용해 ListView와 EditText의 레퍼런스를 얻어온다.

```
public void onCreate(Bundle icicle) {
      super.onCreate(icicle);

      // 여러분의 뷰를 부풀린다.
      setContentView(R.layout.main);

      // UI 위젯의 레퍼런스를 얻어온다.
      ListView myListView = (ListView)findViewById(R.id.myListView);
      final EditText myEditText = (EditText)findViewById(R.id.myEditText);
}
```

7. 계속해서 각각의 해야 할 일들을 저장하기 위한 문자열 ArrayList 하나를 정의한다. ArrayAdapter를 사용하면 ListView를 ArrayList와 묶을 수 있으므로, 새로운 ArrayAdapter 인스턴스 하나를 생성해 해야 할 일들을 담기 위한 배열을 ListView와 묶는다. ArrayAdapter는 5장에서 다시 살펴보기로 하자.

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    ListView myListView = (ListView)findViewById(R.id.myListView);
    final EditText myEditText = (EditText)findViewById(R.id.myEditText);

    // 해야 할 일들을 담기 위한 배열 리스트(array list)를 생성한다.
    final ArrayList<String> todoItems = new ArrayList<String>();
    // 위 배열을 리스트 뷰와 묶기 위한 배열 어댑터(array adapter)를 생성한다/
    final ArrayAdapter<String> aa;
    aa = new ArrayAdapter<String>(this,
                                  android.R.layout.simple_list_item_1,
                                  todoItems);
    // 위 배열 어댑터를 리스트 뷰와 묶는다.
    myListView.setAdapter(aa);
}
```

8. 이 할 일 목록 애플리케이션이 기능하게끔 만들기 위한 마지막 단계는, 사용자로 하여금 새로운 해야 할 일을 추가할 수 있도록 하는 것이다. "D패드 가운데 버튼"의 클릭을 기다리는 `onKeyListener`를 `EditText`에 추가한 다음, `EditText`의 내용을 할 일 목록 배열에 추가하고 이 변경을 `ArrayAdapter`에 통지한다. 그런 뒤 `EditText`를 비워 또 다른 해야 할 일이 입력될 수 있도록 준비한다.

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    ListView myListView = (ListView)findViewById(R.id.myListView);
    final EditText myEditText = (EditText)findViewById(R.id.myEditText);

    final ArrayList<String> todoItems = new ArrayList<String>();
    final ArrayAdapter<String> aa;
    aa = new ArrayAdapter<String>(this,
                                  android.R.layout.simple_list_item_1,
                                  todoItems);
    myListView.setAdapter(aa);

    myEditText.setOnKeyListener(new OnKeyListener() {
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            if (event.getAction() == KeyEvent.ACTION_DOWN)
```

```
                if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
                {
                    todoItems.add(0, myEditText.getText().toString());
                    aa.notifyDataSetChanged();
                    myEditText.setText("");
                    return true;
                }
            return false;
        }
    });
}
```

9. 이 애플리케이션을 실행 또는 디버그하면, 그림 2-13과 같이 할 일 목록 위에 텍스트 입력 박스 하나가 보일 것이다.[8]

10. 이로써 여러분은 첫 번째 "진짜" 안드로이드 애플리케이션을 완성했다. 시험 삼아 코드에 브레이크포인트를 걸어 디버거를 테스트해보고 DDMS 퍼스펙티브를 살펴보자.

현재로서는 이 할 일 목록 애플리케이션이 아주 유용하다고는 할 수 없다. 사용자가 입력한 해야 할 일들을 세션 간에 저장하고 있지 않으며, 목록에 있는 항목을 편집하거나 삭제할 수 없고, 만기일과 작업 우선순위같이 작업 목록에 필요한 전형적인 항목들이 기록되거나 출력되지 않는다. 모든 걸 감안해볼 때, 지금까지 살펴본 훌륭한 모바일 애플리케이션 설계를 위한 대부분의 기준을 만족하지 못한다.

