iPhone OSテクノロジーの概要

2009-05-27

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

序章 はじめに 7

対象読者 7
この書類の構成 8
iPhone SDKの入手方法 8
フィードバックの提供 8
関連項目 9

第 1 章 iPhone OSの開発 11

iPhone OSのアーキテクチャ 11
iPhone SDKの構成要素 12
作成可能なアプリケーション 13
リファレンスライブラリの使いかた 14

第 2 章 iPhone OSテクノロジー 17

Cocoa Touchレイヤ 17
Apple Push Notificationサービス 18
Address Book UIフレームワーク 18
アプリケーションでの電子メール 18
Map Kitフレームワーク 18
ピアツーピアサポート 19
UIKitフレームワーク 19
Mediaレイヤ 20
グラフィックステクノロジー 20
オーディオテクノロジー 21
ビデオテクノロジー 23
Core Servicesレイヤ 23
Address Book 23
Core Data 23
Core Foundation 24
Core Location 25
Foundationフレームワーク 25
In App Purchase 26
SQLite 26
XMLサポート 26
Core OSレイヤ 27
CFNetwork 27
アクセサリのサポート 27
セキュリティ 27
System 28

第 3 章 Cocoaからの移行 29

移行の際の一般的な注意 29
データモデルの移行 29
ユーザインターフェイスの移行 30
メモリ管理 30
フレームワークの違い 31
UIKitとAppKit 31
Foundationフレームワークの違い 33
その他のフレームワークの変更 34

付録 A iPhone OSのデベロッパツール 37

Xcode 37
Interface Builder 39
Instruments 40
Shark 41

付録 B iPhone OSのフレームワーク 43

デバイスのフレームワーク 43
シミュレータのフレームワーク 45
システムライブラリ 45

改訂履歴 書類の改訂履歴 47

図、表

第 1 章 **iPhone OSの開発　11**

図 1-1　iPhone OSの最上位層にあるアプリケーション　12
図 1-2　iPhone Reference Library　15
図 1-3　Xcodeリサーチアシスタント(Research Assistant)　16

第 2 章 **iPhone OSテクノロジー　17**

図 2-1　iPhone OSのレイヤ　17
表 2-1　Core Audioフレームワーク　22

第 3 章 **Cocoaからの移行　29**

表 3-1　インターフェイステクノロジーの違い　31
表 3-2　iPhone OSでは利用できないFoundationテクノロジー　34
表 3-3　iPhone OSとMac OS Xに共通のフレームワークにおける違い　34

付録 A **iPhone OSのデベロッパツール　37**

図 A-1　Xcodeのプロジェクトウインドウ　38
図 A-2　Xcodeからのプロジェクトの実行　39
図 A-3　Interface Builderを使用したiPhoneインターフェイスの作成　40
図 A-4　Instrumentsを使ったアプリケーションのチューニング　41

付録 B **iPhone OSのフレームワーク　43**

表 B-1　デバイスのフレームワーク　43

はじめに

iPhone OSは、iPhoneおよびiPod touchデバイスの中核となるオペレーティングシステムです。

iPhone OSプラットフォームは、Mac OS Xの開発で培われた知識を使用して開発されました。このため、iPhone OSプラットフォームでの開発に使われるツールおよびテクノロジーの多くは、Mac OS Xをルーツにしています。iPhone OSはMac OS Xに似ていますが、iPhone OS向けのアプリケーションを開発するためには、必ずしも経験を積んだMac OS Xデベロッパである必要はありません。iPhoneアプリケーションの開発を始めるために必要なすべてのものは、iPhone SDK (Software Development Kit)によって提供されます。

対象読者

『*iPhone OSテクノロジーの概要*』は、iPhone OSプラットフォーム初心者のための入門書です。本書は、開発プロセスに影響を与えるテクノロジーおよびツールの概要について説明し、関連文書やその他の情報源へのリンクを提供します。本書は、次の目的のために参照してください。

- iPhone OSプラットフォームを理解する。
- iPhone OSソフトウェアの各テクノロジーについて、使用する理由、およびどのような場合に使用するかを学ぶ。
- iPhone OSプラットフォーム用の開発のチャンスについて学ぶ。

- ほかのプラットフォームからiPhone OSへの移植方法についてのヒントおよびガイドラインを学ぶ。
- 関心を持ったテクノロジーに関連する主要な文書を見つける。

この文書には、ユーザレベルのシステム機能についての情報や、ソフトウェア開発プロセスに関係のない機能についての情報は含まれていません。

経験の浅いデベロッパの場合は、iPhone OSに慣れるためにこの文書が役立ちます。熟練デベロッパの場合は、特定のテクノロジーや開発手法を探究するためのロードマップとしてこの文書を使用できます。

この書類の構成

この文書は次の章と付録で構成されています。

- 「iPhone OSの開発」（11 ページ）では、iPhone OSと、iPhone SDKを使用したiPhone OS向けアプリケーションの開発の概要について説明します。
- 「iPhone OSテクノロジー」（17 ページ）では、iPhone OSのテクノロジーレイヤと、それがアプリケーションに提供する機能について見ていきます。
- 「Cocoaからの移行」（29 ページ）では、既存のCocoaアプリケーションをiPhone OSに移植するデベロッパへの初歩的なアドバイスを提供します。
- 「iPhone OSのフレームワーク」（43 ページ）では、ソフトウェア開発に使用できるフレームワークについて説明します。この情報を使用して、特定のテクノロジーを見つけたり、対象フレームワークがいつiPhone OSに導入されたかを調べることができます。
- 「iPhone OSのデベロッパツール」（37 ページ）では、iPhone OS用のソフトウェアを作成するために利用できるアプリケーションの概要を説明します。

iPhone SDKの入手方法

iPhone SDKには、iPhone OS用ソフトウェアを設計、作成、デバッグ、および最適化するために必要なツールが含まれています。また、iPhone OSプラットフォームのテクノロジー向けのヘッダファイル、サンプルコード、およびドキュメントも含まれています。iPhone SDKは、iPhone Dev Center (http://developer.apple.com/iphone)のメンバ領域からダウンロードできます。メンバ登録が必要ですが、登録は無料です。

Mac OS Xおよびそのテクノロジーで作業をするために利用できるツールの詳細については、「iPhone OSのデベロッパツール」（37 ページ）を参照してください。

フィードバックの提供

ドキュメントに関するフィードバックは、各ページの一番下にある組み込みのフィードバックフォームを使って送ってください。

Appleのソフトウェアやドキュメントのバグを発見した場合は、Appleにお知らせください。また、製品やドキュメントの今後の改訂版に期待する機能についての機能拡張リクエストを提出することもできます。バグ報告や機能拡張リクエストを提出するには、次に示すURLにあるADC WebサイトのBug Reportingページを使用します。

http://developer.apple.com/jp/bugreporter/

バグ報告を行うには、ADCの有効なログイン名とパスワードが必要です。ログイン名は、Bug Reportingページに説明されている手順に従って無料で入手できます。

関連項目

以下の文書は、iPhone開発に関連する重要な情報を提供します。

- 『*Cocoa Fundamentals Guide*』では、iPhoneアプリケーションの開発に使用するデザインパターンとデザインプラクティスに関する基本的な情報を提供しています。
- 『*iPhone Application Programming Guide*』では、iPhoneアプリケーションのアーキテクチャの概要と、アプリケーションの作成方法に関する実践的なガイダンスを提供しています。
- 『*iPhone Human Interface Guidelines*』では、iPhoneアプリケーションのユーザインターフェイスの設計方法についてのガイダンスと重要な情報を提供しています。
- 『*iPhone Development Guide*』では、iPhoneの開発プロセスに関する重要な情報をツールの視点から提供しています。この文書は、ソフトウェアをビルド、実行、およびテストするための、デバイスの構成とXcode（およびその他のツール）の使用についても言及しています。
- 『*The Objective-C 2.0 Programming Language*』では、Objective-Cと、iPhone OSのほとんどの動的な動作と拡張性の基礎となるObjective-Cランタイムシステムについて解説しています。

iPhone OSの開発

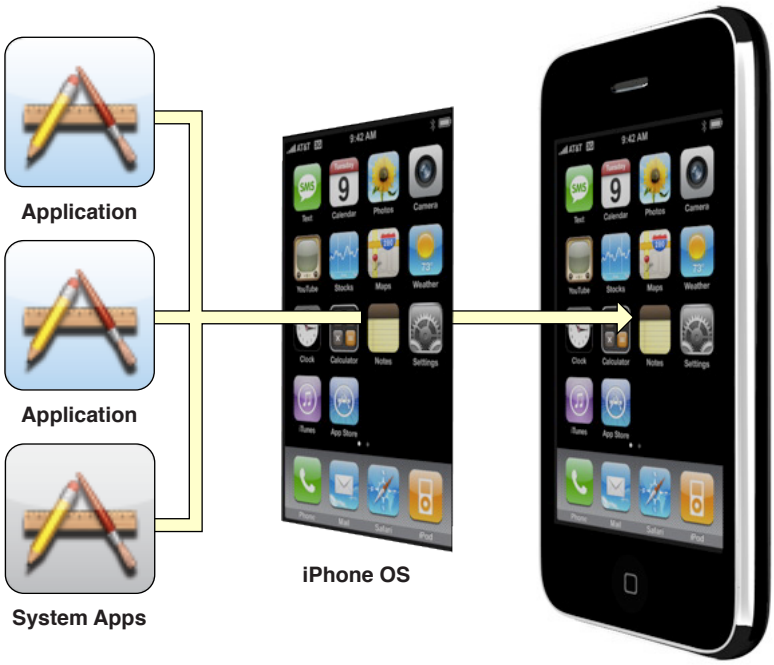
iPhone OSは、iPhoneおよびiPod touchデバイス上で稼働するオペレーティングシステムです。このオペレーティングシステムは、デバイスのハードウェアを管理するとともに、iPhone上でネイティブアプリケーションを実装するために必要な基本テクノロジーを提供します。iPhoneまたはiPod touchのどちらにインストールされているかに応じて、このオペレーティングシステムには、「電話(Phone)」、「メール(Mail)」、「Safari」などの標準的なシステムサービスをユーザに提供するいくつかのシステムアプリケーションが付属します。

iPhone SDKには、カスタムのネイティブアプリケーションを開発、インストール、および実行するために必要なツールとインターフェイスが含まれています。ネイティブアプリケーションは、iPhone OSのシステムフレームワークとObjective-C言語を使用して作成され、iPhone OS上で直接実行されます。Webアプリケーションとは違い、ネイティブアプリケーションはデバイス上に物理的にインストールして、ネットワーク接続の有無に関係なく実行できます。ネイティブアプリケーションはその他のシステムアプリケーションと並んで配置されます。アプリケーションおよびすべてのユーザデータは、iTunesを介してユーザのコンピュータと同期されます。

iPhone OSのアーキテクチャ

iPhone OSのアーキテクチャは、Mac OS Xの基本アーキテクチャに似ています。上位レベルでiPhone OSは、iPhoneおよびiPod touchハードウェアと画面に表示されるアプリケーションとの間の仲介役としての役割を果たします。その様子を図 1-1に示します。作成するアプリケーションは、ハードウェアと直接的にはやり取りせず、適切なドライバとやり取りをするシステムインターフェイスを介します。このような抽象化によって基盤ハードウェアの変更からアプリケーションを保護します。

図 1-1 iPhone OSの最上位層にあるアプリケーション



注： アプリケーションは、一般的に基盤ハードウェアの変更から保護されますが、それでもiPhoneとiPod touchの各デバイス間の違いをコード内では考慮する必要があります。

iPhone OSは、かなり直接的にソフトウェアスタックを使用します。このスタックの底辺部には、Machカーネルドライバとハードウェアドライバがあります。これらは、デバイス上でのプログラムの実行全体を管理します。このレイヤの上に、開発に使用するコアテクノロジーとインターフェイスを含むいくつかのレイヤがあります。iPhone OSでは、カーネルやドライバレベルのインターフェイスは公開されていませんが、スタックより上のレベルのテクノロジーは公開されています。公開されているテクノロジーの詳細については、「iPhone OSテクノロジー」（17 ページ）を参照してください。

iPhone SDKの構成要素

iPhone SDKには、IntelベースのMacintoshコンピュータを利用してiPhoneアプリケーションを開発するために必要なすべてのインターフェイス、ツール、およびリソースが含まれています。

