시작하세요!

아이폰 프로그래밍

iPhone SDK를 이용한 아이폰 개발

데이브 마크 · 제프 라마시 지음 / 이준호 · 정지웅 · 정일영 옮김

Apress® 위키북스

01

정글에 온 것을 환영한다

정말로 아이폰 애플리케이션을 작성하고 싶은가? 글쎄, 여러분을 뭐라고 탓할 수는 없을 것 같다. 사실 아이폰이야말로 오랜 기간 보아왔던 그 어떤 플랫폼보다 흥미롭고 새로운 플랫폼일 것이기 때문이다. 그렇다. 확실히, 최근에 나온 가장 흥미로운 모바일 플랫폼이다. 게다가, 얼마 전부터는 특별히 애플Apple이 제공하는 문서화가 잘 된 아이폰 애플리케이션 개발을 위한 일련의 도구들이 준비되어 있기도 하다.

이 책은 어떤 종류의 책인가

이 책은 여러분이 아이폰 애플리케이션을 작성할 때 그 시작과정에 필요한 길을 알려주는 안내자 역할을 하는 책이다. 우리의 가장 주된 목표는 여러분이 아이폰 애플리케이션의 동작 방식과 구성을 이해함으로써 초기의 학습 곡선Learning Curve을 빠르게 극복할 수 있도록 도와주는 데 있다. 책을 읽어나가면서 여러분은 계속해서 작은 애플리케이션들을 만들 것이다. 이 애플리케이션들은 각각 특정한 아이폰 기능을 구현하는 데 초점을 맞추고 있으며, 이를 통해 특정한 아이폰 기능들을 제어하고 상호작용하는 방법을 배울 수 있을 것이다. 이 책을 통해 얻은 기본적인 이해를 바탕으로 여러분만의 창의성과 결단력, 그리고 애플이 제공하는 잘 작성된 광범위한 분야의 문서들을 모두 한데 모은다면 비로소 전문가적인 아이폰 애플리케이션을 만들기 위해 필요한 모든 것을 갖추었다고 자신할 수 있을 것이다.

시작하기 전에 준비해 두어야 하는 것들

아이폰 소프트웨어를 작성하기에 앞서, 몇 가지 준비할 것이 있다. 입문자라면 레오파드 운영체제(OS X 10.5.3 이상)가 구동되는 인텔 기반 매킨토시를 준비하자. 사실 랩톱이든 데스크톱이든 상관없이, 2006년 이후에 출시된 매킨토시 컴퓨터라면 어떤 기종이든지 무방하다. 최상위 모델이 필요한 것도 아니기 때문에 맥북MacBook이나 맥미니Mac Mini도 훌륭한 환경이라고 볼 수 있다. 구모델이나, 저사양 모델이라면, 램 업그레이드를 좀 해두는 편이 낫다. 어쨌든 간에, 등록된 아이폰 개발자로 가입하는 과정은 반드시 필요하다. 아이폰 소프트웨어 개발 도구 SDK를 다운로드하려면 먼저 애플 사이트에서 등록과정을 거쳐야 한다.

가입을 위해 http://developer.apple.com/iphone 사이트에 들어가면 그림 1-1과 유사한 화면이 보일 것이다. 페이지 어딘가에 최신 버전의 iPhone SDK에 대한 링크가 있을 것이다. 링크를 클릭하면 세 가지 옵션이 나오는 가입페이지가 나타난다

가장 단순한 (뿐만 아니라 무료인) 옵션은 Download the Free SDK이다. 링크를 누르면 Apple ID를 입력하라는 프롬프트가 보일 것이다. 여러분의 Apple ID를 사용해서 로그인하자. 아직 없다면 Create Apple ID 버튼을 눌러서 계정을 생성한 후에 로그인하면 된다. 로그인을 하면 아이폰 개발 페이지의 메인 화면으로 이동한다. SDK 다운로드 링크뿐만 아니라 풍부한 문서, 비디오, 샘플 코드는 물론 아이폰 애플리케이션 개발에 필요한 핵심내용들을 배울 수 있는 모든 자료들을 찾아볼 수 있을 것이다

iPhone SDK에 포함된 도구 가운데 가장 중요한 것은 애플이 내놓은 통합개발환경(IDE)인 Xcode다. Xcode는 소스코드를 작성하고, 디버깅하고, 컴파일할 수 있으며, 작성한 애플리케이션에 대한 성능 튜닝도 할 수도 있는 도구다. 아마 이 책을 다 읽을 무렵이면 여러분은 XCode의 열렬한 애호가가 되어 있을 것이다.

무료 SDK는 대부분의 아이폰 프로그램을 맥에서 돌려볼 수 있는 시뮬레이터도 포함하고 있다. 무료 SDK는 아이폰상에서 프로그래밍하는 방법을 배우는 용도로는 더 이상 부족함이 없다. 하지만, 무료 버전으로는 실제로 여러분의 애플리케이션을 실제 아이폰 또는 아이팟 터치로 다운로드하는 것을 허용하지 않는다. 또한 애플리케이션을 애플 아이폰 앱스토어에 배포할 수도 없다. 이를 위해서는 유료인 다른 두 가지 프로그램 가운데 하나에 가입해야 한다.

NOTE

시뮬레이터는 아이폰의 가속도 센서accelerometer나 카메라 같은 하드웨어 의존적인 기능들은 지원하지 않는다. 이미 알고 있겠지만 이런 것들은 따로 옵션을 조정할 필요가 있다.

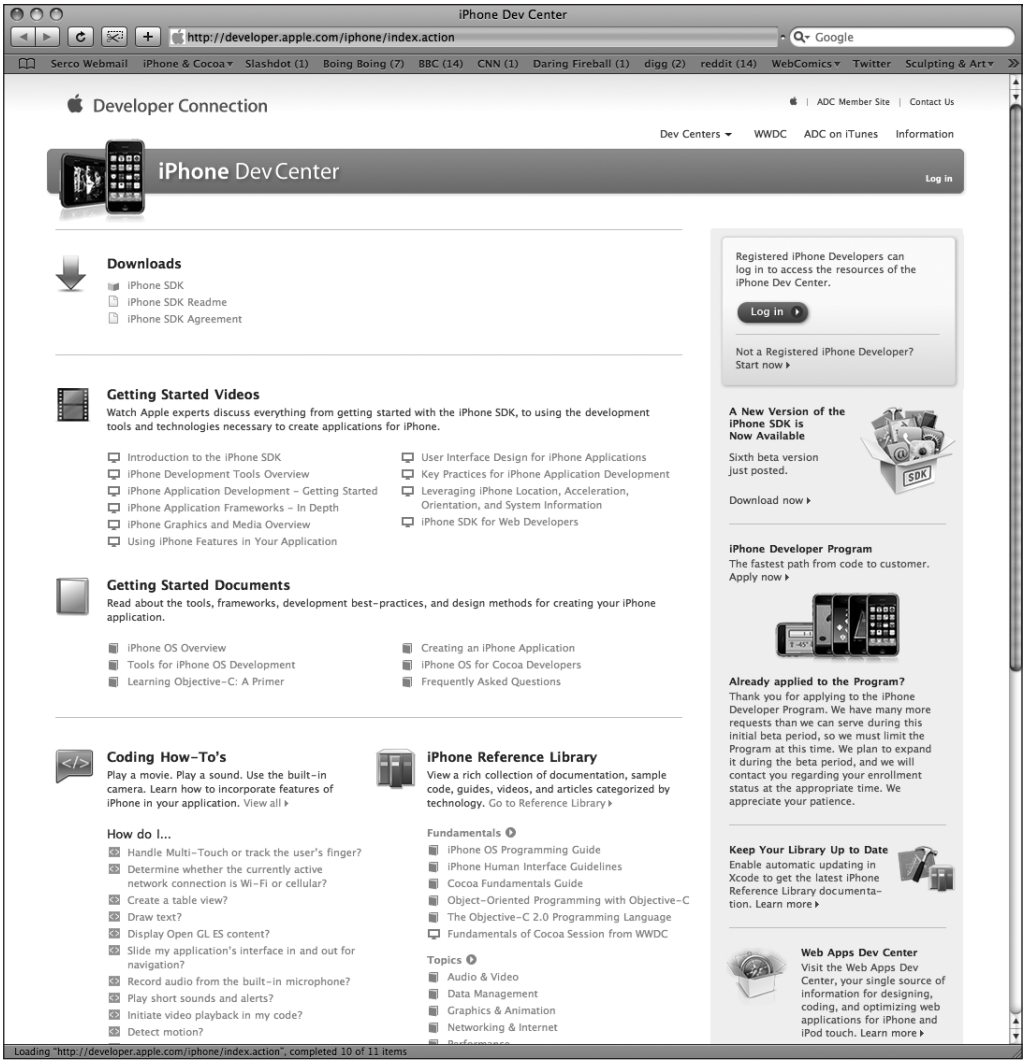

그림 1-1. Apple iPhone Dev Center 웹사이트

표준 프로그램은 99달러로 책정되어 있다. 개발 도구와 기술 지원 자료, 애플리케이션을 애플 앱스토어에 배포할 수 있는 권리, 그리고 가장 중요한, 여러분의 코드를 시뮬레이터가 아닌 아이폰에서 테스트하고 디버깅할 수 있는 일련의 도구들을 제공한다.

기업용 프로그램은 299달러이며, 아이폰과 아이팟 터치를 위한 독점적인 사내in-house 애플리케이션을 개발할 수 있도록 설계되어 있다.

두 프로그램에 대한 좀 더 자세한 내용은 http://developer.apple.com/iphone/program/를 참고하기 바란다.

아이폰은 다른 기업의 무선 인프라를 이용해서 네트워크에 항상 연결될 수 있는 모바일 디바이스이기 때문에 애플은 결함여부나 별도의 인허가 없이도 프로그램을 작성하고 배포할 수 있었던 맥 개발자들보다 훨씬 더 많은 제약을 아이폰 개발자들에게 부과했다.