역주 8 그림 2-13과 같은 결과 화면은커녕 컴파일조차 되지 않아 막막해하고 있다면 이유를 곰곰이 한번 생각해보자(힌트! 자바 소스 파일 맨 앞부분에 보통 나열하는 그것!). 그렇다. 위에서 재정의한 onCreate 메서드는 기존에 없던 ListView, EditText, ArrayList, ArrayAdapter 등의 새로운 클래스들을 사용하고 있으므로, 소스 파일 앞에 이들에 대한 import 문을 추가해야 한다. ToDoList.java 파일을 열어 아래의 import 문을 추가하자. 혹, 왜 그래야 하는지 이유를 알 수 없어 고개를 갸우뚱하는 독자들이 있다면, 그림 2-13의 할 일 목록 맨 앞에 "자바 배우기"를 먼저 추가해야 할 것이다.

```
import java.util.ArrayList;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnKeyListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
```

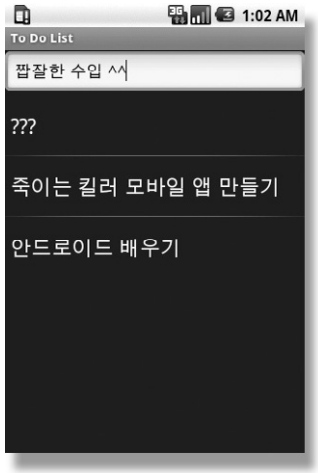

그림 2-13

이러한 부족한 부분들은 이후의 장들을 통해 수정해 나갈 것이다.

안드로이드 개발 도구

안드로이드 SDK에는 여러분의 프로젝트를 생성, 테스트, 디버그하기 위한 여러 가지 도구와 유틸리티가 포함되어 있다. 이들 개발 도구 각각에 대한 자세한 설명은 이 책의 범위를 벗어나지만, 어떠한 것들을 사용할 수 있는지에 대해서는 간단히 살펴볼 만하다. 보다 자세한 내용을 원한다면 아래의 주소에 있는 안드로이드 문서를 살펴보자.

```
http://developer.android.com/guide/developing/tools/index.html
```

앞서 말했듯이 ADT 플러그 인은 이들 도구의 대부분을 이클립스 IDE로 편리하게 통합하는데, 아래에 나열된 것들을 포함한 이들 도구는 DDMS 퍼스펙티브를 통해 접근할 수 있다.

- ✓ **안드로이드 에뮬레이터** 여러분이 현재 개발에 사용하고 있는 컴퓨터상에서 실행되도록 만들어진 안드로이드 가상 머신의 한 구현. 여러분은 이 에뮬레이터를 사용해 여러분의 안드로이드 애플리케이션을 테스트하고 디버그할 수 있다.

- ✓ **Dalvik 디버그 모니터링 서비스**DDMS, Dalvik Debug Monitoring Service 여러분의 애플리케이션을 디버깅하고 있는 Dalvik 가상 머신을 모니터하고 제어하려면 DDMS 퍼스펙티브를 사용한다.
- ✓ **안드로이드 에셋 패키징 도구**AAPT, Android Asset Packaging Tool 배포 가능한 안드로이드 패키지 파일(.apk)을 생성한다.
- ✓ **안드로이드 디버그 브리지**ADB, Android Debug Bridge ADB는 실행 중인 에뮬레이터에 대한 연결을 제공하는 클라이언트-서버 애플리케이션이다. 이를 통해 파일을 복사하고, 컴파일된 애플리케이션 패키지(.apk)를 설치하며, 셸 명령을 실행할 수 있다.

또한 아래에 있는 추가 도구 역시 사용할 수 있다.

- ✓ **SQLite3** 안드로이드에 의해 생성되고 사용되는 SQLite 데이터베이스 파일을 접근하는 데 사용할 수 있는 데이터베이스 도구
- ✓ **Traceview** 안드로이드 애플리케이션의 추적 로그trace logs를 보기 위한 그래픽 분석 도구
- ✓ **MkSDCard** 에뮬레이터가 외부 저장 카드를 시뮬레이션하기 위해 사용할 수 있는 SDCard 디스크 이미지를 생성한다.
- ✓ **dx** 자바 .class 바이트코드를 안드로이드 .dex 바이트코드로 변환한다.
- ✓ **activityCreator** ADT 플러그 인 없이 안드로이드 애플리케이션을 컴파일하는 데 사용할 수 있는 Ant 빌드 파일을 빌드하는 스크립트.