Appleでは、フレームワークと呼ばれる特殊なパッケージとしてほとんどのシステムインターフェイスを提供しています。**フレームワーク**は、共有ダイナミックライブラリとそのライブラリをサポートするために必要なリソース（ヘッダファイル、画像、ヘルパアプリケーションなど）を含む1つのディレクトリです。フレームワークを使用するには、その他の共有ライブラリと同様にそれらをアプリケーションプロジェクトにリンクします。フレームワークをプロジェクトにリンクするとそのフレームワークの機能にアクセスできるようになります。また、開発ツールにヘッダファイルとその他のフレームワークリソースの場所を知らせることもできます。

フレームワークのほかにAppleでは、標準の共有ライブラリという形でいくつかのテクノロジーを提供しています。iPhone OSはUNIXをベースにしているため、オペレーティングシステムの下位レベルを構成する多くのテクノロジーはオープンソースのテクノロジーから派生しています。このため、これらのテクノロジー用のインターフェイスは標準のライブラリおよびインターフェイスのディレクトリで利用できます。

iPhone SDKのその他の主要なコンポーネントには、以下のものがあります。

- Xcodeツール - iPhoneアプリケーションの開発をサポートするツール群を提供します。これには、以下の重要なアプリケーションが含まれています。
  - **Xcode** - アプリケーションのプロジェクトを管理し、コードの編集、コンパイル、実行、およびデバッグが可能な統合開発環境。Xcodeはその他の多くのツールと統合されており、開発の際に使用するメインアプリケーションです。
  - **Interface Builder** - ユーザインターフェイスを視覚的に組み立てるためのツールです。これを利用して作成したインターフェイスオブジェクトは、特殊なリソースファイル形式として保存され実行時にアプリケーションにロードされます。
  - **Instruments** - 実行時のパフォーマンス分析およびデバッグツール。Instrumentsを使用すると、アプリケーションの実行時の動作についての情報を収集し、潜在的な問題を識別できます。
- **iPhone Simulator** - iPhoneテクノロジーのスタックをシミュレートするMac OS Xアプリケーション。これを利用すると、iPhoneアプリケーションをIntelベースのMacintoshコンピュータ上でローカルにテストできます。
- **iPhone Reference Library** - iPhone SDKには、デフォルトでiPhone OSのリファレンス文書が含まれています。より完全なバージョンのiPhone Reference Library（サンプルコードと概念文書を含む）は、iPhone OS Libraryドキュメントセット(Doc Set)に照会すると、ダウンロードできます。Xcodeから、「ヘルプ(Help)」＞「製品ドキュメント(Documentation)」を選び、左側のカラムにあるiPhone OS Libraryドキュメントセット(Doc Set)の隣の「照会(Subscribe)」ボタンをクリックします。

iPhone SDKはアプリケーションを記述するために必要なソフトウェアを提供しますが、XcodeとInstrumentsを利用すると、接続されたデバイスと直接やり取りをしてターゲットハードウェア上でコードを実行したりデバッグしたりもできます。実際のデバイス上で開発するには、Appleの有料のiPhoneデベロッパプログラムにサインアップし、デバイスを開発用に構成する必要があります。iPhoneデベロッパプログラムの詳細については、http://developer.apple.com/jp/iphone/programにアクセスしてください。

iPhone SDKのインストール方法、およびiPhoneアプリケーションを開発するためのiPhone SDKの使用方法については、『*iPhone Development Guide*』を参照してください。iPhone OSのフレームワークの詳細、および下位レベルのシステムライブラリの場所の詳細については、「iPhone OSのフレームワーク」 （43 ページ）を参照してください。

作成可能なアプリケーション

ユーザは、デバイス上で2つの異なる種類のカスタムアプリケーション（Webアプリケーションとネイティブアプリケーション）を実行できます。Webアプリケーションは、HTML、CSS (Cascading Style Sheets)、およびJavaScriptコードを組み合わせて対話型アプリケーションを実現します。これは、

Webサーバ上に配置され、ネットワーク経由で転送されてSafariのWebブラウザ内で実行されます。一方、ネイティブアプリケーションはデバイスに直接インストールされ、ネットワーク接続がなくても実行できます。

iPhone SDKは、デバイスのホーム(Home)画面にのみ表示されるネイティブなフォアグラウンドアプリケーションの作成をサポートします。その他の種類のコード（ドライバ、バックグラウンドアプリケーション、フレームワーク、またはダイナミックライブラリなど）の作成はサポートしません。フレームワークのコードやダイナミックライブラリをアプリケーションに組み込むには、プロジェクトをビルドする際にそのコードを静的にアプリケーションの実行可能ファイルにリンクする必要があります。

リファレンスライブラリの使いかた

iPhone Reference Libraryには、iPhoneアプリケーションを書き始めるのに役立つドキュメント、サンプルコード、チュートリアルなどが含まれています。リファレンスライブラリには、概念的な入門書から下位レベルのAPIリファレンス文書まで数千ページにもおよぶ文書が含まれています。このため、情報の検索方法を理解することは開発プロセスにおける重要なステップです。リファレンスライブラリでは、内容を参照しやすくするためにいくつかの異なる技法を使用しています。

iPhone Reference Libraryには、Apple Developer ConnectionのWebサイト、またはXcodeからアクセスできます。Xcodeで、「ヘルプ(Help)」＞「製品ドキュメント(Documentation)」を選ぶと、Xcodeの「ドキュメント(Documentation)」ウインドウが表示されます。これは、iPhone向け開発についての情報にアクセスするための中心的なリソースです。このウインドウを使用して、ドキュメントの参照や検索の実行、および後から参照したい文書にブックマークを付けることができます。これらの文書は、内容ごとに「ドキュメントセット(Doc Set)」にグループ化されています。これは、更新しやすくするためと関連するドキュメントセットのみに検索範囲を限定できるようにするためです。

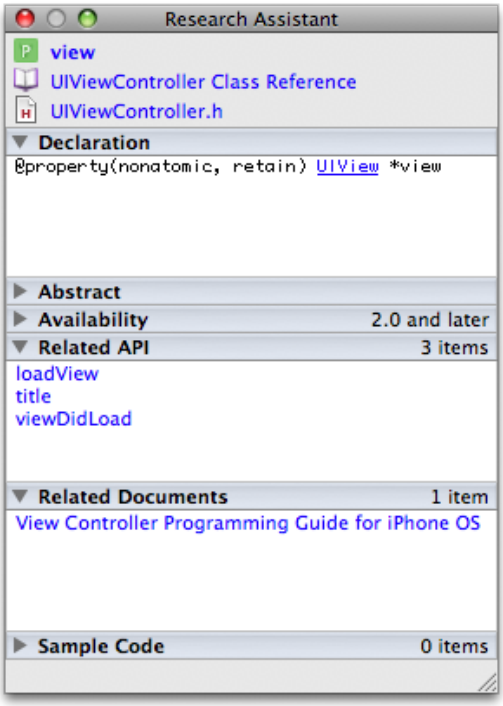
iPhoneアプリケーションを書き始める前に、iPhone OS Libraryの「ドキュメントセット(Doc Set)」に照会してその内容をざっと参照する必要があります。iPhone Reference Libraryにはたくさんの情報が含まれているため、少なくともそのレイアウトに慣れておくことが重要です。図 1-2は、Xcodeの「ドキュメント(documentation)ウインドウ」に表示されたリファレンスライブラリのメインページを示しています。このページには、推奨コンテンツと入門書にアクセスするためのクイックリンクがあります。左側のカラムには、より対象を絞ったライブラリのコンテンツへのリンクがあります。ライブラリは、探しているトピック、フレームワーク、またはリソースタイプごとに参照できます。ウインドウの右上隅にある「検索(Search)」フィールドを使用して、特定の情報についてのリファレンスライブラリの内容を検索することもできます。

図 1-2 iPhone Reference Library

重要： iPhone Reference Libraryの内容は定期的に更新されます。したがって、これを照会していると常に最新情報を参照できることが保証されます。照会するとライブラリの内容がデスクトップにダウンロードされますが、iPhone Dev Center (http://developer.apple.com/iphone)からの最新のドキュメント、リリースノート、テクニカルQ&A、およびサンプルコードにアクセスすることもできます。すべての文書はHTML形式で参照できます。また、そのほとんどはPDF形式でも参照できます。

リファレンスライブラリには膨大な量の情報が含まれているため、コードを記述しようとしている最中にそれらすべての情報をソートするのはやっかいです。特定の情報をすばやく検索できるように、Xcodeには図 1-3に示すような「リサーチアシスタント(Research Assistant)」があります。Xcodeでこのウインドウを表示するには、「ヘルプ(Help)」＞「リサーチアシスタントを表示(Show Research Assistant)」を選び、コードを記述している間デスクトップ上でこのウインドウを開いたままにします。コードを入力すると、Xcodeは、現在の選択内容またはテキスト挿入位置に基づいて常にこのウインドウの内容を更新します。そこには、指定されたシンボルについての情報（構文と説明を含む）が表示されます。また、関連するドキュメントやサンプルコードなどのリソースも表示されます。このウインドウ内のリンクをクリックすると、リファレンスライブラリ内の対応するリソースが表示されます。

図 1-3 Xcodeリサーチアシスタント(Research Assistant)

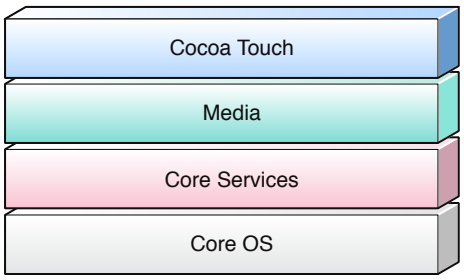


「ドキュメント(Documentation)ウインドウ」と「リサーチアシスタント(Research Assistant)」の使いかたの詳細については、『*Xcode Workspace Guide*』を参照してください。

iPhone OSテクノロジー

iPhone OSテクノロジーの実装は、図2-1に示すように1つのレイヤセットで表現できます。システムの下位の各レイヤには、すべてのアプリケーションが依存する基本的なサービスがあります。一方、上位レベルの各レイヤには、より洗練されたサービスとテクノロジーが含まれます。

図 2-1 iPhone OSのレイヤ



コードを記述する場合、可能な限り下位レベルのフレームワークではなく上位レベルのフレームワークを選ぶようにします。上位レベルのフレームワークは、下位レベルの構成体に関してオブジェクト指向の抽象化を提供するために用意されています。一般的に、これらの抽象化によって記述しなければならないコードの行数が少なくなり、ソケットやスレッドなどの複雑になりがちな機能がカプセル化されるため、コード記述はずっと簡単になります。下位レベルのテクノロジーは抽象化されますが、隠されるわけではありません。下位レベルのフレームワークを使用したいデベロッパや、上位レベルでは公開されていない下位レベルのフレームワークの側面を使用したいデベロッパは、依然としてこれらを利用できます。

以下の各セクションでは、最上位のレイヤから下位に向かって、公開されているiPhone OSのそれぞれのレイヤに関する詳細情報を説明します。

Cocoa Touchレイヤ

Cocoa TouchはiPhone OSの最も重要なレイヤの1つです。このレイヤは、iPhone OSでのアプリケーションの実装に必要なインフラストラクチャを提供する重要なフレームワークで構成されています。アプリケーションを開発する場合は常にこれらのフレームワークから開始し、必要な場合にのみ下位レベルのフレームワークを利用すべきです。

Apple Push Notificationサービス

iPhone OS 3.0以降では、Apple Push Notificationサービスが、アプリケーションが実際には実行中でないときでも新着情報があることをユーザに通知する手段を提供します。このサービスを使用すると、文字による通知のプッシュ、音声による警告のトリガ、アプリケーションアイコンへの数字付きバッジの追加を行うことができます。これらのメッセージは、ユーザに、アプリケーションを開いて関連情報を受け取る必要があることを知らせます。

