Chapter 11

상호작용하는 iOS 아이폰 앱 만들기

앞 장에서 iOS 5 기반의 아이폰 애플리케이션 개발에 사용되는 디자인 패턴에 대해 알아보았다. 이 장에서는 타깃-액션(Target-Action) 패턴과 뷰-컨트롤러 관계에 대해 예제를 통해 조금 더 자세하게 알아보자.

11.1 새 프로젝트 만들기

예제로 화씨(Fahrenheit)를 섭씨(Centigrade)로 변환하는 프로그램을 만들 것이다. 첫 단계는 애플리케이션을 위한 Xcode 프로젝트를 생성하는 것이다. Xcode를 실행하고 환영 화면에서 Create a new Xcode Project를 선택한다. 템플릿 화면에서 왼편의 iOS 아래쪽의 Application 옵션을 선택한 후 Single View Application을 선택한다. Next를 클릭하고, product name과 class prefix에 UnitConverter를 입력한 후 회사 식별자를 입력한다. Device Family 메뉴가 iPhone으로 되어 있음을 확인하고 스토리보드와 유닛 테스트 옵션이 꺼져 있음을 확인한다. 마지막 화면에서 프로젝트 파일을 저장할 위치를 선택한 후 Create를 클릭한다.

11.2 유저 인터페이스 만들기

애플리케이션의 로직을 작성하기 전에 유저 인터페이스부터 디자인한다. 새로운 프로젝트가 만들어질 때 Xcode는 UnitConverterViewController.xib라는 인터페이스 빌더 NIB 파일을 생성한다. 이 파일은 우리가 만들 유저 인터페이스를 저장할 것이므로 왼쪽 화면의 프로젝트 내비게이터에서 이 파일을 선택하여 인터페이스 빌더로 로드한다. 파일이 로드되면 Xcode 툴바의 오른쪽 끝을 선택하여 오른편 화면이 표시되도록 한다. 아래쪽 화면에서 Show the Object library 툴바 메뉴를(3차원 정육면체) 선택하여 UI 컴포넌트가 표시되도록 한다. 아니면 간단하게 View ➞ Utilities ➞ Show Object Library를 선택한다.

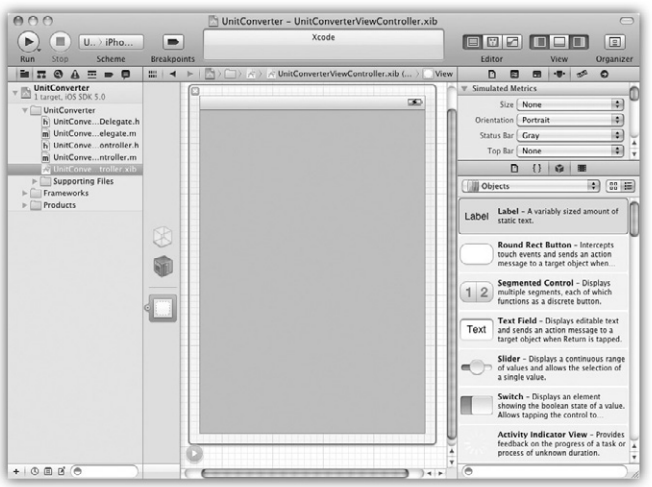

그림 11-1

Object Library 화면에서 Text Field 오브젝트를 뷰 디자인 영역으로 끌어놓는다. 오브젝트의 크기와 위치를 조절하여 그림 11-2처럼 보이게 한다.

Attribute Inspector 화면에서(View ➞ Utilities ➞ Show Attribute Inspector) Placeholder 문자열에 Enter temperature를 입력한다. 이 문장은 텍스트 필드에 밝은 회색으로 보일 것이다.

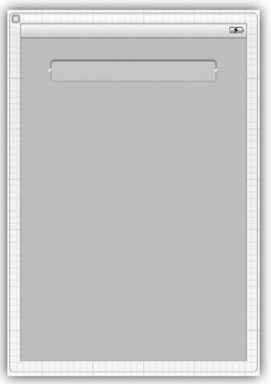

그림 11-2

이제 유저가 온도 값을 입력하는 텍스트 필드까지 만들었다. 다음 단계는 눌리면 변환을 시작하는 버튼 오브젝트를 추가하는 것이다. 이를 위해 Rounded Rect Button 오브젝트를 라이브러리에서 뷰로 끌어놓는다. 버튼 오브젝트를 더블 클릭하여 문자열을 입력할 수 있는 상태로 만들고 Convert라고 입력한다. 이제 버튼을 텍스트 필드 아래쪽으로 끌면 파란색 점선이 보일 것이다. 이를 이용하여 텍스트 필드의 가운데 쪽으로 위치시키고 마우스 버튼에서 손을 뗀다.