사실 애플의 본래 의도는 제약을 추가하려는 것보다는 악성 프로그램 또는 잘못 작성된 프로그램들이 공유 네트워크의 성능을 저하시킬 수 있는 기회를 최소화하기 위한 것이라고 보는 편이 맞다. 자, 이쯤 되면 아이폰 개발이 마치 불구덩이를 뛰어넘는 것처럼 어렵게 느껴질 수도 있을 것이다. 하지만, 애플은 가능한 한 개발자들이 괴로워하지 않을 개발 프로세스를 만들려고 많은 노력을 기울이고 있다. 그리고 99달러라는 금액이 여전히 비주얼 스튜디오나 마이크로소프트의 개발 도구를 구입하는 것보다는 훨씬 저렴한 금액이라는 사실에 주목할 필요가 있다.

물론, 한 가지 분명한 사실은 아이폰은 반드시 구입해야 한다는 사실이다. 대부분의 코드를 아이폰 시뮬레이터를 사용해서 테스트할 수는 있겠지만, 모든 프로그램이 다 제대로 돌아가는 것은 아니다. 특히 애플리케이션의 공개 릴리즈를 고려하고 있다면 실제 아이폰 환경에서 신중하게 테스트해 볼 필요가 있다.

NOTE

표준 또는 기업용 프로그램에 가입하려 한다면 지금 당장 가입하는 편이 좋다. 승인과정에 시간이 꽤 걸리고, 애플리케이션을 아이폰 또는 아이팟 터치에서 구동하기 위한 승인 과정이 별도로 필요하기 때문이다. 너무 걱정할 필요는 없다. 이 책의 초반 몇 장에 나오는 모든 프로젝트들과 이 책에 나오는 대부분의 애플리케이션들은 아이폰 시뮬레이터에서도 잘 동작한다.

시작하기 전에 알아두어야 할 것들

이 책은 여러분이 이미 프로그래밍에 대해 다소간의 사전지식이 있다고 가정한다. 특히, 객체지향 프로그래밍의 기본을 이해하고 있다고 가정한다. 예를 들면 객체, 순환문, 변수와 같은 개념들을 이미 알고 있다고 가정한다. 또, 오브젝티브C 프로그래밍 언어에도 이미 친숙하다고 가정한다. 이 책의 거의 모든 부분에서 사용하게 될 SDK에 포함된 코코아 터치Cocoa Touch는 개발언어로 오브젝티브C 2.0을 사용한다. 하지만, 오브젝티브C의 최신 변경사항 몇 개 정도를 조금 모른다고 해서 너무 걱정할 필요는 없다. 우리가 사용상의 이점을 직접 체감한 2.0 버전만의 특성에 대해서는 따로 책에서 분명하게 강조할 것이기 때문이다. 이 특성들이 어떻게 동작하고 왜 우리가 이 특징을 사용했는지도 물론 설명할 것이다.

아이폰과도 친숙해져야만 한다. 여러분이 애플리케이션을 작성하고자 여느 플랫폼에서 거쳤던 절차처

럼 아이폰만이 가진 미묘한 차이와 특성을 숙지해두어야 한다. 또한 아이폰 인터페이스와 애플의 아이폰 프로그램 룩앤필Look and Feel과도 친숙해져야만 한다.

오브젝티브C가 처음인가?

오브젝티브C를 이전에 접해본 적이 없는 독자들을 위해 시작에 도움이 될 만한 자료들을 몇 가지 소개하겠다. 먼저, 『Learn Objective-C on the Mac』(Apress, 2008)을 읽어보자. 맥 프로그래밍 전문가인 마크 데일림플과 스콧 내스터Scott Knaster가 쓴 훌륭한 책으로, 주제를 쉽게 접근해가며 서술하는 오브젝티브C 입문서이다.

```
http://www.apress.com/book/view/9781430218159
```

다음으로, 애플 아이폰 개발 센터 사이트에 방문해 『The Objective-C 2.0 Programming Language』를 다운로드 하자. 이 문서는 매우 상세하고 포괄적인 설명이 담겨있는 뛰어난 레퍼런스 가이드다.

```
http://developer.apple.com/iphone/library/ ➥
        documentation/Cocoa/Conceptual/ObjectiveC
```

물론, 문서를 읽기 전에 로그인해야 한다는 사실은 잊지 말자.

아이폰 코딩은 어떤 점이 다른가?

코코아나 그 전신격인 NextSTEP 환경의 개발경험이 전무하다면 아이폰 애플리케이션을 작성하는 데 사용하게 될 애플리케이션 프레임워크인 코코아 터치를 보고 조금 이상하다고 느낄 것이다. 닷넷이나 자바 애플리케이션을 개발할 때 사용하는 다른 일반적인 애플리케이션 프레임워크와는 다른 몇 가지 근본적인 차이가 있기 때문이다. 처음에 조금 당황하게 되더라도 너무 걱정할 필요는 없다.

그냥 예제들을 따라하자. 그러면 금새 적응이 되기 시작할 것이다. 코코아나 NextSTEP 환경에서 프로그래밍을 해본 경험이 있다면 iPhone SDK가 많은 부분에서 친숙하게 느껴질 것이다. 많은 클래스들이 Mac OS X상에서 개발에 사용했던 버전에서 별반 변하지 않았다. 혹여 다르다 할지라도 여러분이 이미 친숙해진 예전 버전과 비슷한 종류의 기본적인 원칙들을 그대로 따르거나 유사한 디자인 패턴을 사용하고 있다. 하지만, 코코아와 코코아 터치 간의 차이점도 분명히 몇 가지 존재하는 것은 사실이다.

여러분이 그동안 어떤 환경을 경험했든지 간에 아이폰 개발과 데스크톱 애플리케이션 개발 간에 존재하는 주요한 차이점들을 주의 깊게 알아 둘 필요가 있다.

한 번에 단 하나의 애플리케이션만 구동한다

운영체제를 제외하고는 아이폰에서는 항상 한 개의 애플리케이션만이 구동될 수 있다. 아이폰이 좀 더 강력한 프로세서를 탑재하거나 메모리가 늘어나게 되면 달라질 수도 있을 것이다. 하지만, 현재 시점에서는 여러분의 코드가 실행되는 동안에는 여러분의 애플리케이션만이 유일하게 동작한다. 사용자가 지금 사용 중인 애플리케이션이 아닐 경우 해당 애플리케이션은 아무것도 할 수 없을 것이다.

단 하나의 윈도우

동시에 여러 개의 프로그램이 존재하고 복수의 윈도우를 생성하고 제어할 수 있는 데스크톱과 랩톱의 운영체제와 다르게 아이폰은 애플리케이션에게 단 1개의 윈도우만을 제공한다. 여러분이 만든 애플리케이션의 사용자와의 모든 상호작용은 이 윈도우 안에서만 이루어지고 윈도우의 크기는 아이폰의 화면 크기로 고정된다.

접근 권한의 제약

컴퓨터상의 프로그램은 사용자가 실행하면 사용자의 모든 것에 접근할 수 있는 반면에, 아이폰은 애플리케이션이 그런 행위를 하는 것 자체를 심각하게 제한한다. 애플리케이션은 단지 애플리케이션이 자체적으로 생성한 아이폰의 파일 시스템의 일부로써 파일을 읽고 쓸 수 있을 뿐이다.

이 영역을 애플리케이션의 샌드박스sandbox라고 부른다. 애플리케이션이 문서, 환경설정 그리고 다른 종류의 데이터들을 저장할 필요가 있을 때 사용하는 공간이다. 또 다른 방법의 제약도 존재하는데, 예를 들어 데스크톱 컴퓨터에서였다면 일반적으로 루트root나 관리자 권한 접근이 필요할 법한 아이폰의 네트워크 포트 중 낮은 번호를 사용하는 대역에는 아예 접근조차 할 수 없다.

응답 시간의 제약

프로그램이 실행되면 애플리케이션을 열고 환경설정과 데이터가 로딩된다. 그리고 메인 뷰가 최대한 빨리 화면에 표시되어야 한다. 물론 이 모든 작업에 걸리는 시간이 몇 초를 넘기지 않아야 한다.

프로그램이 실행되는 동안에 중요한 데이터가 날아갈 수도 있다. 사용자가 홈 버튼을 누르면 아이폰은 홈으로 이동하고, 이때 애플리케이션은 재빨리 모든 것을 저장하고 종료해야 한다. 저장하고 제어권을 포기하는 데 5초 이상의 시간이 걸린다면 애플리케이션 프로세스는 저장이 종료되었는지의 여부와 상관없이 강제적으로 종료된다.

결과적으로, 사용자가 애플리케이션을 종료할 때 데이터가 유실되지 않도록 아이폰 애플리케이션을 신중히 다루어야 할 필요가 있다.

화면 크기의 제약

아이폰의 화면은 정말 멋지다. 처음 소개되었을 때부터 여태까지 계속 소비자용 기기 중에서는 최고 해상도의 화면을 유지하고 있다. 하지만 아이폰 디스플레이의 크기는 그렇게까지 크지는 않은 편이다. 따라서 현재 여러분의 컴퓨터 작업에 사용하는 화면 작업공간보다 훨씬 작은 480×320 픽셀을 사용할 수 있다.

비교를 위해, 이 책이 쓰여질 시점의 애플의 가장 저렴한 아이맥iMac의 해상도를 살펴보면 1680×1050 픽셀임을 알 수 있다. 가장 저렴한 맥북 노트북 컴퓨터마저도 1280×800 픽셀의 해상도를 제공한다. 아예 좀 더 극단적으로 나아가, 애플의 가장 큰 모니터인 30인치 시네마 디스플레이는 2560×1600이라는 터무니 없는 해상도를 제공한다.

시스템 자원의 제약

예전의 프로그래머들이라면 최소한 128MB의 램과 4GB이나 되는 저장공간을 가진 기계에서 어떤 형태로든 자원의 제약이란 게 존재한다는 사실에 실소할지 모르겠지만, 오늘날을 살아가는 우리들에게 이런 점은 엄연한 사실이다.

48KB의 메모리 공간 안에서 복잡한 스프레드시트 애플리케이션을 작성해야 하는 예전의 기기들과는 다르지만, 모든 작업이 그래픽 인터페이스에 기반해 이루어진다는 아이폰의 태생적 특성상, 메모리의 소모가 매우 쉽게 일어나기 때문이다

아이폰 버전은 현재 128MB의 물리적 램을 가지고 있고 시간이 지나면 이후 버전에서는 좀 더 늘어나리라고 예상된다. 현재, 아이폰 메모리의 일부는 화면 버퍼로 사용되고 나머지가 시스템 프로세스에서 사용된다. 보통 메모리의 절반 정도를 애플리케이션이 사용할 수 있다.