設計の観点から見ると、iPhoneアプリケーションのPush Notificationを機能させるには2つの部分が必要です。1つ目として、iPhoneアプリケーションへの通知の送信を要求し、アプリケーションデリゲートがそれらを処理するよう設定する必要があります。デリゲートは、`UIApplication`共有オブジェクトと一緒に機能してこれらの両方のタスクを実行します。2つ目に、まず初めの段階に通知を生成するために、サーバ側のプロセスを提供する必要があります。このプロセスは、独自のローカルサーバ上に置き、Apple Push Notificationサービスと一緒に機能させて通知をトリガします。

アプリケーションを設定してRemote Notificationを使用する方法については、『*Apple Push Notification Service Programming Guide*』を参照してください。

Address Book UIフレームワーク

Address Book UIフレームワーク(`AddressBookUI.framework`)は、新規の連絡先を作成したり既存の連絡先を編集または選択したりするための、標準的なシステムインターフェイスの表示に使われるObjective-Cのプログラミングインターフェイスです。このフレームワークを利用するとアプリケーションに連絡先情報を表示するために必要な作業が簡単になります。また、ほかのアプリケーションと同じインターフェイスを使用できるため、プラットフォーム全体の一貫性を保証できます。

Address Book UIフレームワークのクラス、およびその使いかたの詳細については、『*Address Book Programming Guide for iPhone OS*』および『*Address Book UI Framework Reference*』を参照してください。

アプリケーションでの電子メール

iPhone OS 3.0以降では、Message UIフレームワーク(`MessageUI.framework`)が、ユーザの送信ボックス内での電子メールメッセージ作成とキューイングのサポートを提供します。メッセージ作成サポートはView Controllerインターフェイスで構成され、これをアプリケーションに提供できます。このインターフェイスの各フィールドに、送信するメッセージの内容をそれぞれ埋めることができます。メッセージに含める宛先、件名、本文の内容、および任意の添付も設定できます。ユーザには、その後、メッセージを確定する前にメッセージを編集する機会があります。確定すると、メッセージは送信用にユーザの送信ボックスのキューに入れられます。

Message UIフレームワークのクラスの詳細については、『*Message UI Framework Reference*』を参照してください。

Map Kitフレームワーク

iPhone OS 3.0以降では、**Map Kitフレームワーク**(`MapKit.framework`)にマップインターフェイスが提供されており、これを自身のアプリケーションに埋め込むことができます。このインターフェイスは、「マップ(Maps)」アプリケーション内のこのインターフェイスの動作に基づいて、カスタム情報の注釈を付けられるスクロール可能なMap Viewを提供しています。自身のアプリケーションビュー

の中にこのビューを埋め込み、現在表示されている地図の領域やユーザの位置など、地図のさまざまな属性をプログラムで設定できます。また、独自の注釈を定義したり標準的な注釈（ピンマーカなど）を使用したりして、地図の領域をハイライトしたり追加情報を表示したりもできます。

Map Kitフレームワークのクラスの詳細については、『*MapKit Framework Reference*』を参照してください。

ピアツーピアサポート

iPhone OS 3.0以降では、**Game Kitフレームワーク**(`GameKit.framework`)を使ってアプリケーションにピアツーピアネットワーク機能を追加できます。特にこのフレームワークは、ピアツーピア接続機能とゲーム内ボイス機能のサポートを提供します。これらの機能はマルチプレーヤーゲームでもっともよく使用されていますが、ゲーム以外のアプリケーションに組み込むこともできます。このフレームワークは、Bonjourの上に構築されているシンプルかつ強力な一連のクラスを通じてネットワーク機能を提供します。これらのクラスはネットワークの詳細の多くを抽象化しており、ネットワークプログラミングの経験が浅いデベロッパでも簡単にアプリケーションにネットワーク機能を組み込むことができるようになっています。

Game Kitフレームワークの使いかたの詳細については、『*Game Kit Programming Guide*』および『*Game Kit Framework Reference*』を参照してください。

UIKitフレームワーク

UIKitフレームワーク(`UIKit.framework`)には、グラフィカルな、イベント駆動型アプリケーションをiPhone OS上に実装するための主要な基盤を提供するObjective-Cプログラミングインターフェイスが含まれています。iPhone OS上のアプリケーションはすべてこのフレームワークを使用して、核となる以下の機能セットを実装します。

- アプリケーション管理
- カット、コピー、ペーストのサポート
- グラフィックスとウインドウ処理のサポート
- タッチベースイベントおよびモーションベースイベントの処理のサポート
- ユーザインターフェイス管理
- 標準的なシステムビューとコントロールを表すオブジェクト
- テキストおよびWebコンテンツのサポート
- URLスキームによるシステム上のほかのアプリケーションとの統合
- Apple Push Notificationサービスのサポートについては、「Apple Push Notificationサービス」（18 ページ）を参照
- 体の不自由なユーザのためのアクセシビリティのサポート

アプリケーション開発の基本的なコードを提供するほかに、UIKitは以下のようなデバイス固有機能のサポートも組み込んでいます。

- 加速度センサー
- 内蔵カメラ（ある場合）

- ユーザのフォトライブラリ
- デバイス名およびモデル情報
- バッテリー状態情報
- 近接センサー情報

UIKitフレームワークのクラスの詳細については、『*UIKit Framework Reference*』を参照してください。

Mediaレイヤ

Mediaレイヤには、モバイルデバイス上で利用できる最高のマルチメディア体験を演出することを目的とする、グラフィックス、オーディオ、およびビデオの各テクノロジーが含まれています。さらに重要なことには、これらのテクノロジーは見た目もサウンドもすばらしいアプリケーションを簡単に開発できるように設計されています。iPhone OSの上位レベルのフレームワークを利用すると、高度なグラフィックスとアニメーションをすばやく作成できます。また、下位レベルのフレームワークを利用すると、望みどおりの動作をさせるために必要なツールにアクセスできます。

グラフィックステクノロジー

高品質のグラフィックスは、iPhoneアプリケーションの重要な部分です。可能な場合は常に、既存のビューのあらかじめレンダリングされた画像とUIKitフレームワークの関数を使用して、アプリケーションのユーザインターフェイスを描画すべきです。ただし、UIkitが提供する基本的なクラスと関数ではデベロッパのニーズを満たすことができない場合もあります。このような場合は、以降の各セクションで説明するテクノロジーを使用してカスタム描画を実行できます。

Quartz

Core Graphicsフレームワーク(`CoreGraphics.framework`)には、Quartz 2D描画API用のインターフェイスが含まれています。**Quartz**は、Mac OS Xで使われているものと同じ、高度なベクトルベースの描画エンジンです。パスベースの描画、アンチエイリアス化されたレンダリング、グラデーション、画像、色、座標空間の変換、およびPDF文書の作成、表示、解析をサポートします。APIはC言語ベースですが、オブジェクトベースの抽象化を使用して基礎的な描画オブジェクトを表します。これにより、グラフィックスコンテンツの保存と再利用が簡単に行えます。

Quartzを使用してコンテンツを描画する方法の詳細については、『*Quartz 2D Programming Guide*』および『*Core Graphics Framework Reference*』を参照してください。

Core Animation

Quartz Coreフレームワーク(`QuartzCore.framework`)には、Core Animationインターフェイスが含まれています。**Core Animation**は、最適化されたレンダリングパスを使用して複雑なアニメーションや視覚効果を実装する、アニメーションと合成の高度なテクノロジーです。このテクノロジーは、アニメーションや効果を設定し、パフォーマンスを発揮できるようにハードウェアを使用してレンダリングされる、高水準のObjective-Cインターフェイスを提供します。Core Animationは、システムの標準的な動作の多くをアニメーション化する`UIView`などのUIKitクラスをはじめ、iPhone OSの多くの部分に組み込まれています。また、このフレームワークのOvjective-Cインターフェイスを使用してカスタムアニメーションを作成することもできます。

アプリケーションでのCore Animationの使いかたの詳細については、『*Core Animation Programming Guide*』および『*Core Animation Reference Collection*』を参照してください。

OpenGL ES

OpenGL ESフレームワーク(`OpenGLES.framework`)は、2Dコンテンツや3Dコンテンツを描画するためのツールを提供します。これはC言語ベースのフレームワークで、デバイスハードウェアと密接に連携して動作し、フルスクリーンのゲーム形式アプリケーションに対して高いフレームレートを提供します。

OpenGLフレームワークは常にEAGLインターフェイスと組み合わせて使用します。これらのインターフェイスはOpenGL ESフレームワークの一部で、OpenGL ESの描画コードとアプリケーションのネイティブウインドウオブジェクトとの間のインターフェイスを提供します。

iPhone OS 3.0以降では、OpenGL ESフレームワークは、OpenGL ES 2.0およびOpenGL ES 1.1のどちらのインターフェイス仕様もサポートしています。2.0仕様は、フラグメントシェーダおよび頂点シェーダのためのサポートを提供し、iPhone OS 3.0以降を実行する特定のiPhone OSデバイスでのみ利用できます。OpenGL ES 1.1のサポートは、すべてのiPhone OSベースのデバイスおよびすべてのバージョンのiPhone OSで利用できます。

アプリケーションでのOpenGL ESの使いかたの詳細については、『*OpenGL ES Programming Guide for iPhone*』を参照してください。リファレンス情報については、『*OpenGL ES Framework Reference*』を参照してください。

オーディオテクノロジー

iPhone OSで利用可能なオーディオテクノロジーは、ユーザに豊かなオーディオ体験を提供するのに役立つよう設計されています。これには、高品質オーディオの再生や録音の機能、これらの機能をサポートするデバイス上でのバイブレーションのトリガ機能が含まれています。

iPhone OSのオーディオテクノロジーは、以下のオーディオフォーマットをサポートします。

- AAC
- Apple Lossless (ALAC)
- A-law
- IMA/ADPCM (IMA4)
- リニアPCM
- µ-law
- DVI/Intel IMA ADPCM
- Microsoft GSM 6.10
- AES3-2003

AV Foundation

iPhone OS 2.2以降には、AV Foundationフレームワーク(AVFoundation.framework)にオーディオコンテンツを再生するためのObjective-Cクラスが含まれています。このサポートを使用して、ファイルベースまたはメモリベースの任意の長さのサウンドを再生できます。複数のサウンドを同時に再生したり、各サウンドのさまざまな再生の側面を制御できます。iPhone OS 3.0以降では、このフレームワークには録音オーディオのサポートおよびオーディオセッション情報の管理も含まれます。

AV Foundationフレームワークのクラスの詳細については、『*AV Foundation Framework Reference*』を参照してください。

Core Audio

オーディオのネイティブサポートは、表2-1に示すCore Audioファミリのフレームワークによって提供されます。**Core Audio**は、ステレオベースのオーディオの操作をサポートするC言語ベースのインターフェイスです。iPhone OS内でCore Audioを使用すると、アプリケーション内でオーディオを生成、録音、ミキシング、および再生することができます。また、バイブレーション機能をサポートしている機種では、Core Audioを使用してデバイス上でバイブレーション機能にアクセスできます。

表 2-1 Core Audioフレームワーク

フレームワーク	サービス
`CoreAudio.framework`	Core Audio全体で使われるオーディオデータタイプを定義します。
`AudioToolbox.framework`	オーディオファイルおよびストリームに対して再生および録音サービスを提供します。このフレームワークは、オーディオファイルの管理やシステムの警告音もサポートします。
`AudioUnit.framework`	オーディオ処理モジュールである組み込みのAudio Unitを使用するためのサービスを提供します。

Core Audioの詳細については、『*Core Audio Overview*』を参照してください。Audio Toolboxフレームワークを使用してサウンドを再生する方法については、『*Audio Queue Services Programming Guide*』および『*Audio Toolbox Framework Reference*』を参照してください。

OpenAL

Core Audioに加え、iPhone OSには**Open Audio Library (OpenAL)**のサポートが含まれています。OpenALインターフェイスは、アプリケーションで定位のオーディオを提供するために使用できるクロスプラットフォーム標準です。これを使用すると、定位のオーディオ出力が必要なゲームやその他のプログラムに高性能で高品質なオーディオを実装できます。OpenALはクロスプラットフォームの標準であるため、OpenALを使用して記述したiPhone OS上のコードモジュールは、ほかの多くのプラットフォームに移植して実行できます。

