

PART I

Getting Started

- ▶ **CHAPTER 1:** Getting Started with iPad Programming
- ▶ **CHAPTER 2:** Write Your First Hello World! Application
- ▶ **CHAPTER 3:** Views, Outlets, and Actions
- ▶ **CHAPTER 4:** View Controllers
- ▶ **CHAPTER 5:** Keyboard Inputs
- ▶ **CHAPTER 6:** Screen Rotations

1

Getting Started with iPad Programming

WHAT YOU WILL LEARN IN THIS CHAPTER:

- How to obtain the iPhone SDK
- The components included in the iPhone SDK
- The features of the development tools — Xcode, Interface Builder, iPhone Simulator
- The capabilities of the iPhone Simulator
- The architecture of the iPhone OS
- The frameworks of the iPhone SDK
- The limitations and characteristics of the iPad

Welcome to the world of iPad programming! That you are now holding this book shows that you are fascinated with the idea of developing iPad applications and want to join the ranks of those tens of thousands of developers whose applications are already deployed in the AppStore.

As the old Chinese adage says, “To accomplish your mission, first sharpen your tools.” Successful programming requires you first of all to know your tools well. Indeed, this couldn’t be more true for iPad programming — you need to know quite a few tools before you can even get started. Hence, the goal of this chapter is to show you the various relevant tools and information you need to jump on the iPad development bandwagon.

Without further ado, it’s time to get down to work.

OBTAINING THE IPHONE SDK

To develop for the iPad, you first need to sign up as a Registered iPhone Developer at <http://developer.apple.com/iphone/program/start/register/>. The registration is free and provides you with access to the iPhone SDK (software development kit) and other resources that are useful for getting started.



NOTE The iPad uses the same operating system (OS) as the iPhone and iPod touch. So, although you are developing for the iPad, you will still use the iPhone SDK. Hence, throughout this book you will see the term “iPhone” used very often.

After signing up, you can download the iPhone SDK (version 3.2; see Figure 1-1).

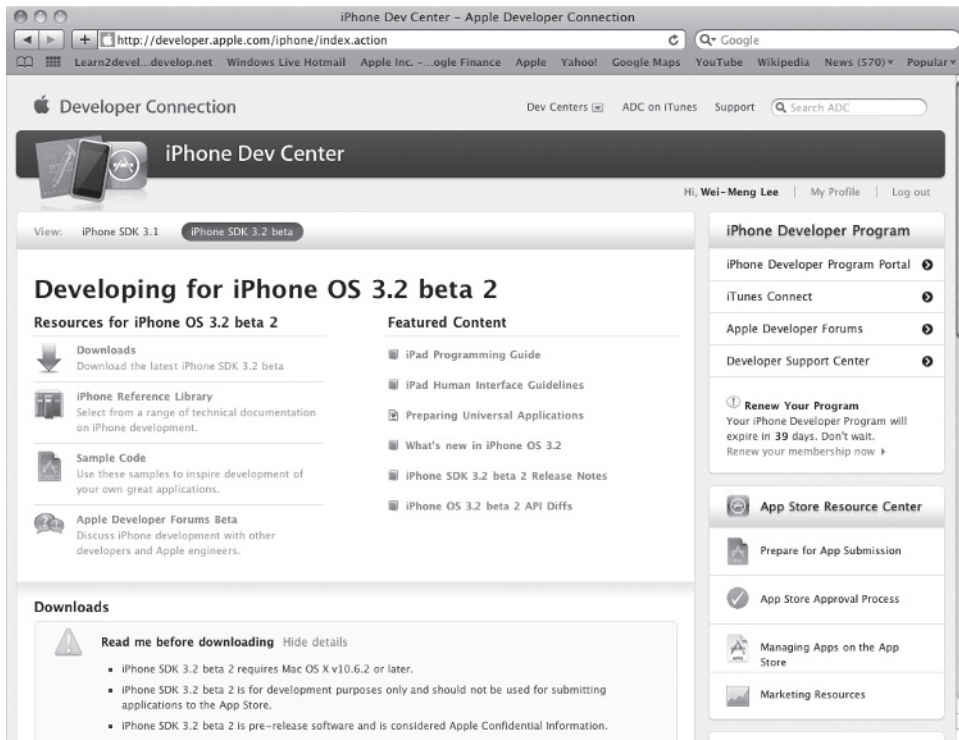


FIGURE 1-1

Before you install the iPhone SDK, make sure you satisfy the following system requirements:

- Only Intel Macs are supported, so if you have another processor type (such as the older G4 or G5 Macs), you're out of luck.
- You have updated your system with the latest Mac OS X release.

An actual iPad is highly recommended, although not strictly necessary. To test your application, you can use the included iPhone Simulator (which allows you to simulate an iPhone or an iPad). However, to test certain hardware features like GPS, the Accelerometer, and such, you need to use a real device.

When the SDK is downloaded, proceed with installing it (see Figure 1-2). Accept a few licensing agreements and then select the destination folder in which to install the SDK.



FIGURE 1-2

If you select the default settings during the installation phase, the various tools will be installed in the `/Developer/Applications` folder (see Figure 1-3).



FIGURE 1-3

COMPONENTS OF THE IPHONE SDK

The iPhone SDK includes a suite of development tools to help you develop applications for your iPhone, iPod touch, and iPad. It includes:

- Xcode — Integrated development environment (IDE) that enables you to manage, edit, and debug your projects.
- Dashcode — Integrated development environment (IDE) that allows you to develop web-based iPhone and iPad applications and Dashboard Widgets. Dashcode is beyond the scope of this book.
- iPhone Simulator — Provides a software simulator to simulate an iPhone or an iPad on your Mac.
- Interface Builder — Visual editor for designing your user interfaces for your iPhone and iPad applications.
- Instruments — Analysis tool to help you optimize your application and monitor for memory leaks in real-time.

The following sections discuss each tool (except Dashcode) in more detail.

Xcode

To launch Xcode, double-click the Xcode icon (located in the `/Developer/Applications` folder; refer to Figure 1-3). Alternatively, go the quicker route and use Spotlight: simply type **Xcode** into the search box and Xcode should be in the Top Hit position.

Figure 1-4 shows the Xcode Welcome screen.



FIGURE 1-4

Using Xcode, you can develop different types of iPhone, iPad, and Mac OS X applications using the various project templates shown in Figure 1-5.