마지막으로 필요한 유저 오브젝트는 변환된 결과 값이 표시된 라벨이다. Label 오브젝트를 라이브러리에서 뷰로 끌어놓고 버튼의 아래쪽으로 위치시킨다. 라벨의 폭을 넓혀 대략 화면의 1/3 정도의 넓이가 되도록 하고 파란색 점선을 이용하여 화면의 가운데에 위치시킨다.

라벨을 더블 클릭하여 문자열을 선택한 후 백스페이스 키를 눌러 문자열을 모두 삭제한다(문자열은 변환계산의 결과 값이 뷰 컨트롤러 클래스의 메서드에 의해 설정된다). 문자열을 삭제하였기 때문에 이제 라벨이 있긴 하지만 화면에 보이지는 않을 것이다. 라벨이 위치하고 있는 부분을 클릭하면 크기변환 점선이 표시되어 라벨의 정확한 위치가 표시된다.

이제 프로젝트에서 사용할 유저 인터페이스의 디자인이 완료되었으며, 뷰는 그림 11-3과 같이 보일 것이다. 이제 테스트를 위해 빌드하고 실행할 수 있다.

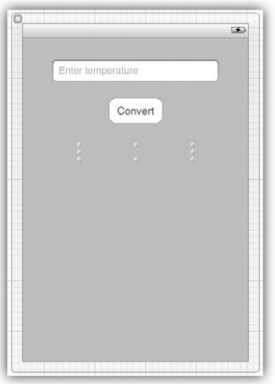

그림 11-3

11.3 예제 애플리케이션 빌드하고 실행하기

컨트롤러 코드를 작성하고 유저 인터페이스와 연결하기 전에 테스트 빌드를 만들고 애플리케이션을 실행해보도록 하자. 툴바의 Run 버튼을 클릭하여 컴파일을 하고 시뮬레이터에서 실행해보자. 인터페이스가 마음에 들지 않는다면 인터페이스 빌더로 다시 수정하면 된다. 시뮬레이터에서 보이는 유저 인터페이스가 마음에 든다면 이제 오브젝티브-C로 컨트롤러에 로직을 작성한다. iOS Simulator ➡ Quit iOS Simulator를 선택하여 시뮬레이터를 종료시킨다.

11.4 액션과 아웃렛 추가하기

유저가 온도 값을 입력하고 변환 버튼을 터치하면 온도를 변환하는 연산을 수행하는 액션을 호출해야 한다. 그 결과 값은 라벨 오브젝트를 통해 유저에게 보여질 것이다. 액션은 뷰 컨트롤러 클래스에 선언되고 구현되는 메서드의 형태가 된다. 뷰 컨트롤러 메서드에서 텍스트 필드나 라벨 오브젝트를 접근하는 것은 아웃렛(Outlet)을 통해 구현된다.

진행하기에 앞서 10장 "**아이폰 iOS 5 애플리케이션 개발 아키텍처 개요**"에서 배운 서브클래싱의 사용법에 대해 다시 한 번 알아보자. UIKit 프레임워크는 UIViewController라는 클래스를 가지고 있으며, 이는 일반적인 애플리케이션의 뷰 컨트롤러의 뼈대를 제공

한다. 그러나 실제 동작을 하는 애플리케이션을 개발하려면 우리의 애플리케이션에서 필요한 기능들을 이 보편적인 뷰 컨트롤러 클래스에 추가해야 한다. 이는 UIView Controller 클래스를 서브클래싱하고 필요한 기능들을 추가하는 것을 통해 이루어진다.

새로운 프로젝트를 만들 때 Xcode는 필요할 것이라 예상하고, UIViewController의 서브클래스를 만들고 UnitConverterViewController라고 이름 짓는다(프로젝트를 처음 만들 때 사용한 클래스 접두사를 사용한다). 이를 위해 Xcode는 두 개의 소스 파일을 생성하는데, 헤더 파일의 이름은 UnitConverterViewController.h이며 소스 코드 파일의 이름은 UnitConverterViewController.m이다.

Xcode의 프로젝트 내비게이터 화면에서 UnitConverterViewController.h 파일을 선택한 후 에디터 화면에서 내용을 확인하자.

```
#import <UIKit/UIKit.h>

@interface UnitConverterViewController : UIViewController {
}
@end
```

위의 코드에서 확인할 수 있듯이, UnitConverterViewController라는 새로운 클래스는 UIKit 프레임워크의 UIViewController 클래스의 서브클래스다.