OpenALおよびその使いかたの詳細については、http://www.openal.orgを参照してください。

ビデオテクノロジー

iPhone OSでは、**Media Playerフレームワーク**(`MediaPlayer.framework`)によって、フルスクリーンのビデオ再生をサポートしています。このフレームワークは、`.mov`、`.mp4`、`.m4v`、および`.3gp`のファイル名拡張子をもち、次の圧縮規格を使用するムービーファイルの再生をサポートします。

- H.264ビデオ：最高1.5Mbps、640×480ピクセル、毎秒30フレーム、H.264バージョンのLow-Complexityベースラインプロファイル（最高160KbpsのAAC-LCオーディオ）、48kHz、`.m4v`、`.mp4`、`.mov`ファイルフォーマットのステレオオーディオ
- H.264ビデオ：最高768Kbps、320×240ピクセル、毎秒30フレーム、最高レベル1.3のベースラインプロファイル（最高160KbpsのAAC-LCオーディオ）、48kHz、`.m4v`、`.mp4`、`.mov`ファイルフォーマットのステレオオーディオ
- MPEG-4ビデオ：最高2.5Mbps、640×480ピクセル、毎秒30フレーム、シンプルプロファイル（最高160KbpsのAAC-LCオーディオ）、48kHz、`.m4v`、`.mp4`、`.mov`ファイルフォーマットのステレオオーディオ
- 複数のオーディオフォーマット(「オーディオテクノロジー」（21 ページ）で示したフォーマットを含む)

Media Playerフレームワークのクラスの詳細については、『*Media Player Framework Reference*』を参照してください。

Core Servicesレイヤ

Core Servicesは、すべてのアプリケーションで使用する基本的なシステムサービスを提供します。これらのサービスを直接使わないとしても、システム内のほとんどの部分がこれらの上に構築されています。

Address Book

Address Bookフレームワーク(`AddressBook.framework`)は、ユーザのデバイス上に格納されている連絡先情報にプログラムからアクセスするための手段を提供します。アプリケーションが連絡先情報を使用する場合は、このフレームワークを使用してユーザの連絡先データベース内のレコードにアクセスしたり修正したりできます。たとえば、チャットプログラムでこのフレームワークを使用して、チャットセッションを開始可能な連絡先リストを取得したりその連絡先をカスタムビューに表示したりすることができます。

Address Bookフレームワークの関数の詳細については、『*Address Book Framework Reference*』を参照してください。

Core Data

iPhone OS 3.0以降では、**Core Dataフレームワーク**(`CoreData.framework`)は、Model-View-Controllerアプリケーションのデータモデルを管理するためのテクノロジーです。Core Dataは、すでにデータモデルが高度に構造化されているアプリケーションでの使用を目的としています。プログラムで

データ構造を定義する代わりに、Xcodeのグラフィカルなツールを使ってデータモデルを表すスキーマを作成できます。実行時には、Core Dataフレームワークを通してデータモデルエンティティのインスタンスの作成、管理、使用が行われます。

アプリケーションのデータモデルを管理することにより、Core Dataはアプリケーションの作成に必要なコード量を大幅に削減します。Core Dataは以下の機能も提供します。

- パフォーマンスの最適化のためのSQLiteデータベース内のオブジェクトデータ格納域
- Table Viewの結果を管理するための新しい`NSFetchedResultsController`クラス
- 基本的なテキスト編集以外の取り消し(Undo)／やり直し(Redo)の管理
- プロパティ値の検証のサポート
- 変更の伝達のサポート、およびオブジェクト間の関係の一貫性の保証
- メモリ内のデータの、グループ化、フィルタ処理、編成のサポート

新しいアプリケーションの開発や、既存のアプリケーションに対して重要な更新を計画している場合は、Core Dataの使用を検討してください。iPhoneアプリケーションでのCore Dataの使用方法の例については、『*Core Data Tutorial for iPhone OS*』を参照してください。Core Dataフレームワークのクラスの詳細については、『*Core Data Framework Reference*』を参照してください。

Core Foundation

Core Foundationフレームワーク(`CoreFoundation.framework`)は、iPhoneアプリケーションの基本的なデータ管理およびサービス機能を提供する、C言語ベースのインターフェイスセットです。このフレームワークは、以下をサポートしています。

- コレクションデータ型（配列、集合など）
- バンドル
- 文字列管理
- 日付と時刻の管理
- 未加工データブロック管理
- 環境設定管理
- URLおよびストリーム操作
- スレッドおよび実行ループ
- ポートおよびソケット通信

Core Foundationフレームワークは、Foundationフレームワークと密接に関係しており、同じような基本機能にObjective-Cインターフェイスを提供します。FoundationのオブジェクトとCore Foundationの型を混在させる必要がある場合は、この2つのフレームワーク間にある「toll-free bridging」（犠牲を伴わない橋渡し）を利用できます。**toll-free bridging**とは、いくつかのCore Foundation型とFoundation型は、どちらのフレームワークのメソッドや関数でも同じように使用できることを意味します。このサポートは、コレクションデータ型や文字列データ型など多くのデータ型に利用できます。それぞれのフレームワークのクラスと型の説明には、オブジェクトがtoll-free bridgingに対応しているかどうかが示されています。また、対応している場合には対応先のオブジェクトが示されています。

このフレームワークの詳細については、『*Core Foundation Framework Reference*』を参照してください。

Core Location

Core Locationフレームワーク(`CoreLocation.framework`)を利用すると、デバイスの現在の緯度と経度を判別できます。このフレームワークは利用可能なハードウェアを使い、近隣のGPS、携帯電話、またはWi-Fiの信号情報に基づいてユーザの位置を三角法で求めます。「マップ(Maps)」アプリケーションは、この機能を使用してユーザの現在の位置を地図上に表示します。このテクノロジーをアプリケーションに組み込んで、ユーザに位置情報を提供することができます。たとえば、近隣のレストラン、店舗、または施設を検索するようなサービスでは、ユーザの現在位置に基づいて検索ができます。

iPhone OS 3.0以降では、このフレームワークは、適切なハードウェアが装備されているiPhone OSベースデバイス上のコンパス情報へのアクセスもサポートします。

Core Locationフレームワークのクラスの詳細については、『*Core Location Framework Reference*』を参照してください。

Foundationフレームワーク

Foundationフレームワーク(`Foundation.framework`)は、Core Foundationフレームワーク（「Core Foundation」（24 ページ）を参照）内の多くの機能へのObjective-Cのラッパーを提供します。Foundationフレームワークは、以下の機能をサポートします。

- コレクションデータ型（配列、集合など）
- バンドル
- 文字列管理
- 日付と時刻の管理
- 未加工データブロック管理
- 環境設定管理
- URLおよびストリーム操作
- スレッドおよび実行ループ
- Bonjour
- 通信ポート管理
- 国際化

Foundationフレームワークのクラスの詳細については、『*Foundation Framework Reference*』を参照してください。

In App Purchase

iPhone OS 3.0以降では、**Store Kitフレームワーク**(`StoreKit.framework`)は、iPhoneアプリケーション内からのコンテンツやサービスの購入をサポートします。たとえば、この機能を使用すると、ユーザは追加のアプリケーション機能のロック解除ができるようになります。また、ゲームデベロッパであれば、この機能を使用して追加のゲームレベルを提供することもできます。どちらの場合も、会計面の処理、ユーザのiTunes Storeアカウントを通じてのペイメントリクエストの処理、購入に関する情報のアプリケーションへの提供は、Store Kitフレームワークによって扱われます。

Store Kitは、会計面のトランザクションに焦点をしぼり、トランザクションが安全かつ正確に行われることを保証します。アプリケーションでは、購入インターフェイスの表示や適切なコンテンツのダウンロード（またはロック解除）など、会計面以外のトランザクションを処理します。この役割分担により、デベロッパはコンテンツ購入のユーザ体験を制御できます。デベロッパは、ユーザにどんな種類の購入インターフェイスを表示するか、またそれをいつ行うかを決定します。また、アプリケーションに最も適した配布メカニズムを決定します。

Store Kitフレームワークの使いかたの詳細については、『*Store Kit Programming Guide*』および『*Store Kit Framework Reference*』を参照してください。

SQLite

SQLiteライブラリにより、軽量なSQLデータベースをアプリケーションに組み込むことができます。これにより、リモートで別にデータベースサーバプロセスを実行する必要がありません。アプリケーションからローカルデータベースファイルを作成し、ファイル内のテーブルやレコードを管理できます。ライブラリは汎用の目的のために設計されていますが、データベースのレコードにすばやくアクセスできるようにさらに最適化されています。

SQLiteライブラリのアクセス用ヘッダファイルは*<iPhoneSDK>*`/usr/include/sqlite3.h`にあります。*<iPhoneSDK>*は、Xcodeのインストールディレクトリ内のターゲットSDKへのパスです。SQLiteの使用の詳細については、http://www.sqlite.orgを参照してください。

XMLサポート

Foundationフレームワークは、XMLドキュメントから要素を取得するための`NSXMLParser`クラスを提供しています。XMLコンテンツの操作の追加サポートは、`libXML2`ライブラリによって提供されています。このオープンソースライブラリを利用すると任意のXMLデータをすばやく解析したり記述したりできます。また、XMLコンテンツをHTMLに変換することもできます。

`libXML2`ライブラリのアクセス用ヘッダファイルは*<iPhoneSDK>*`/usr/include/libxml2/`ディレクトリにあります。*<iPhoneSDK>*は、Xcodeのインストールディレクトリ内のターゲットSDKへのパスです。`libXML2`の使用に関する詳細については、http://xmlsoft.org/index.htmlを参照してください。

Core OSレイヤ

CFNetwork

CFNetworkフレームワーク(`CFNetwork.framework`)は、ネットワークプロトコルを処理するためのオブジェクト指向の抽象化を提供する、高性能なC言語ベースのインターフェイスセットです。これらの抽象化によって、プロトコルスタックに対して細かな制御が可能になり、BSDソケットなどの下位レベルの構成体を使うのが簡単になります。このフレームワークを使用することで、FTPサーバおよびHTTPサーバとの通信やDNSホスト解決などの作業が簡単に行えます。CFNetworkフレームワークを使用して実行できる作業をいくつか示します。

- BSDソケットの使用
- SSLまたはTLSを使用した暗号化接続の作成
- DNSホストの解決
- HTTPサーバおよびHTTPSサーバの認証を伴うHTTPの使用
- FTPサーバの使用
- Bonjourサービスの公開、解決、ブラウズ

CFNetworkは、物理的にも論理的にもBSDソケットを基にしています。CFNetworkの使いかたの詳細については、『*CFNetwork Programming Guide*』および『*CFNetwork Framework Reference*』を参照してください。

アクセサリのサポート

iPhone OS 3.0以降では、External Accessoryフレームワーク(`ExternalAccessory.framework`)が、iPhoneまたはiPod touchデバイスに接続されているハードウェアアクセサリとの通信のためのサポートを提供します。アクセサリは、デバイスの30ピンDockコネクタを使用して接続するか、Bluetoothを使用してワイヤレスに接続できます。External Accessoryフレームワークは、利用可能な個々のアクセサリについての情報を取得し、通信セッションを初期化する方法を提供します。その後、サポートされているコマンドのいずれかを使用して、アクセサリを直接自由に操作できます。

このフレームワークの使いかたの詳細については、『*iPhone Application Programming Guide*』の「Device Support」を参照してください。External Accessoryフレームワークのクラスの詳細については、『*External Accessory Framework Reference*』を参照してください。iPhoneおよびiPod touchデバイス向けアクセサリの開発については、http://developer.apple.com/jpを参照してください。

セキュリティ

組み込みのセキュリティ機能に加えて、iPhone OSでは、アプリケーションが管理するデータのセキュリティを保証するために利用できる、明示的な**Securityフレームワーク**(`Security.framework`)も提供しています。このフレームワークは、証明書、公開鍵と非公開鍵、および信用ポリシーを管理するインターフェイスを提供します。暗号技術的にセキュアな擬似乱数の生成をサポートします。また、キーチェーンへの証明書や暗号鍵の保存もサポートします。キーチェーンは、機密性の高いユーザデータのためのセキュリティ保護されたリポジトリです。