64MB라는 공간은 예전의 소형컴퓨터를 생각한다면 충분히 큰 공간이지만, 아이폰의 메모리에서는 또 다른 고려해야 할 사안이 남아있다. Mac OS X와 같은 현대의 컴퓨터 운영체제는 사용하지 않는 메모리의 일부를 이용해 디스크의 스왑 파일Swap file이라는 공간에 저장해 놓는다. 이를 통해서 애플리케이션은 실제 컴퓨터가 가진 것보다 더 많은 메모리 공간을 요청할 수 있는 것이다. 하지만, 아이폰은 애플리케이션 데이터와 같은 휘발성 데이터를 이러한 스왑 파일에 기록할 수 없다. 결과적으로 애플리케이션이 사용할 수 있는 메모리 공간은 실제 폰의 물리적 메모리에서 사용되지 않는 공간만큼으로 줄어들게 된다.

코코아 터치는 남은 메모리가 적으면 애플리케이션이 감지할 수 있는 방식을 내장하고 있다. 이러한 상황이 발생하면 애플리케이션은 반드시 필요하지 않은 메모리를 해제해야 한다. 그렇지 않으면 운영체제에 의해 강제종료를 당할 위험이 있기 때문이다.

코코아 개발 도구들의 누락

코코아 개발경험이 있는 상태에서 아이폰 개발을 시작하는 경우라면 이전에 친숙하게 사용했던 몇몇 도구들이 아이폰 개발환경에서는 빠져 있는 것을 발견할 것이다. iPhone SDK는 코아 데이터Core Data나 코코아 빌딩Cocoa Binding 같은 도구들을 지원하지 않는다. 앞서 코코아 터치가 오브젝티브C 2.0 기반이라고 언급한 바 있다. 하지만, 아이폰은 새로운 언어(오브젝티브C 2.0)의 새로운 핵심기능 중에서 하나를 지원하지 않는다. 다시 말해 코코아 터치는 가비지 컬렉션을 지원하지 않는다.

새로운 기능

코코아 터치가 기존의 코코아의 기능 일부를 빠뜨렸다고 설명했으니, 반대로 코코아가 지원하지 못하거나 최소한 모든 맥에서 지원되지는 않는 iPhone SDK만이 가진 신기능도 언급하는 편이 공평할 것이다. iPhone SDK는 코아 로케이션Core Location을 사용해 폰의 현재 위치 좌표를 알아낼 수 있다. 또한 아이폰은 내장 카메라와 포토 라이브러리를 가지고 있으며, SDK를 통해 이 두 기능에 접근할 수 있는 방법을 제공한다. 아이폰이 어떤 형태로 놓여있고, 어떻게 이동하는지를 감지할 수 있는 가속도 센서 또한 내장하고 있다.

다른 접근법

아이폰은 물리적인 키보드나 마우스를 가지고 있지 않다. 이 말은 즉, 사용자와의 상호작용에 있어 종래의 범용 컴퓨터와는 근본적으로 다른 방식을 사용해야 한다는 것을 의미한다. 다행히, 아이폰은 대부분의 상호작용 과정을 대신 처리해준다. 애플리케이션에 텍스트필드를 추가하면 별도의 부가적인 코드작성 없이도 아이폰이 사용자가 필드를 클릭할 때 키보드를 띄워준다.

이 책의 구성

책의 이후 구성이 어떻게 되어 있는지 간단히 살펴보자.

2장

이 장에서는 XCode의 단짝이라고 할 수 있는 인터페이스 빌더Interface Builder에 대해서 알아보고, 이를 이용해 아이폰 화면에 문자들을 출력하는 간단한 인터페이스를 만들어 본다.

3장

3장에서는 사용자와의 상호작용을 배운다. 사용자의 버튼 입력에 따라 표시된 텍스트를 런타임Runtime에 업데이트하는 간단한 애플리케이션을 만들어 본다.

4장

4장에서는 3장의 결과물을 아이폰의 표준 사용자 인터페이스 컨트롤들을 사용해서 만들어 본다. 사용자에게 선택을 위한 프롬프트를 띄우거나 특별한 상황이 일어났을 때 알림을 사용할 수 있는 경고alert와 시트Sheet의 사용법에 대해서도 알아본다.

5장

5장에서는 자동회전autorotation 처리를 다룬다. 이 메커니즘을 통해 아이폰 애플리케이션은 세로보기portrait 모드와 가로보기landscape 모드 간에 전환을 할 수 있다.

6장

6장에서는 좀 더 발전된 사용자 인터페이스로 넘어가서 멀티뷰 인터페이스를 생성하는 법을 살펴본다. 사용자에게 보여지는 뷰를 런타임에 변경함으로써, 좀 더 복잡한 사용자 인터페이스를 만들 수 있다.

7장

7장에서 선보이는 툴바Toolbar 컨트롤러는 아이폰 사용자 인터페이스의 표준 가운데 하나이다. 7장에서는 이런 류의 인터페이스를 구현하는 방법을 다룬다.

8장

8장에서는 사용자에게 목록형태의 데이터를 보여주는 데 가장 많이 쓰이는 방법인 테이블 뷰table view를 살펴보고, 계층구조형 내비게이션hierarchical navigation에 기반한 애플리케이션의 기초 구성을 다룬다.

9장

아이폰 애플리케이션 인터페이스에서 가장 일반적인 인터페이스 가운데 하나는 계층구조 리스트hierarchical list다. 계층구조 리스트를 통해서 좀 더 많은 데이터나 세부내용을 편하게 조회할 수 있다. 9장에서는 이 리스트를 구현하는 데 필요한 내용들을 다룬다.

10장

10장에서는 사용자들이 애플리케이션 차원에서 선호사항application- level preference을 설정할 수 있는 메커니즘인 애플리케이션 설정application settings을 구현하는 방법을 알아본다.

11장

11장에서는 아이폰의 데이터 관리data management를 살펴본다. 애플리케이션 데이터를 보관하는 객체를 만드는 방법은 물론, 데이터를 아이폰의 파일 시스템에 저장하는 방법과 SQLite라는 임베디드 데이터베이스에 저장하는 방법을 살펴본다.

12장

그림 그리기는 누구나 좋아하는 주제다. 12장에서는 기본적인 그리기 기능을 제공하는 쿼츠Quartz와 OpenGL ES를 사용해서 그리기 기능을 다룬다.

13장

아이폰의 멀티터치 스크린은 사용자가 입력하는 다양한 종류의 제스처gesture를 인식할 수 있다. 13장에서는 두 손가락을 모으면서 화면을 집는 방법pinch이나 손가락으로 아이폰의 화면을 미는 동작swipe과 같은 기본적인 제스처를 감지하는 방법을 배우게 된다. 새로운 제스처를 정의하는 과정을 살펴보고, 어떠한 경우에 새로운 제스처를 정의하는 것이 적합한지 이야기해보도록 하겠다.

14장

아이폰은 코아 로케이션을 통해 위도latitude와 경도longitude를 판단할 수 있다. 코아 코케이션을 사용해 아이폰이 어느 지역에 위치해 있는지를 알아내서, 이 정보를 우리의 세계정복 프로젝트에 사용할 수 있는 방법을 살펴볼 것이다.

15장

15장에서는 아이폰이 어떤 상태로 놓여져 있는지를 알아보는 데 사용할 수 있는 가속도 센서 인터페이스를 살펴본다. 이 정보를 사용해 애플리케이션에 적용해 볼 수 있는 재미난 것들을 함께 살펴볼 것이다.

16장

모든 아이폰은 카메라와 사진 라이브러리를 가지고 있으며, 잘만 다룬다면 이 둘 모두를 애플리케이션에서 자유자재로 활용할 수 있다. 16장에서는 이 활용법에 대해 다룬다.

17장

아이폰은 현재 70여 개국에서 판매되고 있다. 17장에서는 여러분 애플리케이션의 잠재고객을 좀 더 늘릴 수 있도록, 애플리케이션의 모든 부분을 쉽게 다른 언어로 번역할 수 있게 애플리케이션을 작성하는 법을 알아본다.

18장

이 시점에 오면 아이폰 애플리케이션을 만들기 위한 기본적인 구성물들은 모두 습득한 상태라고 할 수 있다. 하지만 이제부터는 무엇을 더 배워야 할까? 18장에서는 다음단계로 iPhone SDK를 마스터하기 위해 필요한 몇 가지 단계들을 살펴본다.

준비가 되었는가?

아이폰은 여러분에게 개발의 기쁨을 선사하는 흥미로운 첨단기술이자 매우 놀라운 새로운 컴퓨팅 플랫폼이다. 아이폰 프로그래밍은 여러분이 이전까지 작업해왔던 그 어느 플랫폼과도 다른 전혀 새로운 형태의 경험이 될 것이다. 모든 것이 친숙해 보이지만 때때로 낯선 것들을 발견할 수도 있다. 하지만, 책에 나오는 예제코드들을 하나하나 따라하다 보면 개념들이 이내 하나가 되어 머릿속에 쏙쏙 이해되기 시작할 것이다

이 책에 나오는 예제들이 단순한 체크리스트 이상의 의미를 지닌다는 것을 명심해야 한다. 여기 나온 것들을 모두 습득한다면 놀랍게도 여러분을 아이폰 개발 구루Guru 정도의 상태로 만들어 줄 수 있다고 장담할 수 있다. 다만, 다음 프로젝트로 건너뛰기 전에 각 장에서 무엇을 했고, 왜 했는지를 반드시 이해하고 넘어가야 한다. 예제 코드를 변경해보는 것을 전혀 두려워할 필요는 없다. 코코아 터치 같은 환경에서 복잡한 코딩을 할 때는 결과를 이리저리 실험해보고 관찰하는 방법이야말로 여러분의 두뇌에 젊음을 유지할 수 있는 최선의 방법이다.