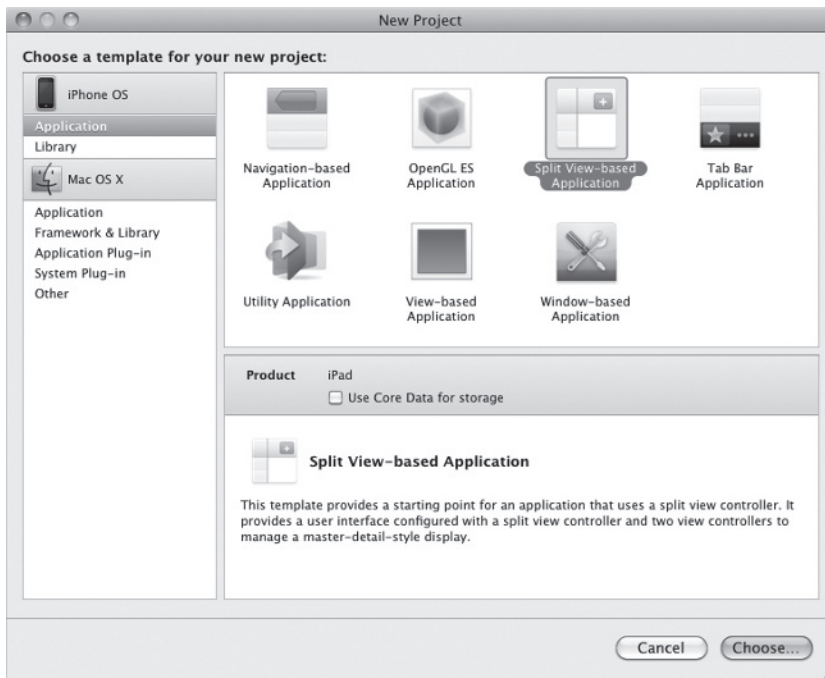


FIGURE 1-5

The IDE in Xcode provides many tools and features that make your development life much easier. One such feature is Code Completion (see Figure 1-6), which displays a pop-up list showing the available classes and members (such as methods, properties, and so on).



NOTE For a more comprehensive description of some of the most commonly used features in Xcode, refer to Appendix B.

iPhone Simulator

The iPhone Simulator (see Figure 1-7) is a very useful tool that you can use to test your application without using your actual iPhone/iPod touch/iPad. The iPhone Simulator is located in the `/Developer/Platforms/iPhoneSimulator.platform/Developer/Applications` folder. Most of the time, you don't need to launch the iPhone Simulator directly — running (or debugging) your application in Xcode automatically brings up the iPhone Simulator. Xcode installs the application on the iPhone Simulator automatically.

The iPhone Simulator can simulate different versions of the iPhone OS (see Figure 1-8). This capability is useful if you need to support older versions of the platform as well as testing and debugging errors reported in the application on specific versions of the OS.



FIGURE 1-7

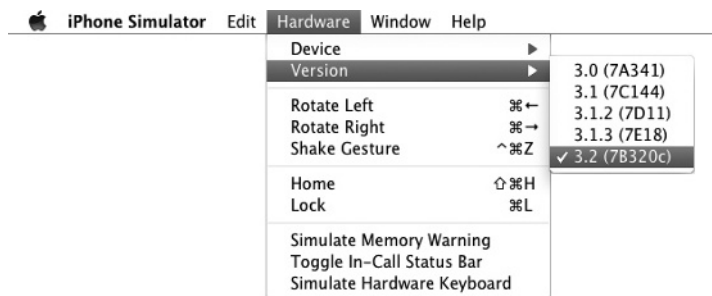


FIGURE 1-8

Features of the iPhone Simulator

The iPhone Simulator simulates various features of a real iPhone, iPod touch, or iPad device. Features you can test on the iPhone Simulator include:

- Screen rotation — left, top, and right
- Support for gestures:
 - Tap
 - Touch and Hold
 - Double Tap
 - Swipe
 - Flick
 - Drag
 - Pinch
- Low-memory warning simulations

However, the iPhone Simulator, being a software simulator for the real device, does have its limitations. Features not available on the iPhone Simulator include:

- Making phone calls
- Accessing the Accelerometer
- Sending and receiving SMS messages
- Installing applications from the App Store
- Camera
- Microphone
- Several features of OpenGL ES



NOTE *In the previous version of the SDK (3.1.3), the iPhone Simulator supports location data by always returning a fixed coordinate, such as Latitude 37.3317 North and Longitude 122.0307 West. In the latest release of the SDK, the iPhone Simulator uses the location data of the Mac it is currently running on and returns the current location.*

It is worth noting that the speed of the iPhone Simulator is more tightly coupled to the performance of your Mac instead of the actual device. Therefore, it is important that you test your application on a real device rather than rely exclusively on the iPhone Simulator for testing.

Although you have limitations with the iPhone Simulator, it is definitely a useful tool to get your applications tested. That said, testing your application on a real iPad is imperative before you deploy it on the AppStore.

Uninstalling Applications from the iPhone Simulator

The user domain of the iPhone OS file system for the iPhone Simulator is stored in the `~/Library/Application Support/iPhone Simulator/` folder.



NOTE The `~/Library/Application Support/iPhone Simulator/` folder is also known as the `<iPhoneUserDomain>`.