다음 단계는 두 개의 아웃렛과 액션 메서드로 서브클래스를 확장시키는 것이다. 텍스트 필드와 라벨 오브젝트의 변수를 선언하고 IBOutlet 키워드를 사용하자.

```
#import <UIKit/UIKit.h>

@interface UnitConverterViewController : UIViewController {
    UITextField     *tempText;
    UILabel         *resultLabel;
}
@property (strong, nonatomic) IBOutlet UILabel *resultLabel;
@property (strong, nonatomic) IBOutlet UITextField *tempText;
@end
```

다음은 유저가 Convert 버튼을 터치했을 때 호출될 액션을 선언해야 한다. IBAction 키워드를 사용하여 선언한다.

```
#import <UIKit/UIKit.h>

@interface UnitConverterViewController : UIViewController {
    UITextField     *tempText;
    UILabel         *resultLabel;
}
@property (strong, nonatomic) IBOutlet UILabel *resultLabel;
@property (strong, nonatomic) IBOutlet UITextField *tempText;
- (IBAction)convertTemp:(id)sender;
@end
```

뷰 컨트롤러 클래스에 convertTemp:라는 이름의 메서드를 선언하였다. 메서드를 정의하였으면 구현 소스 파일에서 메서드를 구현해야 한다. 이를 위해 UnitConverterViewController.m 파일을 선택하여 편집창에 내용이 표시되도록 한다.

```
#import "UnitConverterViewController.h"

@implementation UnitConverterViewController

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // 더 이상 사용하지 않는 데이터, 이미지 등을 해제함
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];
}
- (void)viewDidUnload
{
    [super viewDidUnload];
```

```
    // 메인 뷰의 서브뷰를 해제함
    // e.g. self.myOutlet = nil;
}
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
}
- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
}
- (BOOL)shouldAutorotateToInterfaceOrientation:
(UIInterfaceOrientation)interfaceOrientation
{
    // 지원하는 방향이면 YES를 리턴함
    return (interfaceOrientation != UIInterfaceOrientationPortraitUpsideDown);
}
@end
```

우선 tempText와 resultLabel 오브젝트를 위한 통합된 접근자(synthesize accessor)를 생성하고 convertTemp 메서드를 구현한다. UnitConverterViewController.m 파일의 아래의 부분이 이에 해당한다.

```
#import "UnitConverterViewController.h"

@implementation UnitConverterViewController
@synthesize resultLabel, tempText;

- (void) convertTemp: (id) sender {
    double farenheit = [tempText.text doubleValue];
```

```
    double celsius = (farenheit - 32) / 1.8;

    NSString *resultString = [[NSString alloc]
       initWithFormat: @"Celsius %f", celsius];
    resultLabel.text = resultString;
}
```

더 진행하기 전에 위의 코드가 무슨 일을 하는지 알아보자. 만약 오브젝티브–C를 잘 안다면 다음 몇 문단은 건너뛰어도 된다.

이 파일에서는 .h 파일에서 정의된 convertTemp: 메서드를 구현하였다. 이 메서드는 sender라고 하는 한 개의 참조 인자를 받는다. sender는 이 메서드를 호출한 오브젝트다(이 경우는 버튼 오브젝트). 이번 예제에서는 이를 사용하지 않겠지만, 이를 이용하면 어떤 오브젝트에서 호출되었는지에 상관없는, 일반적인 목적의 메서드를 만들 수 있다. 예를 들면, Convert to Fahrenheit와 Convert to Celsius라는 두 개의 버튼을 만들었을 때 이 두 개의 버튼이 모두 같은 convertTemp: 메서드를 호출할 수 있다. 이 경우 메서드는 sender 오브젝트를 통하여 어떤 버튼이 메서드를 호출했는지를 확인하고 적절한 단위 변환을 수행할 수 있는 것이다.

다음은 유저가 입력할 실수 값을 저장하기 위해 실수형 변수를 선언하였다. 그리고 dot notation을 사용하여 UITextField 오브젝트의 text 속성(텍스트 필드에 표시되는 문자열을 저장함)을 접근한다. 이 속성의 데이터 타입은 NSString 오브젝트다. NSString 클래스는 doubleValue라는 인스턴스 메서드를 가지고 있고, 이를 통해 문자열을 실수 값으로 변환할 수 있다. 그러므로 입력받은 text 속성을 이 메서드를 통해 실수형으로 변환한 후 farenheit 변수에 저장한다.