그럼 이 가운데 중요도가 높은 도구 몇 가지를 좀더 자세히 살펴보자.

안드로이드 에뮬레이터

안드로이드 에뮬레이터는 여러분의 애플리케이션을 테스트하고 디버그하기 위한 완벽한 도구로서, 특히 테스트를 위한 실제 장비를 가지고 있지 않을 경우(또는 위험을 무릅쓰고 싶지 않을 경우) 그 가치가 더욱 빛난다.

안드로이드 에뮬레이터는 Dalvik 가상 머신의 한 구현으로서, 다른 안드로이드 폰과 마찬가지로 안드로이드 애플리케이션을 실행하기 위한 유효한 플랫폼이다. 어떠한 특정 하드웨어와도 결부되어 있지 않기 때문에, 여러분의 애플리케이션을 테스트하는 데 사용할 수 있는 훌륭한 기준선이 된다.

안드로이드 에뮬레이터에는 서로 다른 하드웨어 구성을 표현하기 위해 대체해 사용할 수 있는 많은 사용자 인터페이스가 존재하는데, 이들 각각은 다양한 모바일 장치 종류를 시뮬레이션하기 위해 서로 다른 화면 크기, 해상도, 방향, 하드웨어 기능을 가진다.

또한 안드로이드 에뮬레이터에는 애플리케이션을 디버깅하는 동안 인터넷 연결 속도와 지연 속도를 조절할 수 있는 기능을 갖춘 완전한 네트워크 연결이 제공된다. 뿐만 아니라 전화와 SMS 메시지의 발신 및 수신을 시뮬레이션할 수 있다.

ADT 플러그 인은 여러분이 프로젝트를 실행하거나 디버그할 때 자동으로 안드로이드 에뮬레이터가 띄워지도록 이를 이클립스에 통합하고 있다. 만일 ADT 플러그 인을 사용하고 있지 않거나 에뮬레이터를 이클립스 외부에서 사용하고자 하는 경우에는, 텔넷으로 에뮬레이터에 접속해 이를 콘솔상에서 제어할 수 있다. 에뮬레이터 제어에 관한 보다 자세한 사항은 `http://developer.android.com/guide/developing/tools/emulator.html`에 있는 문서를 참고하자.

안드로이드 에뮬레이터는 현재 카메라, 진동, LED, 실제 전화 통화, 가속도 센서, USB 연결, 블루투스, 오디오 캡쳐, 배터리 충전 수준, SD 카드 삽입/제거 등 안드로이드가 지원하는 모든 모바일 하드웨어 기능을 구현하고 있지는 않다.

Dalvik 디버그 모니터 서비스(DDMS)

안드로이드 에뮬레이터는 여러분의 애플리케이션이 어떤 모습으로 나타나고, 행동하며, 상호작용하는지를 볼 수 있도록 해주지만, 그 이면에서 일어나는 실제적인 것을 들여다보려면 DDMS가 필요하다. Dalvik 디버그 모니터링 서비스는 활성화된 프로세스에 명령을 보내고, 스택과 힙을 살펴보며, 활성화된 스레드를 감시 및 중단하고, 활성화된 모든 에뮬레이터의 파일시스템을 탐색할 수 있도록 해주는 강력한 디버깅 도구다.

이클립스의 DDMS 퍼스펙티브는 이 외에도 에뮬레이터의 화면을 캡쳐하고 LogCat이 생성한 로그를 간단히 살펴볼 수 있는 기능을 제공한다.

ADT 플러그 인을 사용하고 있는 경우, 이 DDMS는 이클립스 안에 완전 통합되며, DDMS 퍼스펙티브를 통해 사용할 수 있다. 만일 ADT 플러그 인이나 이클립스를 사용하고 있지 않다면, 명령행을 통해 DDMS를 실행할 수 있다. 이렇게 실행된 DDMS는 현재 실행 중인 모든 에뮬레이터에 자동으로 접속할 것이다.