CommonCryptoインターフェイスは、対称暗号化、HMAC、およびダイジェストをサポートします。このダイジェスト機能は、OpenSSLライブラリに標準的に含まれる機能と本質的に互換性のある関数を提供します。ただし、OpenSSLライブラリはiPhone OSでは利用できません。

iPhone OS 3.0以降では、作成した複数のアプリケーション間でキーチェーンアイテムを共有できます。アイテムを共有すると、同じスイートのアプリケーションをよりスムーズに相互運用できます。たとえば、この機能を使用して、複数のユーザパスワードやその他の要素（つまり、共有しなければ、個々のアプリケーションから別々にユーザへの指示を要求される可能性のあるような要素）を共有できます。アプリケーション間でデータを共有するには、各アプリケーションのXcodeプロジェクトを正しいエンタイトルメントで設定する必要があります。

Securityフレームワークに関連した関数や機能の詳細については、『*Security Framework Reference*』を参照してください。キーチェーンにアクセスする方法の詳細については、『*Keychain Services Programming Guide*』を参照してください。Xcodeプロジェクトでのエンタイトルメントの設定の詳細については、『*iPhone Development Guide*』を参照してください。設定できるエンタイトルメントの詳細については、『*Keychain Services Reference*』の`SecItemAdd`関数の説明を参照してください。

System

システムレベルには、カーネル環境、ドライバ、オペレーティングシステムの低レベルUNIXインターフェイスが含まれています。カーネル自体はMachを基にしており、オペレーティングシステムのすべての側面に関与しています。また、仮想メモリシステム、スレッド、ファイルシステム、ネットワーク、およびプロセス間通信を管理します。このレイヤのドライバは、利用可能なハードウェアとシステムフレームワークとの間のインターフェイスも提供します。カーネルおよびドライバへのアクセスは、セキュリティを確保するために、限られたシステムフレームワークとアプリケーションに限定されています。

iPhone OSは、オペレーティングシステムの下位レベルの多くの機能にアクセスするためのインターフェイスを提供します。アプリケーションは、`LibSystem`ライブラリを通してこれらの機能にアクセスします。これらのインターフェイスはC言語ベースであり、以下をサポートしています。

- スレッド処理（POSIXスレッド）
- ネットワーク（BSDソケット）
- ファイルシステムアクセス
- 標準I/O
- BonjourサービスおよびDNSサービス
- ロケール情報
- メモリ割り当て
- 数学的演算処理

多くのCore OSテクノロジー用のヘッダファイルは*<iPhoneSDK>*`/usr/include/`ディレクトリにあります。*<iPhoneSDK>*は、Xcodeのインストールディレクトリ内のターゲットSDKへのパスです。これらのテクノロジーに関連付けられた関数の詳細については、『*iPhone OS Manual Pages*』を参照してください。

Cocoaからの移行

Cocoaデベロッパの方であれば、iPhone OSに提供されているフレームワークの多くに馴染みがあるはずです。iPhone OSの基本的なテクノロジースタックは、多くの点でMac OS Xのものと似ています。しかし類似性がある一方、iPhone OSのフレームワークは、Mac OS Xの対応するフレームワークとは完全に同じではありません。この章では、iPhoneアプリケーションを作成する際に直面する違いについて説明します。また、より大きな違いに対処する方法についても説明します。

注： この章は、Cocoaの用語とプログラミング技法をすでによく理解しているデベロッパを対象としています。Cocoaアプリケーション（およびiPhoneアプリケーション）で使われている基本的なデザインパターンの詳細については、『*Cocoa Fundamentals Guide*』を参照してください。

移行の際の一般的な注意

Model-View-Controllerデザインパターンに基づいて設計されているCocoaアプリケーションの場合は、アプリケーションの主要部分をiPhone OSに移植するのは比較的簡単です。iPhone OS用のアプリケーションの設計の詳細については、『*iPhone Application Programming Guide*』を参照してください。

データモデルの移行

FoundationフレームワークおよびCore Foundationフレームワークのクラスに基づくデータモデルを持つCocoaアプリケーションは、ほとんど（またはまったく）変更せずにiPhone OSに移植できます。どちらのフレームワークもiPhone OSでサポートされており、多少の違いはありますがMac OS Xのものと実質的には同じです。その違いのほとんどは、あまり重要でないか、iPhone版のアプリケーションではいずれにしても削除しなければならない機能に関連するものです。たとえば、iPhoneアプリケーションはAppleScriptをサポートしません。違いの詳細なリストについては、「Foundationフレームワークの違い」（33 ページ）を参照してください。

CocoaアプリケーションがCore Dataの上に構築されている場合、iPhone OS 3.0以降のiPhoneアプリケーションへデータモデルを移行できます。Core Dataは、以前のバージョンのiPhone OSではサポートされません。iPhone OSのCore Dataフレームワークは、バイナリおよびSQLiteデータストア（XMLデータストアではない）をサポートし、既存のCocoaアプリケーションからの移行をサポートします。サポートされるデータストアでは、Core Dataリソースファイルを iPhoneアプリケーションプロジェクトにコピーして、それを使用することができます。Xcodeプロジェクトでの Core Dataの使いかたの詳細については、『*Core Data Programming Guide*』を参照してください。

画面上にたくさんのデータを表示するCocoaアプリケーションの場合は、iPhone OSに移植する際にデータモデルを単純化することを考えます。iPhone OSでもたくさんのデータを使用する充実したアプリケーションを作成することはできますが、それがユーザのニーズを満たさない場合もあることを心に留めておいてください。モバイルユーザは一般に、最小限の時間で最も重要な情報だけを必要とします。画面のスペースが限られているため、一度にあまりに多くのデータをユーザに提供す

るのは現実的ではありません。また、そのデータをロードするための余分な処理によってアプリケーションが遅くなる可能性もあります。iPhone OSで優れたパフォーマンスとユーザ体験を提供するには、Cocoaアプリケーションのデータ構造をリファクタリングする価値はあります。

ユーザインターフェイスの移行

iPhone OSのユーザインターフェイスは、Mac OS Xのユーザインターフェイスとはかなり異なる構成および実装になっています。たとえば、Cocoaのビューとウインドウを表すオブジェクトを例に見てみましょう。iPhone OSとCocoaはともにビューとウインドウを表すオブジェクトを持っていますが、オブジェクトの動作はプラットフォームごとに少し異なります。さらに、iPhone OSでは画面サイズが限られているためビューに表示するものを厳選する必要があります。また、ビューには、ユーザが指で差すのに十分なスペースがなければなりません。

ビューオブジェクト自体の違いだけでなく、実行時のビューの表示方法にも大きな違いがあります。たとえば、Cocoaアプリケーションでたくさんのデータを表示する場合は、ウインドウのサイズを大きくしたり、複数のウインドウを使用したり、データを管理するタブを使用したりすることができます。iPhoneアプリケーションには固定サイズのウインドウが1つしかありません。このためアプリケーションは、情報を適当なサイズに分割してそれらを複数のビューで表示しなければなりません。情報を分割する目的は、「画面に相当する分のコンテンツ」を1つ以上作成することです。これをビューを定義する際の基礎として使用します。たとえば、Cocoaで階層的なデータリストを表示するには、`NSBrowser`オブジェクトを1つ使用します。一方iPhone OSでは、階層の各レベルの情報を表示する別のビューセットを作成する必要があります。これによってインターフェイスの設計が多少複雑になりますが、これは情報を表示するための非常に重要な手段であるため、iPhoneではこの種の編成のためにかなりのサポートを提供しています。

View ControllerはMac OS X v10.5でCocoaに導入されましたが、まだ一般的には使用されていません。iPhoneアプリケーションでは、View Controllerがユーザインターフェイスを管理するためのインフラストラクチャの非常に重要な部分を提供しています。View Controllerは、ユーザインターフェイスの表示を管理します。また、システムと連携してアプリケーションのリソースがあまり多くのメモリを消費してパフォーマンスを低下させないようにします。したがって、View Controllerの役割とアプリケーションでのそれらの使いかたを理解することは、ユーザインターフェイスを設計するために非常に重要です。

iPhone OSでのユーザインターフェイスの設計方針についての一般的な情報は、『*iPhone Human Interface Guidelines*』を参照してください。インターフェイスの作成に使用するウインドウとビュー、およびそれらを支える基盤アーキテクチャの追加情報については、『*iPhone Application Programming Guide*』を参照してください。View Controller、およびそれを使用したユーザインターフェイスのフローの作成方法の詳細については、『*View Controller Programming Guide for iPhone OS*』を参照してください。

メモリ管理

iPhone OSでは常に、メモリ管理されたモデルを使ってオブジェクトの保持、解放、自動解放を行います。ガベージコレクションは、iPhone OSではサポートされません。

iPhone OSベースのデバイスでは、メモリはMacintoshコンピュータよりもさらに厳しく制限されるため、自動解放プールの使用を調整して、自動解放されたオブジェクトの蓄積を防ぐ必要があります。可能な場合は常に、オブジェクトを自動解放するのではなく、直接的に解放する必要があります。たくさんのオブジェクトを短いループ内で割り当てる場合は、これらのオブジェクトを直接解

放するか、ループコード内の適切な場所に自動解放プールを作成して一定の間隔で自動解放されたオブジェクトを解放する必要があります。ループの終了まで待機すると、メモリ不足の警告やアプリケーションの強制終了につながることがあります。

フレームワークの違い

iPhone OSのフレームワークのほとんどはMac OS Xにも存在しますが、これらのフレームワークの実装と使用方法にはプラットフォームによる違いがあります。以降の各セクションでは、既存のMac OS Xデベロッパがiphoneアプリケーションを開発する際に気付く重要な違いをいくつか示します。

UIKitとAppKit

iPhone OSでは、UIKitフレームワークが、グラフィカルアプリケーションを作成するためのインフラストラクチャを提供し、イベントループの管理や、その他のインターフェイス関連のタスクを実行します。ただし、UIKitフレームワークはAppKitフレームワークとはまったく別物です。したがって、iPhoneアプリケーションを設計する際には別物として扱わなければなりません。このような理由から、CocoaアプリケーションをiPhone OSに移植する場合は、かなりの数のインターフェイス関連のクラスおよびロジックを置き換えなければなりません。表 3-1は、iPhone OSアプリケーションに必須なものは何かを理解するのに役立つ、フレームワーク間の明確な違いのリストです。

表 3-1　インターフェイステクノロジーの違い

相違	説明
ドキュメントのサポート	iPhone OSでは、アプリケーションは通常、ウインドウは1つのみであり、個別のドキュメントオブジェクトやドキュメントウインドウは使用しません。このため、アプリケーションによって開いたファイルはすべてそのアプリケーションによって直接管理され、ウインドウの内容の更新に使われます。
ビュークラス	UIKitは、デベロッパ向けに厳選されたカスタムビューおよびコントロールのセットを提供しています。AppKitにあるビューとコントロールの多くは、iPhoneおよびiPod touchデバイスではうまく動作しません。その他のビューにもiPhone固有の代替手段があります。たとえば、`NSBrowser`クラスの代わりに、iPhoneではまったく異なるパラダイム(Navigation Controller)を使って階層情報の表示を管理します。iPhone OSで利用できるビューとコントロール、およびその使いかたの詳細については、『*iPhone Human Interface Guidelines*』を参照してください。
ビューの座標系	iPhone OSでは、QuartzとUIKitコンテンツのための描画モデルは、1つの例外を除きMac OS Xのモデルとほとんど同じです。Mac OS Xの描画モデルでは、ウインドウとビューの原点はデフォルトでは左下隅にあり、それぞれの軸は上方向と右方向に伸びます。iPhone OSでは、原点は左上隅にあり、それぞれの軸は下方向と右方向に伸びます。Mac OS Xでは、この座標系を「反転した」座標系と呼んでいます。iPhone OSでは、これがデフォルトの座標系です。グラフィックスと座標系の詳細については、『*iPhone Application Programming Guide*』を参照してください。