All third-party applications are stored in the `<iPhoneUserDomain>/<version_no>/Applications/` folder. When an application is deployed onto the iPhone Simulator, an icon is created on the Home screen (shown on the left in Figure 1-9) and a file and a folder are created within the `Applications` folder (shown on the right in Figure 1-9).

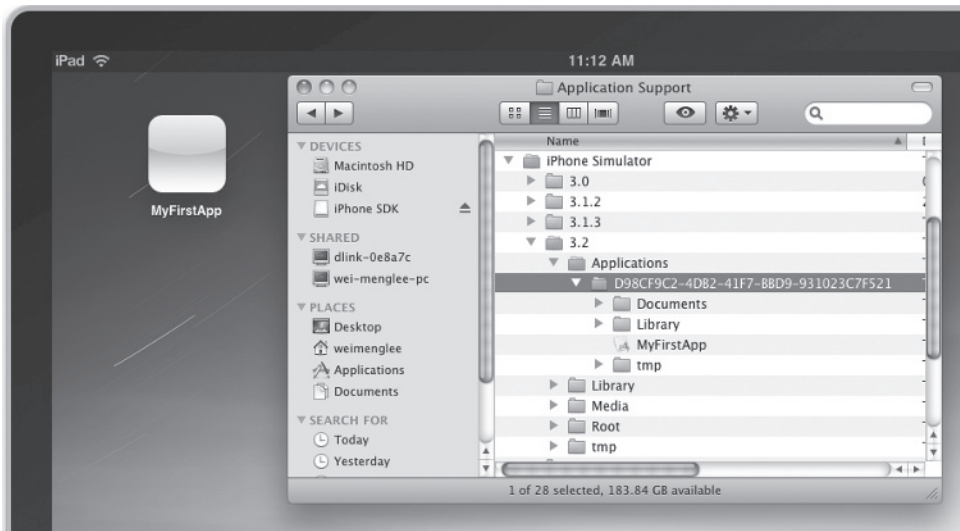


FIGURE 1-9

To uninstall (delete) an application, execute the following steps:

1. Click and hold the icon of the application on the Home screen until all the icons start wriggling. Observe that all the icons now have an X button displayed on their top left corners.
2. Click the X button (see Figure 1-10) next to the icon of the application you want to uninstall.

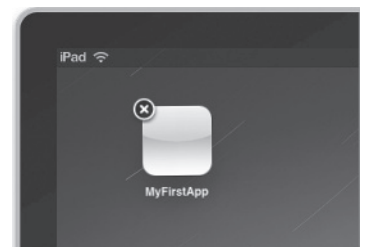


FIGURE 1-10

3. An alert window appears asking if you are sure you want to delete the icon. Click Delete to confirm the deletion.



WARNING When the application is uninstalled, the corresponding files and folders in the Applications folder are deleted automatically.

The easiest way to reset the iPhone Simulator to its original state is to select iPhone Simulator ⇨ Reset Content and Settings.

Interface Builder

Interface Builder is a visual tool that allows you to design your user interfaces for your iPad applications. Using Interface Builder, you drag and drop views on windows and then connect the various views with outlets and actions so that they can programmatically interact with your code.



NOTE Outlets and actions are discussed in more detail in Chapter 3 and Appendix C discusses Interface Builder in more detail.

Figure 1-11 shows the various windows in Interface Builder.

Instruments

The Instruments (see Figure 1-12) application allows you to dynamically trace and profile the performance of your Mac OS X, iPhone, and iPad applications.

Using Instruments, you can:

- Stress test your applications.
- Monitor your applications for memory leaks.
- Gain a deep understanding of the executing behavior of your applications.
- Track difficult-to-reproduce problems in your applications.



NOTE Covering the Instruments application is beyond the scope of this book. For more information, refer to Apple's documentation.