안드로이드 디버그 브리지(ADB)

안드로이드 디버그 브리지(ADB)는 안드로이드 에뮬레이터나 장치에 접속할 수 있게끔 해주는 클라이언트-서버 애플리케이션이다. 이는 에뮬레이터상에서 실행되는 데몬, 여러분의 개발 하드웨어상에서 실행되는 서비스, 그리고 이 서비스를 통해 데몬과 통신하는 클라이언트 애플리케이션(DDMS 같은) 이렇게 세 가지 컴포넌트로 구성된다.

ADB는 여러분의 개발 하드웨어와 안드로이드 장치/에뮬레이터 간에 설치되는 통신 도관導管으로서, 대상 장치에 애플리케이션을 설치하고, 파일을 넣고 빼며, 셸 명령을 실행할 수 있게 해준다. 이 장치 셸을 사용하면 로깅logging 설정을 바꿀 수 있고, 장치상에 사용 가능한 SQLite 데이터베이스를 질의 또는 수정할 수 있다.

ADT 도구는 애플리케이션 설치 및 업데이트, 로그 파일, 파일 전송(DDMS 퍼스펙티브를 통해) 등 흔히 있는 ADB와의 많은 상호작용을 자동화하고 단순화한다.

ADB로 할 수 있는 것에 대해 더 알고 싶다면 `http://developer.android.com/guide/developing/tools/adb.html`에 있는 문서를 참고하자.

요 약

이번 장에서는 안드로이드 SDK를 다운로드받아 설치하는 방법과 더불어 윈도우, 맥 OS, 리눅스 플랫폼에서 이클립스를 사용해 개발 환경을 구축하는 방법 그리고 여러분의 프로젝트를 위한 실행 및 디버그 구성을 생성하는 방법에 대해 살펴봤다. 여러분은 새로운 프로젝트 생성을 단순화하고 개발 주기를 간소화하기 위한 ADT 플러그 인의 설치 및 사용법을 배웠다.

또한 모바일 애플리케이션 개발을 위한 몇 가지 설계 고려사항, 특히 프로세서 속도의 증가보다 긴 배터리 수명과 작은 크기가 더 우선시 되는 경우에 있어서의 속도 및 효율성을 위한 최적화의 중요성에 대해 살펴봤다.

여느 모바일 개발이나 마찬가지로, 작은 화면과 더불어 느리고 비싼데다가 신뢰성이 떨어질 수 있는 모바일 데이터 연결을 위한 설계 시 고려해야 하는 사항들이 존재한다.

할 일 목록 애플리케이션을 만들고 난 뒤에는, 여러분의 애플리케이션을 테스트하고 디버그하기 위해 사용할 안드로이드 에뮬레이터와 개발자 도구에 대해 살펴봤다.

이번 장에서 여러분이 한 일은 다음과 같다.

- ✓ 안드로이드 SDK를 다운로드받아 설치했다.
- ✓ 이클립스에 개발 환경을 설정하고, ADT 플러그 인을 다운로드받아 설치했다.
- ✓ 여러분의 첫 번째 애플리케이션을 만들었으며, 이 애플리케이션의 동작 방식에 대해 배웠다.
- ✓ 여러분의 프로젝트를 위한 실행 및 디버그 시작 구성을 설정했다.
- ✓ 안드로이드 애플리케이션의 종류에 대해 배웠다.
- ✓ 모바일 장치 설계 고려사항 몇 가지와 구체적인 안드로이드 디자인 프랙티스 몇 가지를 살펴봤다.
- ✓ 할 일 목록 애플리케이션을 만들었다.
- ✓ 안드로이드 에뮬레이터와 개발자 도구를 살펴봤다.

다음 장에서는 액티비티와 애플리케이션 설계에 초점을 맞춘다. 여러분은 안드로이드 매니페스트를 사용해 애플리케이션 설정을 정의하는 방법과 여러분의 UI 레이아웃 및 애플리케이션 리소스를 외부화하는 방법에 대해 배울 것이다. 또한 안드로이드 애플리케이션의 실행 주기와 상태에 관해 좀더 알게 될 것이다.