相違	説明
ビューとしてのウインドウ	概念上は、iPhone OSのウインドウとビューは、Mac OS Xの場合と同様に同じ構成体を表しています。しかし、実装の観点では、この2つのプラットフォームはウインドウとビューをまったく異なった方法で実装しています。Mac OS Xでは、`NSWindow`クラスは`NSResponder`のサブクラスですが、iPhone OSでは、`UIWindow`クラスは実際は`UIView`のサブクラスです。この継承関係の違いは、ウインドウがCore Animationレイヤを使用して描画サーフェスを実現していることを意味します。UIKitのすべてでウインドウオブジェクトを持つ理由は、オペレーティングシステム内でウインドウのレイヤ化をサポートするためです。たとえば、システムは、アプリケーションのウインドウの前景に表示される独立したウインドウにステータスバーを表示します。 iPhone OSとMac OS Xのもう1つの違いは、ウインドウの使用方法です。Mac OS Xアプリケーションにはいくつでもウインドウを持たせることができますが、ほとんどのiPhoneアプリケーションではウインドウが1つしかありません。1つのiPhoneアプリケーション内で複数の画面分の情報を表示するには、ウインドウを変更するのではなく、アプリケーションウインドウのカスタムビューを切り替えます。
イベント処理	UIKitのイベント処理モデルは、Mac OS Xのイベント処理モデルとは大きく異なります。マウスやキーボードのイベントの代わりに、UIKitは、タッチイベントをビューに送付します。これらのイベントには、さまざまなメソッドセットを実装する必要があります。また、イベント処理コード全体にいくつかの変更を加える必要もあります。たとえば、ローカルな追跡ループから待機中のイベントを抽出してタッチイベントを追跡してはいけません。iPhone OSアプリケーションのイベント処理の詳細については、『*iPhone Application Programming Guide*』を参照してください。
ターゲット／アクションモデル	UIKitは、3形態のアクションメソッドをサポートします。これは、1つしかサポートしないAppKitとは対照的です。UIKitのコントロールは、対話のさまざまなフェーズでアクションを起動できます。1つの対話に複数のターゲットを割り当てることもできます。このように、UIKitでは、1つのコントロールが1つの対話サイクルの間に複数のターゲットに複数の異なるアクションを送付することができます。iPhone OSアプリケーションのターゲット／アクションモデルについては、『*iPhone Application Programming Guide*』を参照してください。
描画と印刷サポート	UIKitの描画機能は、UIKitクラスのレンダリング要件をサポートするように拡張されています。このサポートには、画像のロードと表示、文字列の表示、カラー管理、フォント管理、および矩形の描画とグラフィックスコンテキストの取得を行ういくつかの関数が含まれています。UIKitには、汎用の描画クラスは含まれていません。いくつかほかの選択肢（QuartzとOpenGL ES）が、iPhone OSにすでにあるためです。印刷は、プリンタやほかの印刷関連ハードウェアをiPhoneまたはiPod touchに接続するための直接的なサポートがないため、サポートされていません。グラフィックスと描画の詳細については、『*iPhone Application Programming Guide*』を参照してください。
テキストサポート	iPhone OSのテキストサポートは、電子メールとメモの作成を対象としています。UIKitクラスを使ってアプリケーションは、簡単な文字列と、多少複雑なHTMLコンテンツの表示や編集を行うことができます。高度なワードプロセッサにあるような高度なテキストレイアウトやグリフ生成機能は、モバイル環境にはほとんど必要とされないため、iPhone OSには含まれていません。

相違	説明
アクセサメソッドとプロパティの使用	UIKitでは、クラス宣言によってプロパティを広範囲に利用しています。プロパティはMac OS Xバージョン10.5から導入されました。つまり、AppKitフレームワークで多くのクラスが作成された後に導入されたものです。UIKitでは、AppKitの同じgetterメソッドとsetterメソッドを単にまねるのではなく、クラスインターフェイスを簡素化する方法としてプロパティが使われます。プロパティの使いかたの詳細については、『*The Objective-C 2.0 Programming Language*』の「Declared Properties」を参照してください。
コントロールとセル	UIKitのコントロールは、セルを使いません。セルは、Mac OS Xではビューに代わる軽量な手段として使われます。UIKitでは、ビュー自体が非常に軽量なオブジェクトであるため、セルは必要ありません。命名規則に反して、`UITableView`クラスで使うように設計されたセルが、実際には`UIView`クラスに基づいています。
Table View	iPhone OSの`UITableView`クラスは、AppKitフレームワークの`NSTableView`クラスと`NSOutlineView`クラスの中間と考えることができます。この2つのAppKitクラスからの機能を使って、小さな画面でのデータの表示により適したツールを作成しています。`UITableView`クラスは一度に1つの列を表示し、関連する行をセクションにグループ化できます。情報の階層的なリストを表示したり編集するための手段でもあります。`UITableView`クラスの詳細については、『*UITableView Class Reference*』を参照してください。
メニュー	iPhone OS向けに記述されたほとんどすべてのアプリケーションは、それに相当するMac OS Xアプリケーションよりもコマンドセットがはるかに小さくなります。このため、iPhone OSではメニューはサポートされません。また、一般にメニューは必要ありません。必要なコマンドがわずかな場合は、一般に、ツールバーまたはボタンセットの方が適しています。データに基づくメニューに対しては、多くの場合はピッカーまたはナビゲーションコントローラインターフェイスが適しています。
Core Animationレイヤ	iPhone OSでは、すべての描画サーフェスの背後にCore Animationレイヤがあり、多くのビュー関連プロパティに対して暗黙的なアニメーションサポートがすでに提供されています。アニメーションサポートが組み込まれているため、通常はコード内で明示的にCore Animationレイヤを使う必要はありません。ほとんどのアニメーションは、単に、対象となるビューの必要なプロパティを変更するだけで実行できます。レイヤを直接使う必要があるのは、レイヤツリーを厳密に制御する必要があるときや、ビューレベルでは公開されていない機能を使う必要があるときだけです。Core AnimationレイヤがiPhone OSの描画モデルにどのように組み込まれているかの詳細については、『*iPhone Application Programming Guide*』を参照してください。

UIKitのクラスの詳細については、『*UIKit Framework Reference*』を参照してください。

Foundationフレームワークの違い

同じバージョンのFoundationフレームワークを、Mac OS XとiPhone OSの両方で利用できます。また、ほとんどのクラスが両方で利用できます。どちらのフレームワークも、値、文字列、コレクション、スレッド、およびその他の一般的なデータ型の管理をサポートしています。表 3-2は、iPhone

OSに含まれていない主な機能領域のリストです。ここには、それに関連するクラスが利用できない理由も示されています。この表には、できる限り代わりに利用できる代替テクノロジーを掲載しました。

表 3-2 iPhone OSでは利用できないFoundationテクノロジー

テクノロジー	メモ
メタデータと述語の管理	Spotlightのメタデータと検索述語は、Spotlight自体がiPhone OSでサポートされていないため、サポートされません。
分散オブジェクトとポート名サーバ管理	分散オブジェクトテクノロジーは利用できませんが、`NSPort`ファミリのクラスを使ってポートおよびソケットとやり取りできます。また、Core FoundationフレームワークとCFNetworkフレームワークを使ってネットワーク要件に対応することもできます。
Cocoaバインディング	CocoaバインディングはiPhone OSではサポートされません。代わりに、iPhone OSでは、ターゲット／アクションモデルを若干変更したバージョンを採用しており、コード中のアクションの処理方法に柔軟性が加わっています。
Objective-Cのガベージコレクション	ガベージコレクションは、iPhone OSではサポートされません。代わりに、メモリ管理されたモデルを使う必要があります。このモデルでは、オブジェクトを保持して所有権を要求し、不要になったらオブジェクトを解放します。
AppleScriptサポート	AppleScriptは、iPhone OSではサポートされません。

Foundationフレームワークは、`NSXMLParser`クラスによってXML解析をサポートします。ただし、その他のXML解析クラス（`NSXMLDocument`、`NSXMLNode`、`NSXMLElement`を含む）は、iPhone OSでは利用できません。`NSXMLParser`クラスのほかに、C言語ベースのXML解析インターフェイスを提供する`libXML2`ライブラリを使用することもできます。

Mac OS Xで利用でき、iPhone OSでは利用できない特定のクラスの一覧については、『*Foundation Framework Reference*』の「The Foundation Framework」にあるクラス階層図を参照してください。

その他のフレームワークの変更

表 3-3は、iPhone OSのその他のフレームワークでの主な違いを示しています。

表 3-3 iPhone OSとMac OS Xに共通のフレームワークにおける違い

フレームワーク	相違
`AddressBook.framework`	このフレームワークには、ユーザの連絡先情報にアクセスするためのインターフェイスが含まれています。iPhone OSでは、このフレームワークのインターフェイスはObjective-C言語ではなく、C言語で書かれています。詳細については、『*Address Book Framework Reference*』を参照してください。

フレームワーク	相違
AudioToolbox.framework AudioUnit.framework CoreAudio.framework	これらのフレームワークのiPhone OSバージョンは、主に、シングルまたはマルチチャンネルのオーディオコンテンツの録音、再生、ミキシングをサポートします。高度なオーディオ処理機能や、カスタムオーディオユニットプラグインはサポートされていません。ただし、iPhone OSには、iPhoneデバイス向けにバイブレートオプションをトリガする機能が追加されています。オーディオサポートの使いかたの詳細については、『*iPhone Application Programming Guide*』の「Multimedia Support」を参照してください。
CFNetwork.framework	このフレームワークには、Core Foundation Networkインターフェイスが含まれています。iPhone OSでは、CFNetworkフレームワークはサブフレームワークではなく、最上位のフレームワークです。ただし、実際のインターフェイスのほとんどは変わっていません。詳細については、『*CFNetwork Framework Reference*』を参照してください。
CoreGraphics.framework	このフレームワークにはQuartzインターフェイスが含まれています。iPhone OSでは、Core Graphicsフレームワークはサブフレームワークではなく、最上位のフレームワークです。Quartzを使ってMac OS Xの場合とまったく同じ方法で、パス、グラデーション、シェーディング、パターン、色、イメージ、ビットマップを作成できます。ただし、PostScriptサポート、イメージのソースとデスティネーション、Quartz Display Servicesサポート、Quartz Event Servicesサポートなど、いくつかのQuartz機能はiPhone OSには含まれていません。詳細については、『*Core Graphics Framework Reference*』を参照してください。
OpenGLES.framework	OpenGL ESは、組み込みシステム向けに特別に設計されたOpenGLのバージョンです。OpenGLデベロッパなら、OpenGL ESインターフェイスには馴染みがあるはずです。ただし、OpenGL ESインターフェイスにもいくつか重要な相違点があります。1つは、使用可能なグラフィックスハードウェアを使って効率よく実行できる機能のみをサポートしているため、非常にコンパクトなインターフェイスである点です。もう1つは、デスクトップOpenGLで標準的に使用する拡張子の多くが、OpenGL ESでは使用できない可能性がある点です。このような相違がありますが、デスクトップ上で通常考えられる操作のほとんどは、同じように実行できます。ただし、既存のOpenGLコードを移植する場合、iPhone OSで異なる描画技術を使用するように、コードの一部を書き直す必要が生じるかもしれません。iPhone OSでのOpenGL ESサポートの詳細については、『*OpenGL ES Programming Guide for iPhone*』を参照してください。
QuartzCore.framework	このフレームワークにはCore Animationインターフェイスが含まれています。Core Animationインターフェイスのほとんどは、iPhone OSとMac OS Xとで同じです。ただし、iPhone OSでは、レイアウトの制約を管理するためのクラスは利用できず、Core Imageフィルタの使用もサポートされていません。また、Core ImageとCore Video用のインターフェイス（これもMac OS Xバージョンのフレームワークに含まれています）はありません。詳細については、『*Quartz Core Framework Reference*』を参照してください。