즉, 내 말은 아까 다운로드받았던 iPhone SDK 설치가 다 되었다면 얼른 페이지를 넘기라는 것이다. 아직도 안 넘겼다면 지금 해보자. 준비되었나? 좋다. 이제 시작이다!

02

티키신 달래기

잘 알다시피, 프로그래밍 책에서 첫 번째 프로젝트를 "Hello, World!"라고 하는 게 하나의 전통이 되었다. 우리는 이 전통을 깨볼까 생각했지만 그렇게 큰 결례로 인해 티키신[1]이 고통의 천벌을 내릴지도 몰라 두려웠다. 그래서 이 책에서도 그냥 "Hello, World!"를 쓰기로 했다. 그럼, 시작해볼까?

이번 장에는 Xcode와 인터페이스 빌더Interface Builder를 사용해서 화면에 "Hello, World!" 문자를 출력하는 작은 아이폰 애플리케이션을 만들 것이다. Xcode에서 아이폰 애플리케이션 프로젝트를 만드는 데 어떤 것이 사용되는지, 또 애플리케이션의 사용자 인터페이스를 설계하기 위한 인터페이스 빌더의 세부 항목에는 어떤 것이 있는지 살펴보고, 애플리케이션을 아이폰 시뮬레이터로 실행해 볼 것이다. 그러고 나서 애플리케이션에 아이콘과 고유 식별자unique identifier를 추가해 실제 애플리케이션과 더욱 비슷하게 만들 것이다.

할 일이 많으니, 어서 시작해보자.

Xcode에서 프로젝트 설정하기

지금쯤, 장비에 Xcode와 iPhone SDK를 설치했을 것이다. 또, Apress 웹사이트에서 이 책의 프로젝트 압축파일도 다운로드해야 한다. 다음은 이 책의 홈페이지이다.

```
http://www.apress.com/book/view/9781430216261
```

페이지의 왼쪽에 Book Extras 부분에 Source Code 링크를 찾는다. 압축파일을 풀고, 프로젝트 폴더를 사용하기 좋은 곳에 둔다.

---

1 (옮긴이) 티키(Tiki)는 Māori(또는 폴리네시아) 신화에서 최초의 사람으로 the tiki gods는 인류를 창조한 최초의 신을 말함. "hello world"라는 프로그래밍 배울 때 하는 기본적인 것을 안 하고 가면 인류 창조 신과 같은 태초의 신이 화내니, 하고 가야 한다는 의미. 우리 문화에서는 산신령 달래기 정도의 의미로 생각하면 된다.

전체 프로젝트 파일들이 있어서 간단히 다운로드한 버전을 실행시키는 것보다는 직접 손으로 각각의 프로젝트를 만들어야 이 책에서 더욱 많이 배울 수 있다. 이렇게 해보는 가장 큰 이유는 자신만의 프로젝트를 만들 때, 이 책 전체에서 사용되는 다양한 툴을 직접 사용해 보면서 전문적인 지식도 얻을 수 있기 때문이다. 실제로 버튼과 슬라이더를 클릭해서 드래그해보고, 소스코드를 수정하기 위해 이리저리 옮겨 다니고, 한 버전의 프로그램을 다른 버전으로 만들어 보는 경험은 다른 어떤 것으로도 대신할 수 없다.

그건 그렇고, 첫 번째 프로젝트는 '02 Hello World' 폴더에 있다. 직접 프로젝트를 만들려면 02 Hello World 폴더를 만들고 진행한다.

/Developer/Applications에 있는 Xcode를 실행한다. Xcode가 처음이라도 걱정하지 마라. 우리가 새 프로젝트를 만드는 과정을 안내할 것이다. 이미 익숙하다면 간단히 훑어보기만 하자.

맨 먼저 Xcode를 실행하면 그림 2-1과 같은 환영 화면이 나온다. 환영 창은 아이폰과 Mac OS X 기술 문서, 영상 튜토리얼, 뉴스, 예제 코드, 그리고 여러 가지 유용한 링크를 담고 있다. 이러한 모든 정보는 애플의 개발자 웹사이트와 Xcode의 문서 브라우저에 있으니, 만약 다시 이 화면을 보고 싶지 않으면 닫기 전에 그냥 Show at Launch 체크박스의 체크를 해제한다.

그림 2-1. Xcode 환영 화면

NOTE

처음 Xcode를 띄울 때 아이폰이나 아이팟 터치가 컴퓨터에 연결되어 있다면 그 장치를 개발용으로 사용하고 싶은지 묻는 메시지를 볼지도 모른다. 지금은 Ignore 버튼을 클릭한다. 만약 유료 iPhone Developer Program에 가입하기로 했다면, 아이폰이나 아이팟 터치로 어떻게 개발하고 테스트하는지 알려주는 프로그램 포털에 접근할 수 있을 것이다.

File 메뉴의 [New Project…]를 선택해서 새 프로젝트를 만들거나 ⇧⌘N을 눌러 New Project 도움창 assistant을 띄운다(그림 2-2 참조).

그림 2-2. New Project 도움창, 새 파일을 만들 때 다양한 파일 템플릿에서 선택할 수 있다.

그림 2-2에서 보듯이 창의 왼쪽 부분은 두 부분으로 나뉜다. 아이폰과 Mac OS X. 많은 종류의 Mac OS X용 프로젝트 템플릿이 있지만 한 종류(최소한 이 글을 쓰는 시점에서는)만 아이폰 애플리케이션용이다.

그림 2-2와 같이 iPhone이라는 항목 밑에 있는 Application을 선택하면 오른쪽 윗부분의 창에 여러 아이콘이 보이는데, 각각은 아이폰 애플리케이션의 토대로 사용할 수 있는 개별 프로젝트 템플릿을 나타낸다. View-Based Application이라고 되어 있는 아이콘이 가장 간단한 템플릿이어서, 처음 몇 장에서는 이 템플릿을 사용할 것이다. 다른 것들은 일반적인 아이폰 애플리케이션 인터페이스를 만드는 데 필요한

추가적인 코드나 자원들을 제공한다. 아직 살펴볼 준비가 안 된 내용들이어서 당장 공부하지는 않지만, 나중에 이것들에 대해서도 다루므로 크게 걱정할 필요는 없다.

첫 번째 프로젝트를 위해, View-Based Application 아이콘(그림 2-2에서 선택했던 아이콘)을 클릭하고 Choose 버튼을 클릭한다.

일단 프로젝트 템플릿을 선택하면 그림 2-3에서처럼 새 프로젝트를 표준 저장 시트standard save sheet를 사용해 저장할지 묻는다. 프로젝트 이름으로 Hello World를 입력하고 원하는 곳에 저장한다. 문서 Documents 폴더도 나쁘지 않지만 여러분만의 Xcode 프로젝트 전용 폴더를 만드는 것도 좋다.

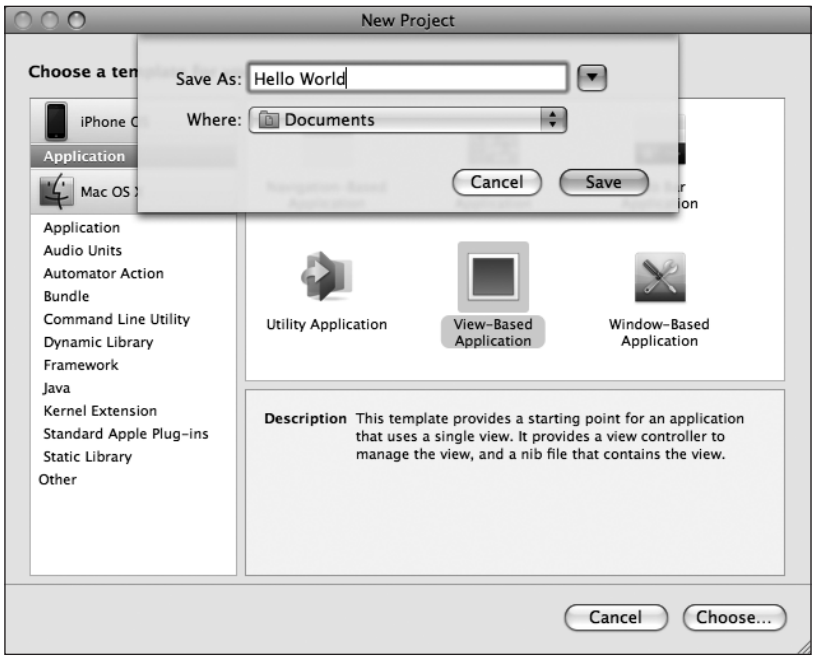

그림 2-3. 프로젝트의 이름과 위치 고르기

Xcode 프로젝트 창

저장 창을 닫고 나면 Xcode는 새 프로젝트를 만들어서 열고 그림 2-4와 같이 새로운 프로젝트 창을 띄운다. 우리 생각에 프로젝트 창은, 처음 띄웠을 때 조금 작게 느껴져서 보통 창을 넓히곤 한다. 이 창에 많은 정보가 빼곡히 차 있고, 이곳이 바로 여러분이 아이폰 애플리케이션을 개발할 때 많은 시간을 보내게 될 곳이다.

프로젝트 창의 위쪽에는 툴바가 있으며, 이 툴바를 통해 자주 사용하는 명령어를 바로바로 실행할 수 있다. 툴바 아래쪽에는 창이 세 부분으로 나뉘어 있다.