이제 입력받은 숫자를 Celsius로 변환한 후 그 결과를 또 다른 변수인 celsius에 저장한다. 다음은 새로운 NSString 오브젝트를 만들고 Celsius라는 문자와 계산한 숫자를 조합한 결과로 NSString 오브젝트를 초기화한다. 이를 위해 새로운 오브젝트를 참조할 포인터를 선언하고 resultText라 이름 붙인다.

마지막으로 dot notation을 사용하여 새로운 문자열을 UILable 오브젝트의 text 속성에 할당하여 이 문자열이 유저에게 표시되도록 한다.

viewDidUnload 메서드를 수정하여 더 이상 필요하지 않은 변수들의 메모리를 사용하지 않음으로 설정한다.

```
- (void)viewDidUnload {
    // 메인 뷰의 서브뷰를 해제함
    // e.g. self.myOutlet = nil;
    self.resultLabel = nil;
    self.tempText = nil;
}
```

다음으로 진행하기 전에 빌드와 실행을 수행하여 코드에 에러가 없음을 확인하자. 툴바의 Run 버튼을 클릭하고 혹시 문법 에러나 경고가 있으면 수정한다.

11.5 액션과 아웃렛을 유저 인터페이스에 연결하기

이번 애플리케이션 개발의 마지막 단계는 뷰 컨트롤러 클래스의 액션과 아웃렛을 실제 유저 인터페이스 뷰의 오브젝트와 연결하는 것이다. 다행스럽게도 이 작업은 인터페이스 빌더를 통해 그래픽 환경에서 수행할 수 있다.

UnitViewController.xib 파일을 클릭하여 유저 인터페이스를 인터페이스 빌더로 읽어들인다. 이제 뷰 컨트롤러 클래스에 만든 IBOutlet 인스턴스 변수를 라벨과 문자열 오브젝트로 각각 연결시킨다. 유저 인터페이스가 표시된 화면 왼편에 세 개의 아이콘이 있는 좁은 화면이 보일 것이다. 화면 아랫부분에 흰색 화살표를 가진 회색 원을 클릭하면 화면이 확장되며 더 자세한 사항들이 표시된다.

그림 11-4

가장 윗부분의 아이콘은 File's Owner다. 이 아이콘은 애플리케이션이 실행될 때 NIB 파일에 있는 유저 인터페이스를 호출하는 클래스를 포함하는 파일을 의미한다. 이번 예제에서는 액션과 아웃렛을 선언한 UnitConverterViewController 클래스다.

resultLable IBOutlet 변수를 라벨 오브젝트에 연결하기 위해 키보드의 Ctrl 키를 누른 상태에서 File's Owner 아이콘을 클릭한 후 유저 인터페이스 디자인의 라벨 오브젝트 위로 끌어놓는다. 그림 11-5와 같이 파란색 줄이 아이콘에서부터 오브젝트까지 연결됨을 확인할 수 있다.

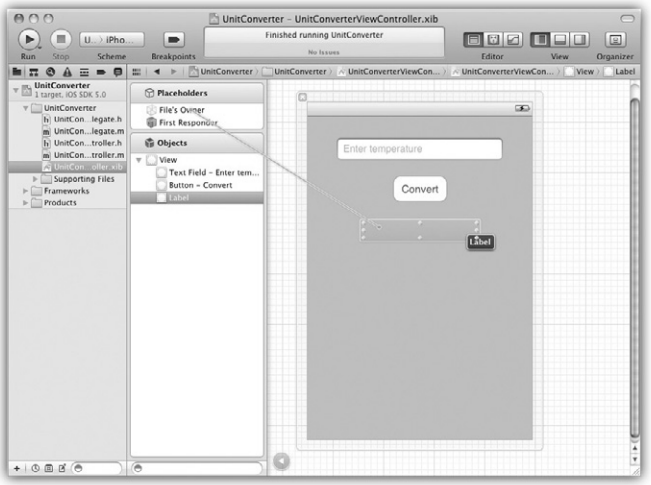

그림 11-5

마우스 버튼을 놓으면 인터페이스 빌더는 선택된 오브젝트의 형과 일치하는 IBOutlet 변수들을 표시해줄 것이다.

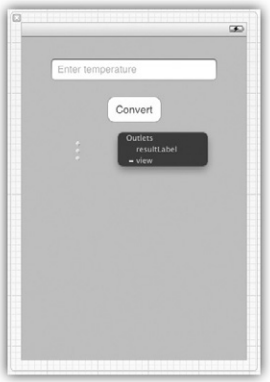

그림 11-6

resultLabel을 선택함으로써 연결을 완료한다. tempText 아웃렛을 텍스트 필드 오브젝트에 연결하기 위해 위의 과정을 반복한다. 만약 여러분이 만든 연결을 확인하고 싶으면, 오른쪽 화면의 윗부분의 툴바 제일 오른쪽을 선택하여 Connections Inspector를 실행시킨다. 또는 View ➞ Utilities ➞ Show Connections Inspector를 선택한다. 그림 11-7은 라벨 오브젝트에 대한 연결 정보를 보여주고 있다.