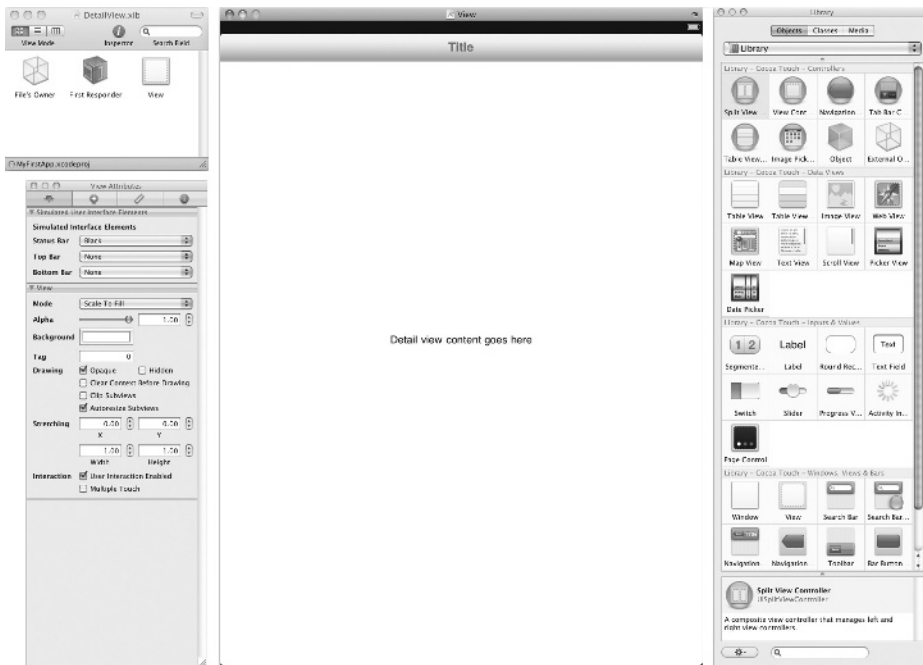


FIGURE 1-11



FIGURE 1-12

ARCHITECTURE OF THE IPHONE OS

Although this book doesn't explore the innards of the iPhone OS, understanding some of the important points of the iPhone OS is useful. Figure 1-13 shows the different abstraction layers that make up the Mac OS X and the iPhone OS (which is used by the iPhone, iPod touch, and iPad).



NOTE *The iPhone OS is architecturally very similar to the Mac OS X except that the topmost layer is Cocoa Touch for iPhone instead of the Cocoa Framework.*



FIGURE 1-13

The bottom layer is the Core OS, which is the foundation of the operating system. It is in charge of memory management, the file system, networking, and other OS tasks, and it interacts directly with the hardware. The Core OS layer consists of components such as:

- OS X Kernel
- Mach 3.0
- BSD
- Sockets
- Security
- Power Management
- Keychain
- Certificates
- File System
- Bonjour

The Core Services layer provides an abstraction over the services provided in the Core OS layer. It provides fundamental access to iPhone OS services and consists of the following components:

- Collections
- Address Book
- Networking
- File Access
- SQLite
- Core Location
- Net Services
- Threading
- Preferences
- URL Utilities

The Media layer provides multimedia services that you can use in your iPad applications. It consists of the following components:

- Core Audio
- OpenGL
- Audio Mixing
- Audio Recording
- Video Playback
- JPG, PNG, TIFF
- PDF
- Quartz
- Core Animation
- OpenGL ES

The Cocoa Touch layer provides an abstraction layer to expose the various libraries for programming the iPad, such as:

- Multi-Touch events
- Multi-Touch controls
- Accelerometer
- View Hierarchy
- Localization
- Alerts

- Web Views
- People Picker
- Image Picker
- Controllers

The iPhone SDK consists of the frameworks shown in Table 1-1.



NOTE A framework is a software library that provides specific functionalities.