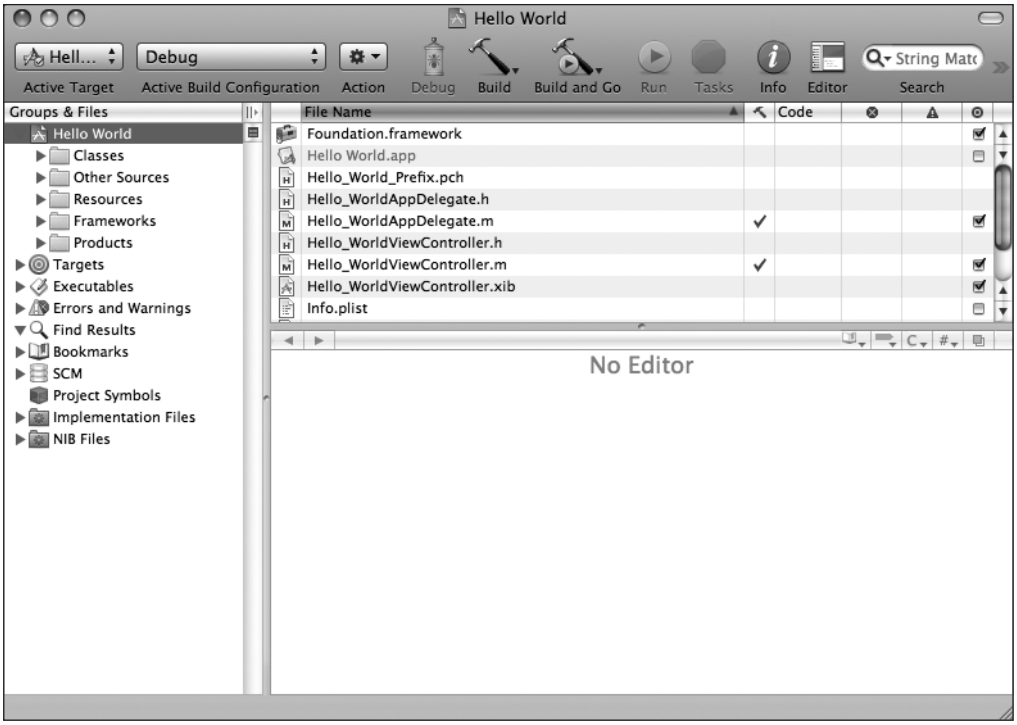

그림 2-4. Xcode에서의 Hello World 프로젝트

프로젝트 창의 왼쪽 부분은 Groups & Files 창이라고 한다. 몇몇 프로젝트 관련 설정과 함께 프로젝트를 구성하는 모든 자원들이 여기 분류되어 있다. 파인더에서와 같이 항목 왼쪽의 작은 삼각형을 클릭하면 항목이 확장되면서 숨겨진 하위 항목들이 나타난다 삼각형을 다시 클릭하면 하위 아이템들은 다시 숨겨진다.

오른쪽 위는 상세보기Detail View 창이라 부르고 Groups & Files 창에서 선택한 항목의 세부정보를 보여준다. 오른쪽 아래 부분은 편집기이다. Groups & Files 창이나 상세보기 창에서 파일 하나를 선택하면, Xcode는 선택한 종류의 파일 내용을 알아서 편집창이나 뷰어로 적절하게 보여준다. 소스코드 같은 편집 가능한 파일들 역시 여기서 편집할 수 있다. 이곳이 바로 애플리케이션의 소스코드를 작성하고 수정하는 곳이다.

이제 새 용어를 훑어봤으니 Groups & Files 창을 살펴보자. 목록의 첫 번째 항목은 프로젝트 이름과 같은데, 이 경우에는 Hello World이다. 이 항목은 프로젝트를 위한 소스코드와 다른 자원들을 모으는 곳이다. 당분간은 Groups & Files 창의 Hello World 밑에 있는 항목 외에는 신경 쓰지 말자.

그림 2-4를 살펴보자. Hello World 왼쪽에 펼쳐보기 삼각형disclosure triangle이 열려있고 Classes, Other Sources, Resources, Frameworks, Products라는 5개의 하위폴더가 있다. 각 하위폴더의 쓰임새에 대해 간단히 살펴보자.

- Classes는 우리가 대부분의 시간을 보낼 곳이다. 여러분이 작성할 대부분의 코드는 여기에 들어가게 되는데, 이곳이 바로 모든 오브젝티브C 클래스들이 원래 있어야 할 곳이기 때문이다. 코드 정리를 위해 Classes 폴더 밑에 자유롭게 하위폴더를 만들어도 된다. 우리는 이 폴더를 다음 장에서 사용할 것이다.
- Other Sources는 오브젝티브C 클래스가 아닌 소스코드를 담는다. 우리는 Other Sources 폴더에서 그다지 많은 시간을 보내지는 않는다. 새 아이폰 애플리케이션 프로젝트를 만들면 두 개의 파일이 이 폴더에 들어간다.
  - Hello_World_Prefix.pch: .pch라는 확장자는 '미리 컴파일한 헤더precompiled header'를 의미한다. 이것은 프로젝트에서 사용하는 외부 프레임워크의 헤더 파일 목록이다. Xcode는 이 파일이 담고 있는 헤더들을 미리 컴파일하는데, 그렇게 하면 Build나 Build and Go를 선택했을 때 프로젝트를 컴파일하는 데 드는 시간이 줄기 때문이다. 가장 일반적으로 사용하는 헤더 파일들은 이미 포함되어 있기 때문에 당분간은 신경쓰지 않아도 된다.
  - main.m: 이것은 애플리케이션의 main() 메서드가 있는 곳이다. 보통 이 파일은 수정하거나 바꿀 필요가 없다.
- Resources는 애플리케이션의 일부로서, 코드가 아닌 파일들을 담고 있다. 여기에서는 애플리케이션의 아이콘 이미지와 다른 이미지, 소리 파일, 동영상 파일, 텍스트 파일, 또는 프로그램을 실행할 때 필요할지도 모르는 프로퍼티 리스트 등을 포함하고 있다. 애플리케이션은 각각의 샌드박스에서 실행되므로 필요한 모든 파일은 여기에 넣어야 한다는 것을 기억해 두자. 이렇게 해야 하는 이유는 허가된 API를 통해 아이폰의 사진 라이브러리나 주소록에 접근하게 해주는 것을 제외하면 아이폰의 다른 곳에 있는 파일들에는 접근할 수 없기 때문이다. 이 폴더에는 아래의 세 항목이 포함되어 있어야 한다.
  - Hello_WorldViewController.xib: 이 파일은 인터페이스 빌더에서 사용하는 정보를 담고 있으며 이 장 조금 뒤에서 다룰 것이다.
  - Info.plist: 이것은 애플리케이션의 정보를 담는 프로퍼티 리스트이다. 역시 이 장 조금 뒤에서 다시 살펴볼 것이다.

  - MainWindow.xib: 이것은 애플리케이션의 기본 인터페이스 빌더 (또는 "nib") 파일이다. 이번 장에서 만들 애플리케이션처럼 간단한 경우, 보통 이 파일을 손댈 필요가 없다. 이어지는 장에서 좀 더 복잡한 인터페이스를 설계할 때, 이 파일을 다루고 더 깊이 살펴볼 것이다.

- Framework는 코드는 물론 이미지나 소리 파일들과 같은 자원들이 담긴 특별한 종류의 라이브러리이다. 이 폴더에 추가한 어떤 프레임워크나 라이브러리는 애플리케이션에 링크되고, 코드에서 그 프레임워크나 라이브러리의 객체, 함수, 자원을 사용할 수 있다. 가장 빈번히 필요한 프레임워크들과 라이브러리들은 기본적으로 프로젝트에 링크되므로 대부분 이 폴더를 다룰 필요가 없다. 그렇지만 간혹 사용하는 라이브러리나 프레임워크가 기본으로 포함되지 않는 경우 경우도 있는데, 애플리케이션에서 이러한 이것들을 어떻게 링크하는지에 관해서는 이 책의 후반부에서 다룬다.

- Products는 이 프로젝트가 컴파일해서 생성한 애플리케이션을 담는다. Products를 펼치면 Hello World.app라는 항목을 볼 수 있다. 이것이 프로젝트가 생성한 애플리케이션이다. Hello World.app는 이 프로젝트의 유일한 결과물이다. 지금은 Hello World.app가 목록에서 빨간색으로 되어 있는데, 이것은 파일을 찾을 수 없음을 의미한다. 이렇게 되는 것은 아직까지 프로젝트를 컴파일하지 않았기 때문이다. 파일의 이름을 빨간색으로 표시하는 것은 실제 물리적 파일을 찾지 못한다는 Xcode식 표현이다.

NOTE

> 프로젝트의 Groups & Files 창에 있는 '폴더들'은 Mac의 파일 시스템과 꼭 일치하지는 않는다. 이것은 애플리케이션을 개발하는 동안 모든 것을 정리하게 도와주고 찾고 싶은 것을 더 빠르고 쉽게 찾아주는 Xcode 안에서의 논리적인 묶음이다. 프로젝트의 실제 폴더를 확인하면 Classes 폴더는 있지만 Other Sources나 Resource 폴더는 없다는 것을 알 수 있다. 이 두 폴더가 담고 있는 항목들은 프로젝트의 종종 루트 디렉토리에 저장되지만 어디에도(원한다면 프로젝트 폴더 바깥이라도) 저장할 수 있다. Xcode 내부에서의 계층은 시스템의 계층과는 완전히 독립적이다. 예를 들어 Xcode에서 Classes 폴더의 파일을 다른 곳으로 옮겨도 하드 드라이브의 파일 위치는 바뀌지 않는다.

인터페이스 빌더 소개

이제 Xcode의 기본은 익숙할 테고, 아이폰 소프트웨어를 개발할 때 사용하는 막강 2인조의 다른 한쪽인 인터페이스 빌더(보통 IB라 부르는)를 살펴보기로 하자.

프로젝트 창의 Groups & Files 목록에서 Resources 묶음을 펼친 후에 파일 Hello_WorldViewController.xib를 더블클릭한다. 이렇게 하면 파일이 인터페이스 빌더에서 열린다. 만약 인터페이스 빌더를 처음

사용한다면 그림 2-5와 같은 창이 나올 것이다. 이미 인터페이스 빌더를 사용했었다면, 창들은 마지막으로 사용할 때 두었던 상태대로 나온다.