フレームワーク	相違
`Security.framework`	このフレームワークにはセキュリティインターフェイスが含まれています。iPhone OSでは、このフレームワークは、暗号化と復号、擬似乱数生成、Keychainのサポートを提供することにより、アプリケーションデータのセキュリティ確保に焦点を当てています。このフレームワークには認証または承認のインターフェイスは含まれていません。また証明書の内容の表示もサポートされていません。さらに、Keychainインターフェイスは、Mac OS Xで使われているKeychainインターフェイスの簡略版です。セキュリティサポートの詳細については、『*iPhone Application Programming Guide*』を参照してください。
`System-Configuration.framework`	このフレームワークにはネットワーク関連のインターフェイスが含まれています。iPhone OSでは、このフレームワークには到達性のためのインターフェイスのみ含まれています。これらのインターフェイスを使って、デバイスがどのような方法でネットワークに接続されているか（たとえば、EDGE、GPRS、またはWi-Fiが使われているなど）を特定できます。

iPhone OSのデベロッパツール

iPhone OS向けのアプリケーションを開発するには、Xcodeツールが動作するMac OS Xコンピュータが必要です。Xcodeは、プロジェクト管理、コード編集、実行可能ファイルのビルド、ソースレベルのデバッグ、ソースコードのリポジトリ管理、パフォーマンスチューニング、その他多数の機能をサポートする、Appleの開発ツールスイートです。この開発ツールスイートの中心にあるのは、基本的なソースコード開発環境を提供する、Xcodeアプリケーション自身です。Xcodeは単なるツールではありません。以降の各セクションでは、iPhone OS向けのソフトウェアの開発に使用する主要なアプリケーションを紹介します。

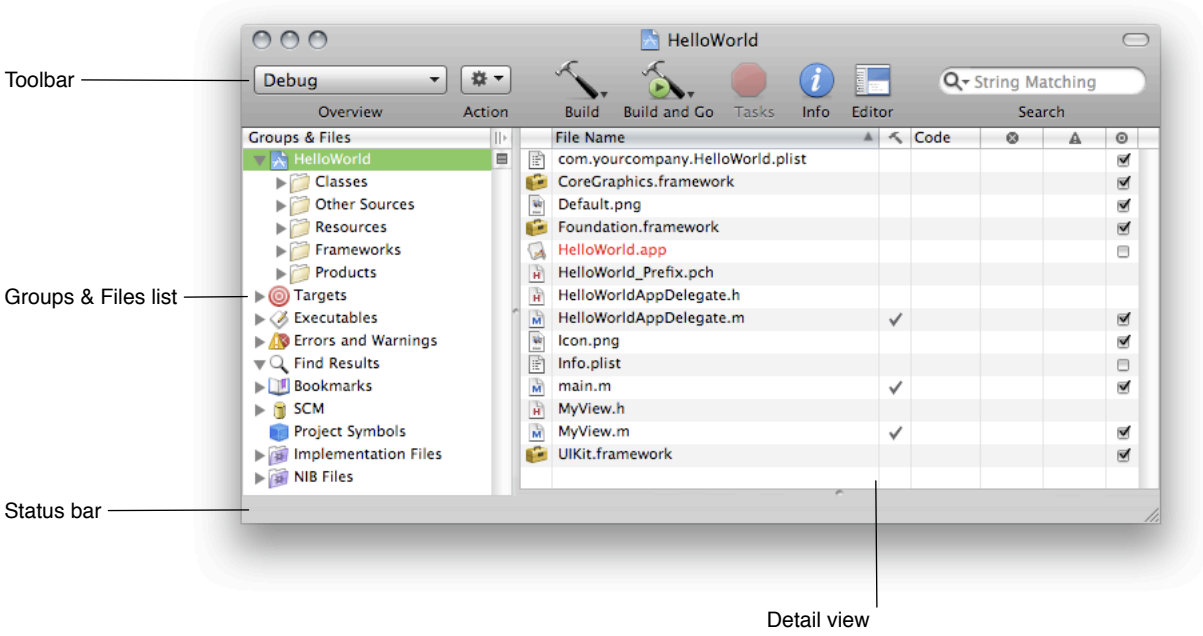
Xcode

開発作業の中心は、Xcodeアプリケーションです。Xcodeは、iPhoneプロジェクトとソースファイルの作成と管理、実行可能ファイルへのコードのビルド、iPhone Simulatorまたはデバイス上でのソースコードの実行とデバッグに必要なあらゆるツールを備えた統合開発環境(IDE)です。Xcodeには、iPhoneアプリケーションの開発を支援する、以下のようなさまざまな機能が組み込まれています。

- ソフトウェア製品を定義するためのプロジェクト管理システム
- 構文の色分け、コード補完、シンボルのインデックス化などの機能を備えたコード編集環境
- Appleドキュメントを表示したり検索するための高度なドキュメントビューア
- 選択中のコードシンボルについての情報を表示するコンテキスト依存のインスペクタ
- 依存関係のチェック機能およびビルド規則の評価機能を備えた高度なビルドシステム
- C言語、C++言語、Objective-C言語、Objective-C++言語、Objective-C 2.0言語、その他の言語をサポートするGCCコンパイラ
- GDBを使用した統合化されたソースレベルでのデバッグ
- 大規模なプロジェクトをネットワーク接続された複数のマシンに分散できる分散コンピューティング
- 1つのファイルのコンパイル時間を短縮する予測コンパイル
- Fix and Continue（修正・続行）、カスタムデータフォーマッタなどの高度なデバッグ機能
- コード全体の動作を変更せずに、コードにグローバルな修正を行う高度なリファクタリングツール
- 軽量なローカルソースコード管理を提供するプロジェクトスナップショットのサポート
- ソフトウェアを解析するためのパフォーマンスツールの起動サポート
- 統合化されたソースコード管理のサポート
- ビルドプロセスを自動化するためのAppleScriptサポート
- DWARFおよびStabsの各デバッグ情報のサポート（DWARFデバッグ情報は、デフォルトですべての新規プロジェクトに生成されます）

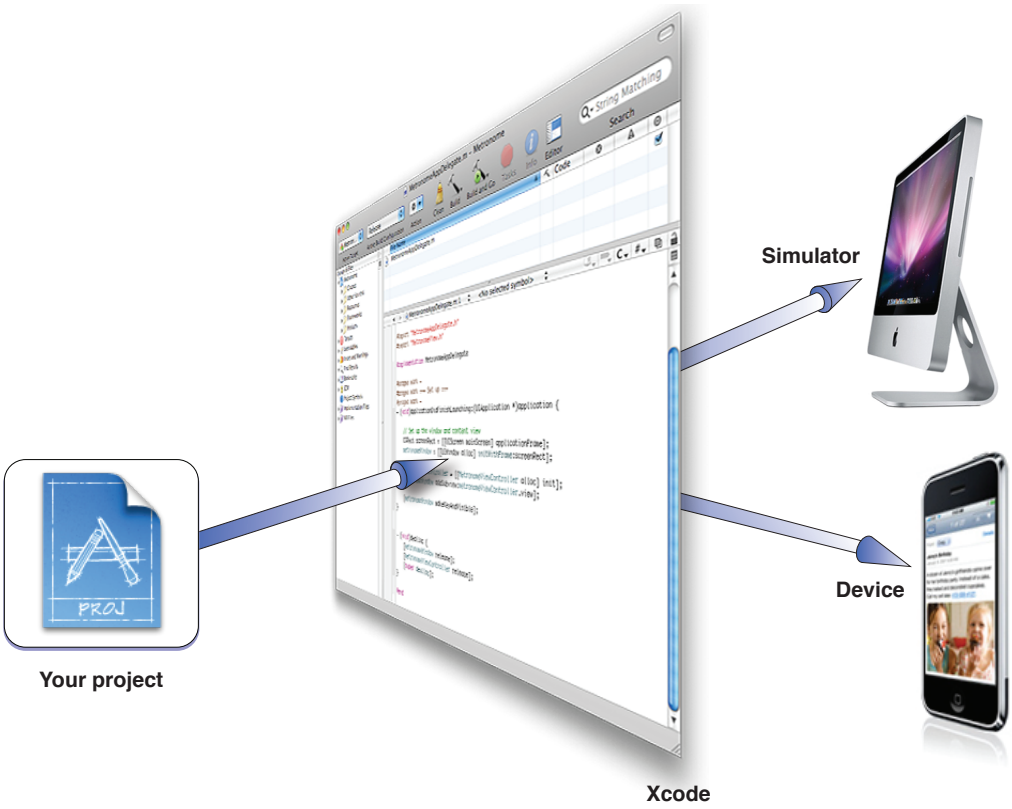
新しいiPhoneアプリケーションを作成するには、まず、Xcodeで新規プロジェクトを作成します。1つのプロジェクトで、ソースファイル、ビルド設定、それらの項目をすべてまとめるために必要なルールなど、作成するアプリケーションに関するすべての情報を管理します。すべてのXcodeプロジェクトの中心にあるのが、図 A-1に示すプロジェクトウインドウです。このウインドウから、作成中のアプリケーションの主要な要素すべてに簡単にアクセスできるようになっています。「グループとファイル(Groups & Files)」リストでは、各種ソースファイルおよびソースファイルから作成されたビルドターゲットなど、プロジェクト内のファイルを管理します。ツールバーからは、よく使うツールやコマンドにアクセスできます。詳細ペインは、プロジェクトの作業領域を構成します。プロジェクトウインドウのその他の部分は、プロジェクトについてのコンテキスト依存の情報を提供します。

図 A-1 Xcodeのプロジェクトウインドウ



Xcodeでアプリケーションをビルドするときには、iPhone Simulator向けにビルドするか、デバイス向けにビルドするかを選択できます。iPhone Simulatorは、アプリケーションが基本的に要求どおりに動作するかをテストするためのローカル環境を提供します。アプリケーションの基本動作に問題がなければ、Xcodeでデバイス用にアプリケーションをビルドし、コンピュータに接続したiPhoneまたはiPod touch上で実行できます。デバイス上でアプリケーションを実行する場合は最高のテスト環境が提供され、Xcodeを使ってそこで実行するコードに組み込みデバッガをアタッチできます。

図 A-2 Xcodeからのプロジェクトの実行

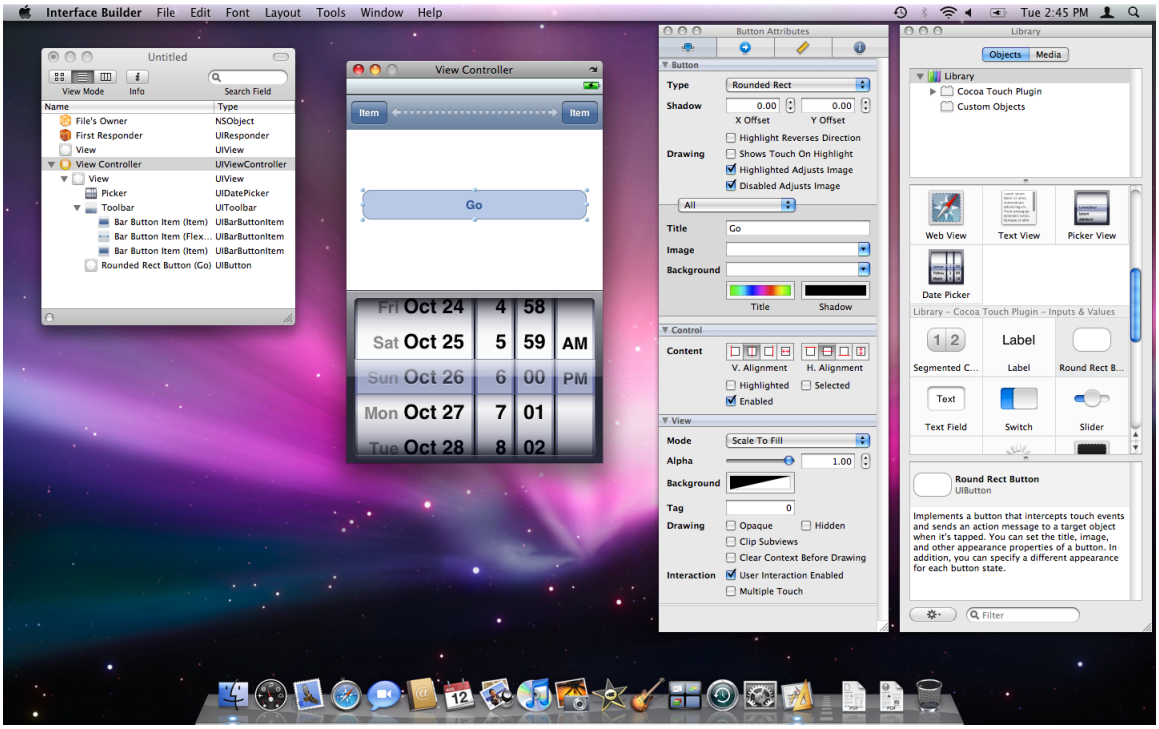


iPhone OSでのプロジェクトのビルドおよび実行の方法の詳細については、『*iPhone Development Guide*』を参照してください。Xcodeの全体的な環境の詳細については、『*Xcode Overview*』を参照してください。