TABLE 1-1: The Frameworks in the iPhone SDK

FRAMEWORK NAME	DESCRIPTION
AddressBook.framework	Provides access to the centralized database for storing a user's contacts.
AddressBookUI.framework	Provides the UI to display the contacts stored in the Address Book database.
AudioToolbox.framework	Provides low-level C APIs for audio recording and playback, as well as managing the audio hardware.
AudioUnit.framework	Provides the interface for iPhone OS-supplied audio processing plug-ins in your application.
AVFoundation.framework	Provides low-level C APIs for audio recording and playback, as well as for managing the audio hardware.
CFNetwork.framework	Provides access to network services and configurations, such as HTTP, FTP, and Bonjour services.
CoreAudio.framework	Declares data types and constants used by other Core Audio interfaces.
CoreData.framework	Provides a generalized solution for object graph management in your application.
CoreFoundation.framework	Provides abstraction for common data types, Unicode strings, XML, URL resource, and so on.
CoreGraphics.framework	Provides C-based APIs for 2D rendering; based on the Quartz drawing engine.
CoreLocation.framework	Provides location-based information using a combination of GPS, cell ID, and Wi-Fi networks.
ExternalAccessory.framework	Provides a way to communicate with accessories.

FRAMEWORK NAME	DESCRIPTION
Foundation.framework	Provides the foundation classes for Objective C, such as NSObject, basic data types, operating system services, and so on.
GameKit.framework	Provides networking capabilities to games; commonly used for peer-to-peer connectivity and in-game voice features.
IOKit.framework	Provides capabilities for driver development.
MapKit.framework	Provides an embedded map interface for your application.
MediaPlayer.framework	Provides facilities for playing movies and audio files.
MessageUI.framework	Provides a view-controller–based interface for composing e-mail messages.
MobileCoreServices.framework	Provides access to standard types and constants.
OpenAL.framework	Provides an implementation of the OpenAL specification.
OpenGLES.framework	Provides a compact and efficient subset of the OpenGL API for 2D and 3D drawing.
QuartzCore.framework	Provides ability to configure animations and effects and then render those effects in hardware.
Security.framework	Provides the ability to secure your data and control access to software.
StoreKit.framework	Provides in-app purchase support for applications.
SystemConfiguration.framework	Provides the ability to determine network availability and state on device.
UIKit.framework	Provides the fundamental objects for managing an application’s UI.

SOME USEFUL INFORMATION BEFORE YOU GET STARTED

You now have a good idea of the tools involved in iPad application development. Before you go ahead and take the plunge, the following sections discuss some useful information that can make your journey more pleasant.

Versions of iPhone OS

At the time of writing, the iPhone OS is in its third revision — that is, version 3.2. Its major versions are as follows:

- 1.0 — Initial release of iPhone
- 1.1 — Additional features and bug fixes for 1.0

- 2.0 — Released with iPhone 3G; comes with App Store
- 2.1 — Additional features and bug fixes for 2.0
- 2.2 — Additional features and bug fixes for 2.1
- 3.0 — Third major release of the iPhone OS
- 3.1 — Additional features and bug fixes for 3.0
- 3.2 — This version release for iPad only; see the sidebar for what is new in iPhone OS 3.2

For a detailed description of the features in each release, check out http://en.wikipedia.org/wiki/iPhone_OS_version_history.

WHAT'S NEW IN IPHONE OS 3.2

In January 2010, Apple announced a new device based on the existing iPhone OS — the iPad. The iPad is a tablet computer that resembles an iPod touch, except that it has a much bigger screen. The iPad comes in six different editions, three with Wi-Fi and another three with Wi-Fi and 3G networks. The iPad will be released in three storage configurations: 16GB, 32GB, and 64GB.

Some of the important new features of iPhone OS 3.2 include:

- Support for existing iPhone applications by running them either in their original screen size or in pixel-doubled mode.
- New UI features such as Popovers, Split Views, and Custom Input Views.
- Support for external displays.
- Support for gestures detection using Gestures Recognizers.
- Improved Text support to allow applications to have more sophisticated text-handling capabilities.
- New File and Document support to facilitate building document-centric applications.