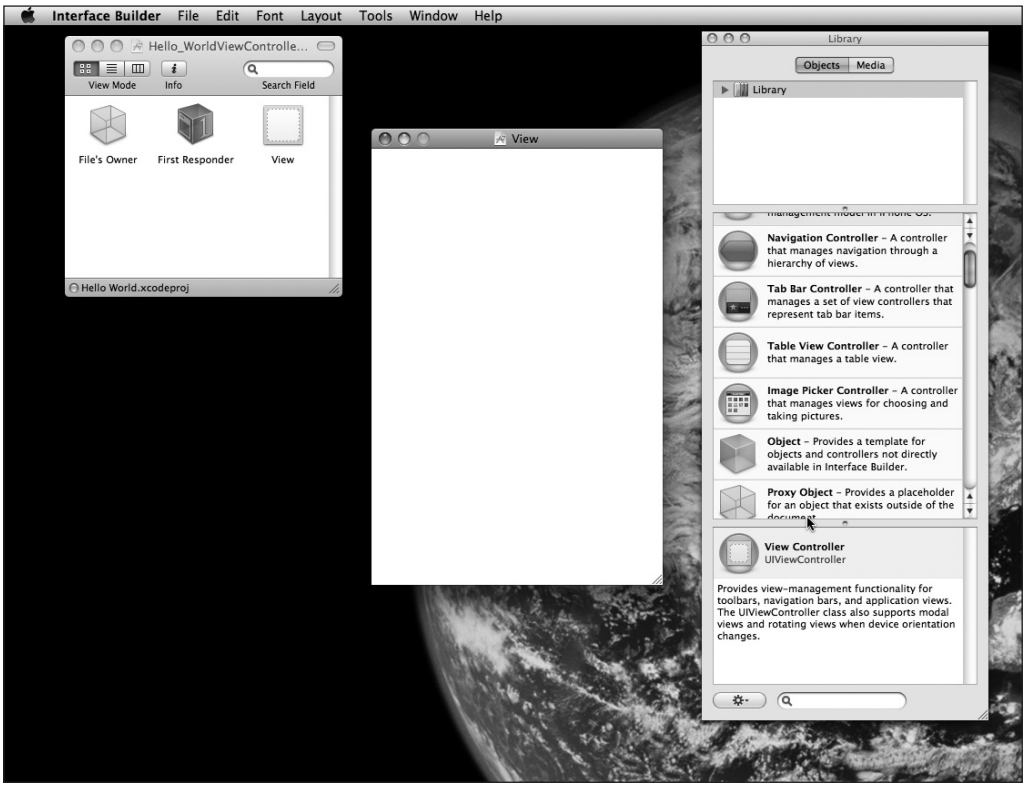

그림 2-5. 인터페이스 빌더에서 Hello_WorldViewController.xib

NOTE

인터페이스 빌더의 역사는 길다. 1988년 무렵에 생겼고 NextSTEP, OpenSTEP, Mac OS X, 지금은 아이폰용 애플리케이션을 개발하기 위해 사용되고 있다. 인터페이스 빌더는 두 가지 파일 형식을 지원하며, .nib라는 확장자를 사용하는 오래된 형식과 .xib라는 확장자를 사용하는 새로운 형식이 있다. 아이폰 프로젝트의 템플릿은 기본적으로 .xib를 사용하지만 아주 최근까지 모든 인터페이스 빌더 파일은 .nib 확장자였고, 그 결과 대부분의 개발자들은 인터페이스 빌더 파일을 "nib 파일"이라고 부르게 되었다. 인터페이스 빌더 파일은 실제 확장자가 .xib건 .nib이건 보통 nib 파일이라고 부른다. 사실, 애플은 실제로 'nib'와 'nib 파일'이라는 용어를 문서의 곳곳에 쓴다.

Hello_WorldViewController.xib라고 표시된 창(그림 2-5의 왼쪽 위의 창)이 nib의 주요 창이다. 그 창이 여러분의 본거지이고 이 특정 nib 파일의 출발점이다. 처음 두 아이콘을 제외하면(File's Owner와 First Responder), 이 창의 모든 아이콘은 nib 파일을 로드할 때 자동으로 생성되는 오브젝티브C 클래스의

인스턴스 하나를 나타낸다.

버튼의 인스턴스를 만들고 싶은가? 물론, 코드를 작성해서 버튼을 만들 수도 있다. 하지만 보통 인터페이스 빌더를 사용하여 버튼을 만들고 속성을 지정한다(모양, 크기, 레이블 등).

지금 살펴보고 있는 Hello_WorldViewController.xib은 애플리케이션이 시작하면 자동으로 로드되므로 지금은 어떻게 하는지 신경 쓰지 않아도 된다. 여기가 사용자 인터페이스를 구성하는 객체를 만들기 위해 적절한 곳이다.

예를 들어 애플리케이션에 버튼을 추가하고 싶으면 UIButton 타입의 객체 인스턴스가 필요하다. 아래와 같은 코드를 입력해서 만들기도 한다.

```
UIButton *myButton = [[UIButton alloc] initWithFrame:aRect];
```

인터페이스 빌더에서도 인터페이스 객체 팔레트에서 애플리케이션의 메인 윈도우로 버튼을 끌어와 똑같은 작업을 할 수 있다. 인터페이스 빌더를 이용하면 버튼의 속성을 쉽게 설정할 수 있고 버튼은 nib 파일에 저장될 것이므로, 애플리케이션이 시작할 때 버튼은 자동으로 인스턴스화될 것이다. 이것이 어떻게 동작하는지에 관해서는 잠시 후에 알아보기로 하자.

Nib 파일에는 무엇이 있나?

그림 2-5를 보자. 앞에서 언급한 것처럼 Hello_WorldViewController.xib 창(왼쪽 위의 창)이 nib 파일의 메인 창이다. 모든 nib 파일은 똑같이 File's Owner와 First Responder라는 두 개의 아이콘으로 시작한다. 두 아이콘은 자동으로 생기고 지우지 못한다. 이런 점으로 미루어, 아마 여러분은 이 아이콘들이 중요한 파일이라 짐작할 수도 있을 텐데, 실제로도 그렇다.

File's Owner는 어떤 nib 파일에서든 항상 첫 번째 아이콘이고 디스크에서 로드된 nib 파일 객체를 나타낸다. 바꿔 말하면 File's Owner는 nib 파일의 사본을 소유한 객체이다. 지금 설명한 개념이 조금 헷갈려도 지금 당장 신경 쓸 필요는 없다. 나중에 이 부분에 관해서 다시 다룰 것이다.

여기와 다른 nib 파일에서 두 번째 아이콘은 퍼스트 리스폰더(First Responder)라고 불린다. 리스폰더에 대해서는 조금 후에 이야기하겠지만 아주 기본적인 관점에서 보면, 퍼스트 리스폰더는 사용자가 현재 상호작용하는 객체이다. 예를 들어 사용자가 현재 텍스트 필드에 데이터를 입력하는 중이라면 해당 텍스트 필드가 현재의 퍼스트 리스폰더다. 퍼스트 리스폰더는 사용자가 인터페이스와 상호작용할 때마다 바뀌기 때문에, 퍼스트 리스폰더 아이콘을 이용하면 현재 어떤 컨트롤이나 뷰가 퍼스트 리스폰더인지 알아내는 코드를 작성할 필요 없이 퍼스트 리스폰더와 수월하게 통신할 수 있는 수단을 제공한다. 이 부분에 관해서도 뒷부분에서 좀 더 자세히 다룰 예정이니, 지금 당장은 혼란스럽더라도 크게 걱정할 필요는 없다.

처음의 특별한 두 아이콘이 아닌 이 창에 있는 다른 아이콘들은 nib 파일을 로드할 때 생성할 인스턴스 객체를 나타낸다. 여기서는 그림 2-5에서 볼 수 있듯이 View라고 하는 세 번째 아이콘이 있다.

View 아이콘은 UIView 클래스의 인스턴스를 나타낸다. UIView 객체는 사용자가 보고 상호작용할 수 있는 영역이다. 이 애플리케이션에서는 하나의 뷰만 가질 것이고, 따라서 사용자가 볼 수 있는 것은 이 아이콘뿐이다. 나중에 여러 개의 뷰를 갖는 한층 복잡한 애플리케이션을 만들겠지만, 지금 당장은 이것을 사용자가 이 애플리케이션을 사용할 때 볼 수 있는 유일한 뷰라고 생각하자.

NOTE

> 엄밀히 말해서, 여기서 만든 애플리케이션은 실제로 하나 이상의 뷰를 가진다. 버튼, 텍스트 필드, 레이블 등 화면에 보이는 모든 사용자 인터페이스의 구성요소들은 UIView의 하위클래스다. 하지만 독자가 이 책에서 '뷰(view)'라는 용어를 볼 때 단지 실제 UIView 인스턴스만 지칭할 것이며, 따라서 이 애플리케이션은 뷰를 하나만 가진다고 볼 수 있다.

그림 2-5로 돌아가보면 메인 창 근처에 두 개의 다른 창이 열려 있음을 확인할 수 있다. 제목 표시줄title bar에 View라는 단어가 있는 창을 보자. 그 창은 메인 창의 세 번째 아이콘을 나타낸 것이다. 이 창을 닫고 nib 파일의 메인 창의 View 아이콘을 더블클릭하면 이 창이 다시 뜬다. 여기가 사용자 인터페이스를 설계할 수 있는 곳이다. 지금 해보자.

View에 레이블 추가하기

그림 2-5의 가장 오른쪽에 있는 창은 라이브러리(library)인데, 그림 2-6에서 좀 더 자세하게 볼 수 있다. 여러분은 이곳에서 인터페이스 빌더가 지원하는 모든 코코아 터치 객체들을 찾아볼 수 있다. 항목을 라이브러리에서 nib 파일 창으로 끌어놓으면 클래스의 인스턴스가 애플리케이션에 추가된다. 라이브러리 창을 닫더라도 Tools 메뉴에서 Library를 선택해서 다시 나타나게 할 수 있다. 이 팔레트에 있는 항목들은 원래 애플리케이션의 사용자 인터페이스를 만드는 데 사용하는 객체들의 프레임워크인 iPhone UIKit에 있던 것들이다.

그림 2-6. 라이브러리 팔레트(Library palette), 인터페이스에서 사용 가능한 UIKit 내장 객체들을 찾는 곳

코코아 터치에서 UIKit은 코코아에서 AppKit과 같은 역할을 수행한다. 개념적으로 두 프레임워크는 비슷하지만 플랫폼의 차이로 인해, 확실히 많은 차이가 있다. 반면, NSString이나 NSArray와 같은

Foundation 프레임워크 클래스는 코코아와 코코아 터치가 공유한다.

Label이란 이름의 객체를 찾을 때까지 라이브러리 팔레트의 객체 목록을 내려보자(그림 2-7을 보라).

