### **MOBILE OS - IOS AND ANDROID**

In The Anatomy of an iPhone 4 we looked at the hardware that is contained within an iPhone 4 device. When we develop apps for the iPhone Apple does not allow us direct access to any of this hardware. In fact, all hardware interaction takes place exclusively through a number of different layers of software that act as intermediaries between the application code and device hardware. These layers make up what is known as an operating system. In the case of the iPhone, this operating system is known as iOS.

In order to gain a better understanding of the iPhone development environment, this chapter will look in detail at the different layers that comprise the iOS operating system and the frameworks that allow us, as developers, to write iPhone applications.

### **IPHONE OS BECOMES IOS**

Prior to the release of the first iPad in 2010, the operating system running on the iPhone was referred to as iPhone OS. Given that the operating system used for the iPad is essentially the same as that on the iPhone it didn't make much sense to name it iPad OS. Instead, Apple decided to adopt a more generic and non-device specific name for the operating system.

Given Apple's predilection for names prefixed with the letter 'i' (iTunes, iBookstore, iMac etc) the logical choice was, of course, iOS. Unfortunately, iOS is also the name used by Cisco for the operating system on its routers (Apple, it seems, also has a predilection for ignoring trademarks). When performing an internet search for iOS, therefore, be prepared to see large numbers of results for Cisco's iOS which have absolutely nothing to do with Apple's iOS.

# An Overview of the iOS 4 Architecture

As previously mentioned, iOS consists of a number of different software layers, each of which provides programming frameworks for the development of applications that run on top of the underlying hardware. These operating system layers can be presented diagrammatically as illustrated in the following figure:

Cocoa (Application)
Media
Core Services
Address Book Core Data Core Foundation Foundation
Quick Look Social Security WebKit
Core OS
Kernel and Device Drivers

some diagrams designed to graphically depict the iOS software stack show an additional box positioned above the Cocoa Touch layer to indicate the applications running on the device.

In the above diagram we have not done so since this would suggest that the only interface available to the app is Cocoa Touch. In practice, an app can directly call down any of the layers of the stack to perform tasks on the physical device.

That said, however, each operating system layer provides an increasing level of abstraction away from the complexity of working with the hardware. As an iOS developer you should, therefore, always look for solutions to your programming goals in the frameworks located in the higher level iOS layers before resorting to writing code that reaches down to the lower level layers.

In general, the higher level of layer you program to, the less effort and fewer lines of code you will have to write to achieve your objective. And as any veteran programmer will tell you, the less code you have to write the less opportunity you have to introduce bugs.

Now that we have identified the various layers that comprise iOS 4 we can now look in more detail at the services provided by each layer and the corresponding frameworks that make those services available to us as application developers.

#### The Cocoa Touch Layer

The Cocoa Touch layer sits at the top of the iOS stack and contains the frameworks that are most commonly used by iPhone application developers. Cocoa Touch is primarily written in Objective-C, is based on the standard Mac

OS X Cocoa API (as found on Apple desktop and laptop computers) and has been extended and modified to meet the needs of the iPhone. The Cocoa Touch layer provides the following frameworks for iPhone app development:

#### UIKit Framework (UIKit.framework)

The UIKit framework is a vast and feature rich Objective-C based programming interface. It is, without question, the framework with which you will spend most of your time working. Entire books could, and probably will, be written about the UIKit framework alone. Some of the key features of UIKit are as follows:

- User interface creation and management (text fields, buttons, labels, colors, fonts etc) Application lifecycle management
- Application event handling (e.g. touch screen user interaction) Multitasking
- Wireless Printing
- Data protection via encryption
- Cut, copy, and paste functionality
- Web and text content presentation and management
- Data handling
- Inter-application integration
- Push notification in conjunction with Push Notification Service
- Local notifications (a mechanism whereby an application running in the background can gain the user's attention)
- Accessibility
- Accelerometer, battery, proximity sensor, camera and photo library interaction.
  Touch screen gesture recognition
- File sharing (the ability to make application files stored on the device available via iTunes) Blue tooth based peer to peer connectivity between devices
- Connection to external displays

### Map Kit Framework (MapKit.framework)

If you have spent any appreciable time with an iPhone then the chances are you have needed to use the Maps application more than once, either to get a map of a specific area or to generate driving directions to get you to your intended destination.

The Map Kit framework provides a programming interface that enables you to build map based capabilities into your own applications. This allows you to, amongst other things, display scrollable maps for any location, display the map corresponding to the current geographical location of the device and annotate the map in a variety of ways.

## Push Notification Service

The Push Notification Service allows applications to notify users of an event even when the application is not currently running on the device. Since the introduction of this service it has most commonly been used by news based applications.

Typically when there is breaking news the service will generate a message on the device with the news headline and provide the user the option to load the corresponding news app to read more details. This alert is typically accompanied by an audio alert and vibration of the device. This feature should be used sparingly to avoid annoying the user with frequent interruptions.

## Message UI Framework (MessageUI.framework)

The Message UI framework provides everything you need to allow users to compose and send email messages from within your application. In fact, the framework even provides the user interface elements through which the user enters the email addressing information and message content. Alternatively, this information can be predefined within your application and then displayed for the user to edit and approve prior to sending.

## Address Book UI Framework (AddressUI.framework)

Given that a key function of the iPhone is as a communications device and digital assistant it should not come as too much of a surprise that an entire framework is dedicated to the integration of the address book data into your own applications. The primary purpose of the framework is to enable you to access, display, edit and enter contact information from the iPhone address book from within your own application.

# Game Kit Framework (GameKit.framework)

The Game Kit framework provides peer-to-peer connectivity and voice communication between multiple devices and users allowing those running the same app to interact. When this feature was first introduced it was anticipated by Apple that it would primarily be used in multi-player games (hence the choice of name) but the possible applications for this feature clearly extend far beyond games development.

# iAd Framework (iAd.framework)

The purpose of the iAd Framework is to allow developers to include banner advertising within their applications. All advertisements are served by Apple's own ad service.

#### **Event Kit UI Framework**

The Event Kit UI framework was introduced in iOS 4 and is provided to allow the calendar events to be accessed and edited from within an application.