그림 11-7

마지막 단계는 버튼 오브젝트를 convertTemp 액션 메서드에 연결하는 것이다. 코코아 터치 오브젝트는 보통 다양한 이벤트를 지원한다. 오브젝트가 지원하는 모든 이벤트를 확인하고 싶으면, Connections Inspector에서 뷰 윈도우의 버튼 오브젝트를 선택한다. Connections 화면의 Sent Events 아랫부분의 항목들은 버튼 오브젝트에 의해 호출될 수 있는 이벤트들이다. 이번 예제에서는 Touch Up Inside가 처리해야 하는 이벤트다.

이 이벤트는 사용자가 버튼의 안쪽을 터치하고 손을 떼었을 때 발생한다. 이러한 이벤트가 발생했을 때 convertTemp 메서드를 호출하게 설정하자. 이 연결을 설정하기 위해 connections 화면의 Touch Up Inside 이벤트 옆의 작은 원을 클릭한 후 그림 11-8처럼 File's Owner 아이콘으로 드래그한다.

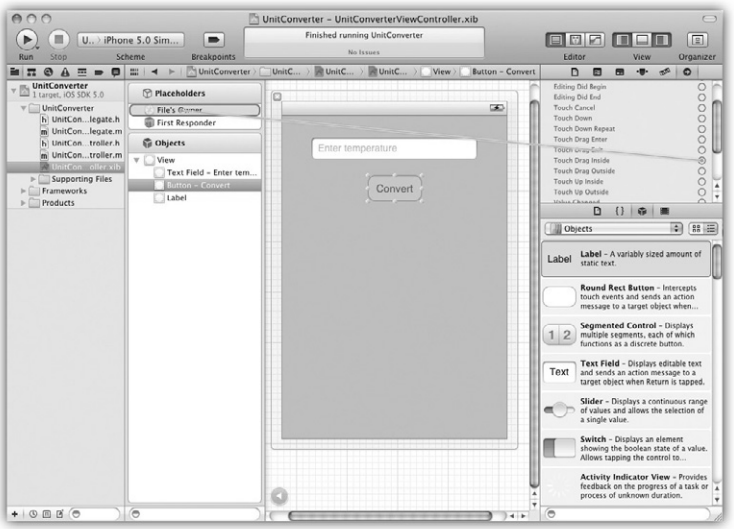

그림 11-8

File's Owner 위에서 마우스 버튼을 놓으면 뷰 컨트롤러에서 제공하는 메서드의 목록이 표시될 것이다. 이번 예제에서는 오직 convertTemp 메서드만 있으므로 이를 선택한다. Connections 화면의 이벤트 목록이 새로 만든 연결을 반영하여 표시될 것이다.

11.6 완성된 애플리케이션 빌드 및 실행하기

Xcode 프로젝트 윈도우에서 Run을 클릭하여 애플리케이션을 컴파일하고 시뮬레이터에서 실행시킨다. 애플리케이션이 실행되면 텍스트 필드를 클릭하고 화씨 온도 값을 입력한다. 이제 Convert 버튼을 클릭하면 입력한 온도에 해당하는 섭씨 온도가 표시된다. 모든 것이 예상한 대로 이루어졌다면 다음과 같은 화면이 표시될 것이다(만약 키보드가

결과 값을 가린다면, 인터페이스 빌더로 유저 인터페이스를 불러드린 후 라벨, 버튼 등을 화면의 윗부분으로 이동시킨다).

그림 11-9

11.7 요약

이번 장에서는 앞 장에서 배운 이론을 실습해보았다. 특히 뷰와 컨트롤러의 분리, 서브클래싱, 액션과 아웃렛을 통한 타깃-액션 패턴을 연습해보았다.

이번 예제 애플리케이션을 실행해보면 키보드가 한 번 표시된 후에는 리턴키를 누르거나 화면의 다른 부분을 선택해도 키보드가 사라지지 않는다는 것을 알 수 있을 것이다. 키보드가 사라지게 하기 위해서는 추가적인 코드가 필요하다. 다음 장인 12장 "**아이폰 키보드를 감추는 iOS 5 코드 만들기**"에서는 리턴키를 누르거나 화면을 터치했을 때 키보드를 사라지게 하는 방법에 대해 알아보자.