레이블label은 아이폰 화면에 나타나지만 사용자가 직접 수정할 수는 없는 짧은 텍스트를 의미한다. 바로, 뷰에 레이블을 추가할 것이다.

사용자 인터페이스가 계층적이므로, 레이블을 메인 뷰(이름이 View인 뷰)의 하위뷰subview로 추가할 것이다. 인터페이스 빌더는 지능적으로 동작해서 하위뷰가 객체를 허용하지 않으면, 거기로 객체들을 끌지 못한다.

라이브러리에서 View라 부르는 뷰로 레이블을 끌어다 놓으면 UILabel 인스턴스가 애플리케이션의 하위뷰로 추가된다.

계속 진행하면서 해보자. 라이브러리 팔레트에서 레이블을 끌어 View 창에 놓는다. 다 하고 나면 뷰는 그림 2-8처럼 보일 것이다.

레이블을 수정해서 무언가 심오한 내용을 표시하도록 바꿔보자. 방금 만든 레이블을 더블클릭하고 Hello, World!를 입력한다. 다음으로, 그 레이블을 화면 어디든 놓고 싶은 곳에 끌어놓자.

어떤가? 이제 저장하기만 하면 끝이다. File 메뉴의 Save를 선택하고 Xcode로 돌아가서 애플리케이션을 빌드하고 실행한다.

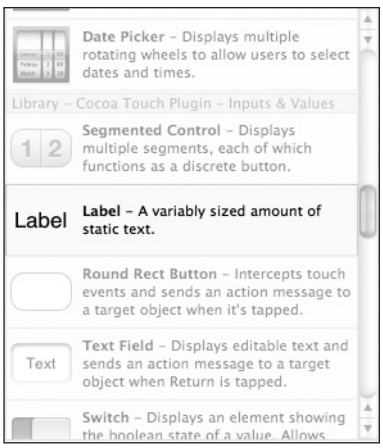

그림 2-7. 라이브러리 팔레트 안의 레이블 객체

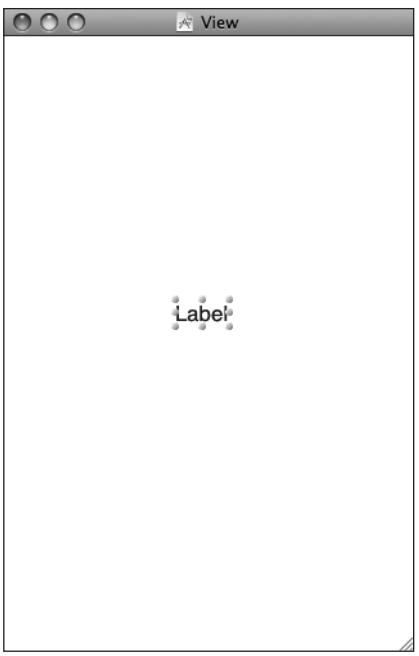

그림 2-8. 애플리케이션의 뷰 창에 레이블 추가하기

Xcode에서 Build 메뉴의 Build and Run을 선택하자(또는 ⌘R을 누른다). Xcode는 애플리케이션을 컴파일하고, 그림 2-9와 같이 아이폰 시뮬레이터에서 애플리케이션을 실행한다.

그림 2-9. 여기 Hello, World 프로그램이 전성기를 구가하고 있다!

작품 감상이 끝나면 꼭 시뮬레이터를 종료하자. Xcode, 인터페이스 빌더, 그리고 시뮬레이터는 모두 독립적인 애플리케이션이다.

잠깐! 그게 다인가? 코드는 하나도 작성하지 않았다.

맞다. 상당히 근사하지 않나?

하지만 글자 크기나 색 바꾸기처럼 레이블의 특성을 바꾸고 싶다면? 그렇게 하려면 코드를 작성해야 하지 않을까?

아니다.

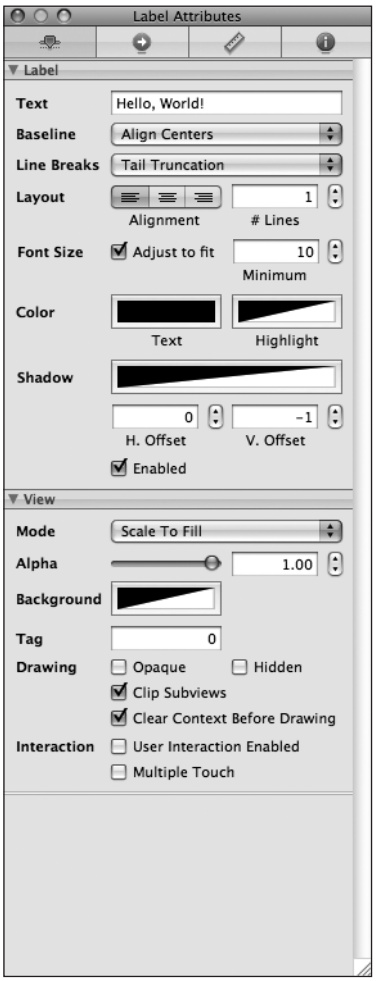

그림 2-10. 레이블의 속성을 보여주는 인스펙터

인터페이스 빌더로 돌아가서 Hello World 레이블을 한 번 클릭해서 선택한다. 이제 ⌘1을 누르거나 Tools 메뉴에서 Inspector를 선택한다. 이렇게 하면 인스펙터inspector라는 창이 뜨는데, 이 창에서 현재 선택된 항목의 속성을 지정할 수 있다(그림 2-10 참조).

인스펙터에서 글자 크기, 색, 그림자 효과drop shadow 같은 많은 항목을 바꿀 수 있다. 따라서 텍스트 필드를 선택했다면 텍스트 필드에서 수정 가능한 속성을 보여주고, 버튼을 선택하면 버튼에서 수정할 수 있는 항목들을 보여주는 것처럼 현재 상황에 맞는 항목을 표시해 준다.

레이블의 모습을 마음 닿는 대로 고치고, 저장하고, Xcode로 돌아가서 Build and Run을 다시 실행한다. 다시 한 번 코드를 전혀 작성하지 않고도, 변경한 내역이 애플리케이션에 나타난다. 인터페이스 빌더를 이용하면 시각적으로 인터페이스를 설계할 수 있어, 지겨운 코드를 작성하는 데 시간을 허비하지 않고도 여러분의 애플리케이션 코드를 작성하는 데 집중할 수 있다.

CAUTION

빌드하고 실행할 때 맥에 아이폰이 연결되어 있으면 예상대로 되지 않을지도 모른다. 간단히 말해 아이폰에서 애플리케이션을 빌드하고 실행하려면 애플의 iPhone developer 프로그램에 가입하고 결제한 후, Xcode의 설정 과정을 적절히 마쳐야 한다. 그 프로그램에 가입할 때 애플이 이 과정을 완료하는 데 필요한 정보를 줄 것이다. 한편, 이 책의 대부분의 프로그램은 아이폰 시뮬레이터에서 잘 동작한다. Build and Run을 선택하기 전에 아이폰이 연결되어 있으면 Project 메뉴의 Active SDK를 선택하고 시뮬레이터(iPhone OS 2.0)를 선택한다.

NOTE

가장 최근의 애플리케이션 개발 환경은 사용자 인터페이스를 시각적으로 만들 수 있는 몇 가지 툴을 포함하고 있다. 인터페이스 빌더와 다른 여러 툴과의 차이에서 두드러진 점은 인터페이스 빌더는 계속 유지보수해야 할 코드를 전혀 만들지 않는다는 것이다. 대신, 인터페이스 빌더는 코드로 하듯이 오브젝티브C 객체를 만들고, 객체를 nib 파일에 직렬화해서 실행할 때 객체들을 메모리에 바로 로드할 수 있다. 이러한 방법은 코드를 만들 때의 많은 문제들을 피하게 하고, 전반적으로 더욱 강력한 접근방법이다.

아이폰을 빛낼 몇 가지 마무리 손질

이 장을 끝내기 전에 실제 아이폰 애플리케이션처럼 느끼게 하는 간단한 끝마무리 작업을 해보자. 먼저, 프로젝트를 실행한다. 시뮬레이터 창이 뜨면 아이폰 홈 버튼(창 맨 아래쪽의 하얀 사각형이 들어있는 까만 버튼)을 클릭하자. 그러고 나면 아이폰 홈 화면(그림 2-11)이 나타날 것이다. 흠, 뭔가 좀 지루한 것을 발견했나?

화면 위의 Hello World 아이콘을 보자. 그렇다, 이 아이콘으로는 충분하지 않다, 어떤가? 이걸 수정하려면 아이콘을 만들고, 크기가 57×57 픽셀인 png 파일로 저장할 필요가 있다. 이미 화면에 있는 버튼의 모양처럼 맞추기 위해 애쓸 필요는 없다. 아이폰이 자동으로 모서리를 둥글게 다듬고 멋진 유리 같은 모습으로 만들어 주기 때문이다. 그냥 보통 사각형 이미지를 만들면 된다. 직접 만들고 싶지 않으면 프로젝트 아카이브(02 Hello World 폴더)에 제공되는 아이콘 이미지를 사용해도 된다.

그림 2-11. 가장 왼쪽 애플리케이션 아이콘은 무척 재미없다

NOTE

애플리케이션 아이콘으로 png 이미지를 사용해야 하지만, 실제로 아이폰 프로젝트에 추가한 모든 이미지를 사용해도 된다. 대부분의 보통 이미지들은 제대로 보이지만 어쩔 수 없이 다른 종류를 사용해야 할 경우가 아니라면 png 파일을 사용하라. Xcode는 자동으로 png 이미지를 빌드할 때 아이폰 애플리케이션에서 사용하는 가장 빠르고 효과적인 이미지 타입으로 최적화한다.

애플리케이션 아이콘을 디자인한 뒤, 그림 2-12처럼 png 파일을 파인더에서 Xcode의 Resources 폴더로 끌어놓거나 Xcode에서 Resources 폴더를 선택하고, Project 메뉴에서 [Add to Project…]를 선택하여 아이콘 이미지 파일을 찾는다.