Testing on Real Devices

One of the most common complaints that beginning iPad programmers made was about the inability to test iPad applications they have developed on their actual devices. It seems odd that as the owner of the device, they can't even test their applications on it. Turns out that for security reasons, Apple requires all applications to be signed with a valid certificate, and for testing purposes, a developer certificate is required.

To test your applications on a device, you must sign up for the iPhone Developer program and request that a developer certificate be installed onto your device.

Screen Resolution

The iPad is a beautiful device with a high-resolution screen. At 9.7 inches (diagonally), the iPad screen supports multi-touch operation and allows a pixel resolution of 1024 x 768 at 132 ppi (see Figure 1-14). When designing your application, note that because of the status bar, the actual resolution is generally limited to 1004 x 768 pixels. Of course, you can turn off the status bar programmatically and gain access to the full 1024 x 768 resolution.

Also, be mindful that users may rotate the device to display your application in Landscape mode. You need to make provisions to your user interface so that the applications can still work properly in Landscape mode.



FIGURE 1-14



NOTE Chapter 6 discusses how to handle screen rotations. Unlike developing on the iPhone, applications running on the iPad must support different screen orientations.

Single-Window Applications

If you are new to mobile programming, be aware that the limited screen real estate means that most mobile platforms support only single-window applications — that is, your application window occupies the entire screen. The iPad is no exception to this platform limitation. Overlapping windows that are so common in desktop operating systems (such as Mac OS X and Windows) are not supported on the iPad.

No Background Applications

One of the major challenges in programming mobile devices is power management. A badly written application can be a resource hog and will drain the battery of the device very quickly. Apple acknowledges this issue, and from reviewing experiences obtained from other platforms, decided that major culprits in hurting battery life and performance are background applications. For example, on platforms such as Windows Mobile, when an application remains in memory when it goes out of view (because of an incoming call, for instance), each background application retained in memory continues to take its toll on the device's performance and battery life.

Apple's solution to this problem is simple: Disallow applications to run in the background. Although this is an effective solution, it has irked a lot of developers. Many useful applications require background operation to function correctly. For example, a chatting application needs to be running to receive messages from other users. To overcome this limitation, Apple has developed its Push Notification Service, which feeds applications with data even when they are not running. This service was released with iPhone 3.0. Using push technology, a device is constantly connected to Apple's server through an IP connection. When a device needs attention, a notification is sent from Apple's server to the device, thereby alerting the specific application that needs to service that notification.



NOTE Push notification is discussed in Chapter 17, "Apple Push Notification Services."

SUMMARY

This chapter offered a quick tour of the tools used for iPhone application development. You also learned some of the characteristics of iPad, such as the one-application limit and the support for existing iPhone applications. In the next chapter, you develop your first iPad application, and you will soon be on your way to iPad nirvana!

► WHAT YOU LEARNED IN THIS CHAPTER

TOPIC	KEY CONCEPTS
Obtaining the iPhone SDK	Register as an iPhone Developer at http://developer.apple.com first and download the free SDK.
iPhone Simulator	Most of the testing can be done on the iPhone Simulator. However, it is strongly recommended that you have a real device for actual testing.
Limitations of the iPhone Simulator	Access to hardware is generally not supported by the simulator. For example, the camera, Accelerometer, voice recording, and so on are not supported.
Frameworks in the iPhone SDK	The iPhone SDK provides several frameworks that perform specific functionalities on the iPad. You program your iPad applications using all these frameworks.
Background applications	The iPad does not support third-party background applications.
Screen resolution	1024 x 768 pixels (with status bar hidden). 1004 x 768 pixels (with status bar visible).
Single-window applications	All iPad applications are single-windowed — that is, all windows fill the entire screen and overlapping windows are not allowed.