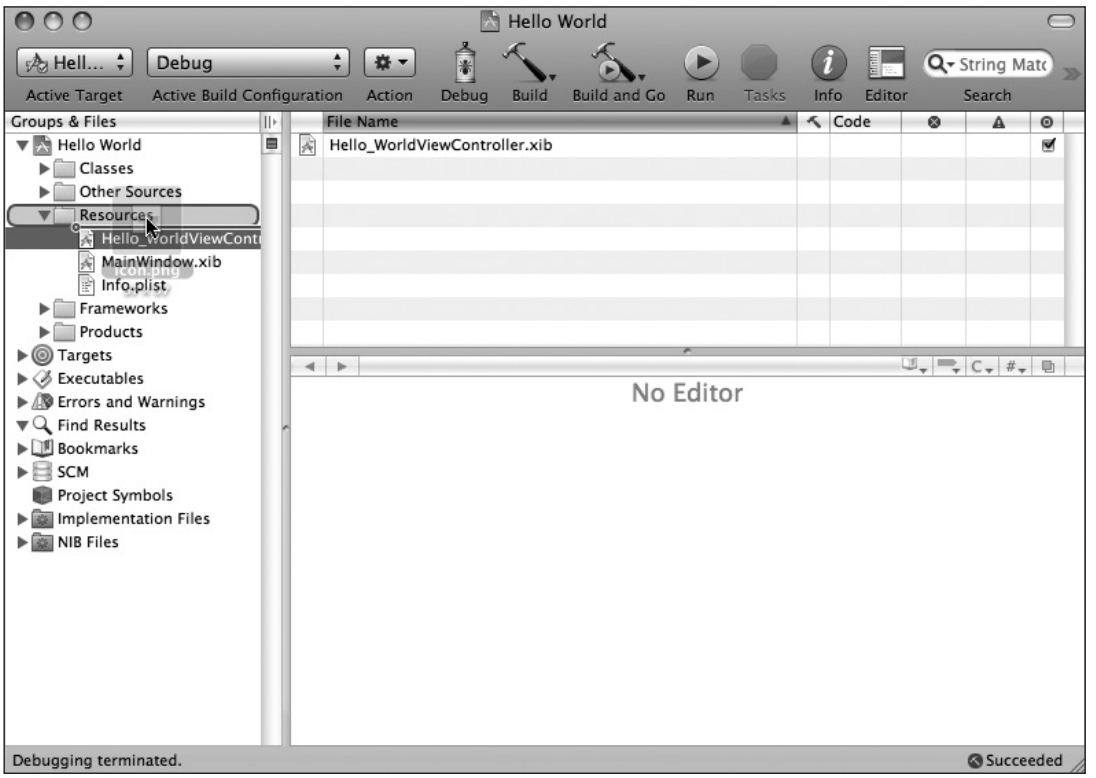

그림 2-12. 아이콘 파일을 Xcode 프로젝트의 Resources 폴더에 끌어놓기

이렇게 하고 나면 Xcode는 몇 가지 세부사항을 입력해 달라는 창을 보여줄 것이다. 이 창에서는 파일을 프로젝트 폴더 안으로 복사하거나 단지 원본 파일의 참조만 프로젝트에 추가하기를 선택할 수 있다. 보통, 파일을 다른 프로젝트와 공유하지 않는다면 Xcode 프로젝트로 복사하는 게 좋다.

자주 사용하는 종류의 파일을 프로젝트에 추가할 때 Xcode는 그 파일에 대해 무엇을 할지 알고 있으므로, 추가적인 작업 없이 이 이미지 파일은 컴파일되어 애플리케이션에 들어간다.

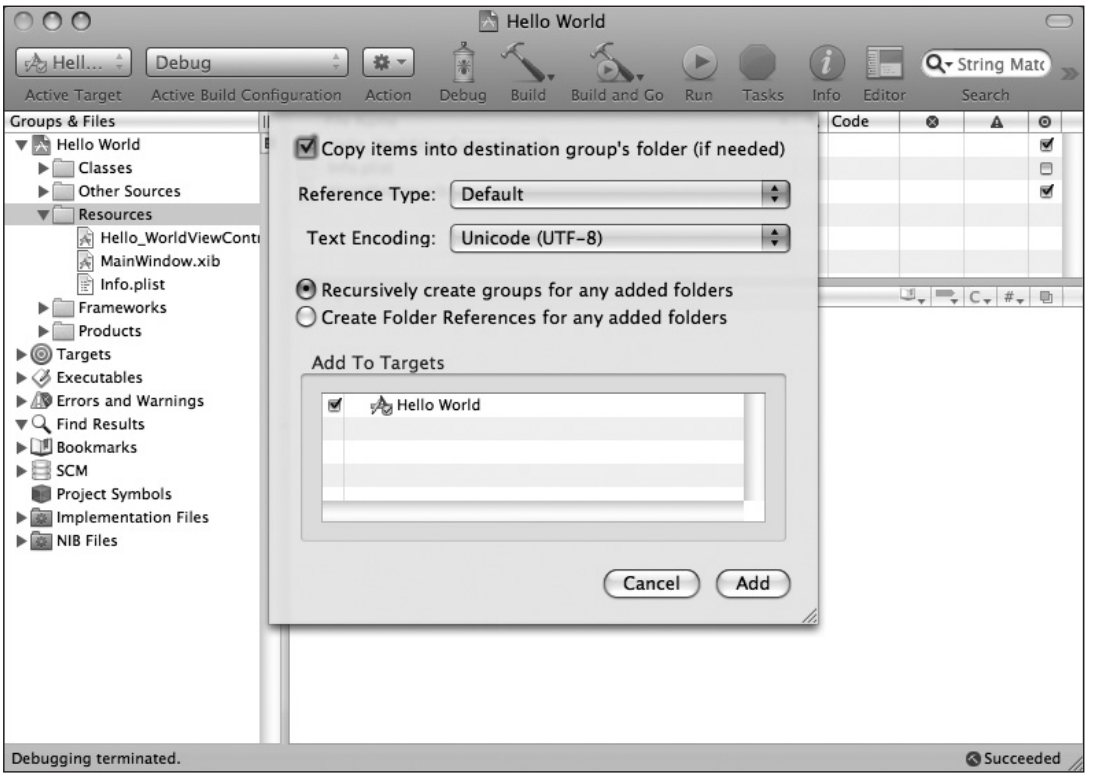

그림 2-13. 파일을 프로젝트에 어떻게 추가할지 선택하기

지금까지 icon.png 이미지 파일을 프로젝트에 추가했는데, 그 결과 이미지가 애플리케이션 번들의 일부로 추가되었다. 다음으로 이 이미지를 우리 애플리케이션의 아이콘으로 지정할 필요가 있다.

Xcode 프로젝트 창에 있는 Groups & Files 창에서 Resources 폴더가 펼쳐져 있지 않다면 펼치고, Info.plist 파일을 한 번 클릭하자. 이것은 애플리케이션에 대한 몇 가지 일반적인 정보를 가진 프로퍼티 리스트property list 파일인데, 다른 속성 중에서 아이콘의 파일명도 포함한다.

Info.plist를 선택하면 속성 목록이 편집 창(그림 2-14)에 나온다. 속성 목록 가운데 왼쪽 칸에 Icon file인 줄을 찾는다. 같은 줄의 오른쪽 칸은 비어있을 것이다. 빈 칸을 더블클릭하고 프로젝트에 추가한 png 파일의 이름을 입력한다.

Key	Value
▼Information Property List	(12 items)
Localization native development re	en
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
Bundle identifier	com.yourcompany.${PRODUCT_NAME:identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
LSRequiresIPhoneOS	☑
Main nib file base name	MainWindow

그림 2-14. 아이콘 파일 지정하기

컴파일과 실행을 위한 준비

컴파일하고 실행하기 전에 Info.plist의 다른 줄을 살펴보자. 대부분의 설정은 그대로 둬도 괜찮지만 Bundle identifier라는 특별한 설정에는 주의가 필요하다. 이것은 애플리케이션을 위한 유일한 식별자이며 항상 설정해야 한다. 애플리케이션을 아이폰 시뮬레이터에서 실행한다면 bundle identifiers의 일반적인 이름 정하기 방법은, com이나 org 다음에 구두점을 찍고 그 뒤에 회사나 조직의 이름을 쓰고 구두점을 찍고 애플리케이션의 이름을 쓰는 것과 같은 최상위 인터넷 도메인을 사용한다. 실제 아이폰에서 여러분의 애플리케이션을 돌리고 싶다면 애플리케이션의 번들 식별자bundle identifier를 만들 때 필요한 절차가 조금 더 있는데, 그것은 뒤쪽 장에서 다룰 것이다. 여기서는 번들 식별자bundle identifier를 com.apress.HelloWorld로 바꿔보자.

그림 2-15. 애플리케이션의 아이콘이 이제 제법 볼만해 졌다!

수정을 하고 나서 컴파일하고 실행하자. 시뮬레이터가 실행되면 홈home으로 가기 위해 하얀 사각 버튼을 누르고, 멋진 새 아이콘을 확인하자. 여기서 만든 아이콘은 그림 2-15에서 확인할 수 있다.

NOTE

아이폰 시뮬레이터의 홈 화면의 오래된 애플리케이션을 지우고 싶으면 간단히 홈 디렉토리의 Library 폴더 안에 있는 Application Support 폴더 속의 iPhone Simulator 폴더를 지우면 된다.

마무리하며

스스로 대견하다고 생각하자. 이번 장에서 많은 것을 이룬 것 같지 않아 보일지라도 실제로 많은 부분을 다루었다. 아이폰 프로젝트 템플릿에 대해 배우고, 애플리케이션을 만들었고, 인터페이스 빌더를 어떻게 쓰는지 봤고, 어떻게 애플리케이션 아이콘과 번들 식별자를 설정하는지 배웠다.

그렇지만 Hello World는 순전히 일방적인 애플리케이션이다. 즉, 사용자에게 정보를 보여주지만 그들에게서 어떤 입력도 받지 않는다. 아이폰 사용자에게 어떻게 입력을 받는지, 입력에 맞게 동작하게 하는지 살펴볼 준비가 되었다면, 크게 심호흡 하고 책장을 넘기자.