Interface Builder

Interface Builderは、アプリケーションのユーザインターフェイスを視覚的に組み立てるためのツールです。Interface Builderを使用すると、あらかじめ用意されたコンポーネントをドラッグアンドドロップすることによって、アプリケーションのウインドウを組み立てることができます。その様子を図 A-3に示します。コンポーネントには、スイッチ、テキストフィールド、ボタンなど標準のシステムコントロールと、アプリケーションが提供するビューを表すカスタムビューも含まれています。ウインドウの表面にコンポーネントを配置したあと、ドラッグ操作によって位置決めを行い、インスペクタを使って属性を設定し、これらのオブジェクトとコードとの間の関係を確立させることができます。インターフェイスが望みどおりの外見になったら、コンテンツをnibファイルに保存します。このファイルは、独自のリソースファイル形式です。

図 A-3 Interface Builderを使用したiPhoneインターフェイスの作成



Interface Builderで作成するnibファイルには、UIKitが実行時にアプリケーション内に同じオブジェクトを再作成するために必要なすべての情報が含まれます。nibファイルをロードすると、このファイルに格納されているすべてのオブジェクトの実行時バージョンが作成され、Interface Builderでの設定とまったく同じように設定されます。また、新たに作成したオブジェクトと、アプリケーション内の既存のオブジェクトとの間の接続を確立するために指定した接続情報も使われます。この接続情報により、コードにnibファイルオブジェクトへのポインタが与えられるほか、オブジェクト自身がユーザアクションをコードに伝えるために必要な情報も与えられます。

全体的には、Interface Builderを使うことによって、アプリケーションのユーザインターフェイス作成の時間は大幅に節約されます。Interface Builderにより、インターフェイスを構成するオブジェクトの作成、設定、位置指定に必要なカスタムコードの量は抑えられます。Interface Builderはビジュアルなエディタであるため、自分のインターフェイスが実行時にどのように表示されるかを正確に見ることができます。

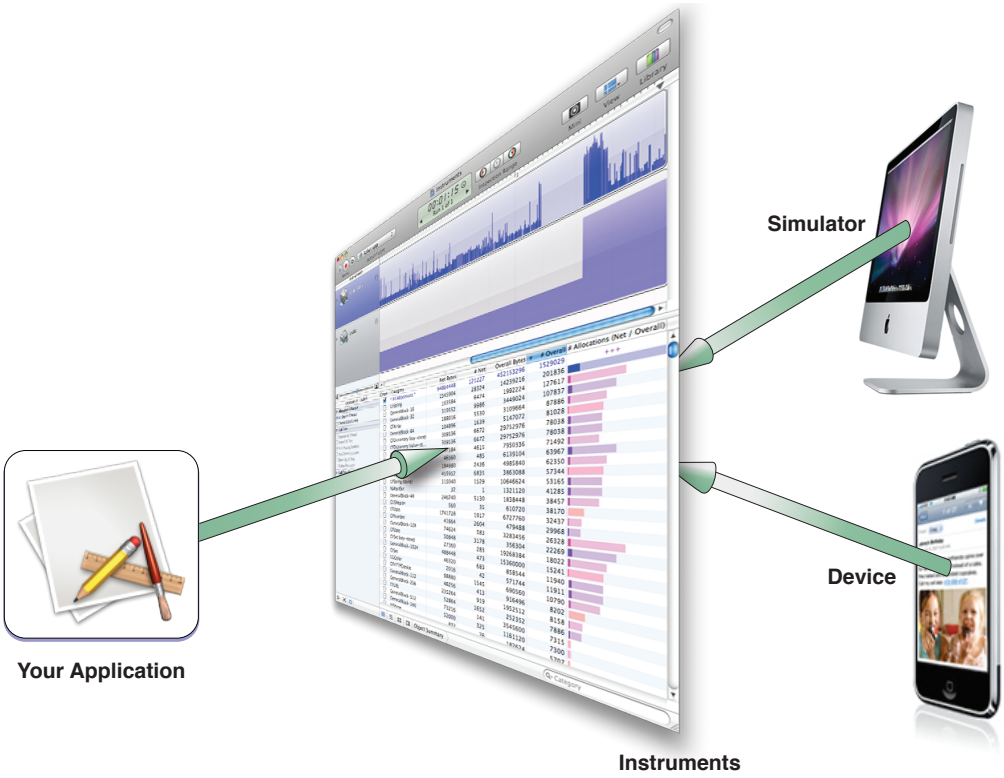
Interface Builderの使用の詳細については、『*Interface Builder User Guide*』を参照してください。

Instruments

自分のソフトウェアに最高のユーザ体験をもたらすことができるように、Instruments環境では、iPhoneアプリケーションをシミュレータまたはデバイス上で実行しながら、そのパフォーマンスを分析できます。Instrumentsは、実行中のアプリケーションからデータを収集し、タイムラインと呼ばれるグラフ表示でそのデータを表現します。アプリケーションのメモリ使用量、ディスクアクティビティ、ネットワークアクティビティ、グラフィックスのパフォーマンスについてのデータを

収集できます。タイムラインビューには、各種の情報がすべて並んで表示されるため、ある特定の領域の動作だけでなく、アプリケーション全体の動作を相関させることができます。さらに詳細な情報を得るには、Instrumentsが収集する詳細なサンプルを表示することもできます。

図 A-4 Instrumentsを使ったアプリケーションのチューニング



タイムラインビューのほかに、Instrumentsは時間の経過に伴うアプリケーションの動作の分析に役立つツールを提供します。たとえば、「Instruments」ウインドウを使って、複数の実行結果からのデータを格納し、アプリケーションの動作が実際に向上しているか、あるいはまだ作業が必要かを見ることができます。これらの実行結果からのデータをInstruments文書に保存し、それをいつでも開くことができます。

iPhoneアプリケーションに対するInstrumentsの使用方法の詳細については、『*iPhone Development Guide*』を参照してください。Instrumentsの使いかたの詳細については、『*Instruments User Guide*』を参照してください。

Shark

Sharkは、iPhoneアプリケーションのパフォーマンス分析に使用できる強力なツールです。Sharkを利用すると、アプリケーションをiPhone OSベースのデバイスで実行している間に、さまざまなオプションを使用してコードのプロファイルができます。プロファイル結果は、アプリケーションの実行時の動作の統計的なサンプリングです。これを、Sharkのデータマイニングツールやグラフツールを使用して表示したり分析できます。これらのツールは、プログラムの実行時の動作を可視化して潜在的なホットスポットを識別するのに役立ちます。

iPhone OSベースのデバイスでのSharkの使用の詳細については、『*Shark User Guide*』を参照してください。

iPhone OSのフレームワーク

この付録には、iPhone OSの各種フレームワークについての情報が掲載されています。これらのフレームワークは、iPhoneプラットフォーム用にソフトウェアを記述するのに必要なインターフェイスを提供します。該当する場合にフレームワークのクラス、メソッド、関数、型、または定数で使用される主要なプレフィックスをこれらの表に掲載しています。独自のシンボル名には、ここに指定されているプレフィックスの使用は避ける必要があります。

デバイスのフレームワーク

表 B-1に、iPhone OSベースのデバイスで利用可能なフレームワークを示します。これらのフレームワークは、
<Xcode>`/Platforms/iPhoneOS.platform/Developer/SDKs/`*<iPhoneSDK>*`/System/Library/Frameworks`
ディレクトリにあります。*<Xcode>*は、Xcodeのインストールディレクトリ、*<iPhoneSDK>*は、ターゲットになっている特定のSDKバージョンを表します。「最初のリリース」の欄は、このフレームワークが最初に導入されたiPhone OSリリースです。

表 B-1 デバイスのフレームワーク

名前	最初のリリース	プレフィックス	説明
`AddressBook.framework`	2.0	`AB`	ユーザの連絡先データベースに直接アクセスするための関数が含まれています。
`AddressBookUI.framework`	2.0	`AB`	システム定義のPeopleピッカーと編集インターフェイスを表示するためのクラスが含まれています。
`AudioToolbox.framework`	2.0	`AU`、`Audio`	オーディオストリームデータを扱ったり、オーディオの再生と録音のためのインターフェイスが含まれています。
`AudioUnit.framework`	2.0	`AU`、`Audio`	オーディオユニットのロードと使用のためのインターフェイスが含まれています。
`AVFoundation.framework`	2.2	`AV`	オーディオの再生と録音のためのObjective-Cインターフェイスが含まれています。
`CFNetwork.framework`	2.0	`CF`	Wi-Fi無線と携帯電話無線によってネットワークにアクセスするためのインターフェイスが含まれています。
`CoreAudio.framework`	2.0	`Audio`	Core Audio全体で使われるオーディオデータタイプを提供します。

名前	最初のリリース	プレフィックス	説明
CoreData.framework	3.0	NS	アプリケーションのデータモデルを管理するためのインターフェイスが含まれています。
CoreFoundation.framework	2.0	CF	基本ソフトウェアサービスを提供します（一般的なデータ型の抽象化、文字列ユーティリティ、コレクションユーティリティ、リソース管理、環境設定など）。
CoreGraphics.framework	2.0	CG	Quartz 2D用のインターフェイスが含まれています。
CoreLocation.framework	2.0	CL	ユーザの位置を決定するためのインターフェイスが含まれています。
External-Accessory.framework	3.0	EA	接続されているハードウェアアクセサリと通信するためのインターフェイスが含まれています。
GameKit.framework	3.0	GK	ピアツーピア接続を管理するためのインターフェイスが含まれています。
Foundation.framework	2.0	NS	Cocoa Foundationレイヤ用のクラスとメソッドが含まれています。
IOKit.framework	2.0	特になし	デバイスによって使用されるインターフェイスが含まれています。このフレームワークを直接インクルードしないでください。
MapKit.framework	3.0	MK	アプリケーションへのマップインターフェイスの埋め込みと、逆ジオコーダ座標の検索のためのクラスが含まれています。
MediaPlayer.framework	2.0	MP	フルスクリーンビデオを再生するためのインターフェイスが含まれています。
MessageUI.framework	3.0	MF	電子メールメッセージの作成とキューイングのためのインターフェイスが含まれています。
MobileCore-Services.framework	3.0	UT	システムによってサポートされるUTI (Uniform Type Identifier)を定義します。
OpenAL.framework	2.0	AL	OpenAL（クロスプラットフォームの定位オーディオライブラリ）用のインターフェイスが含まれています。
OpenGLES.framework	2.0	EAGL、GL	OpenGL ES（クロスプラットフォームのOpenGL 2Dおよび3Dグラフィックスレンダリングライブラリの組み込み版）用のインターフェイスが含まれています
QuartzCore.framework	2.0	CA	Core Animationインターフェイスが含まれています。

名前	最初のリリース	プレフィックス	説明
Security.framework	2.0	CSSM、Sec	証明書、公開鍵と非公開鍵、および信用ポリシーを管理するインターフェイスが含まれています。
StoreKit.framework	3.0	SK	In App Purchaseに関連付けられた会計処理を扱うためのインターフェイスが含まれています。
System-Configuration.framework	2.0	SC	デバイスのネットワーク設定を判別するインターフェイスが含まれています。
UIKit.framework	2.0	UI	iPhoneアプリケーションユーザインターフェイスレイヤ用のクラスとメソッドが含まれています。

シミュレータのフレームワーク

コードを記述しているときは、常にデバイスのフレームワークをターゲットにする必要がありますが、テスト中は特別にシミュレータ用にコードをコンパイルする必要があります。デバイスとシミュレータで利用可能なフレームワークはほとんど同じですが、少しだけ違いがあります。たとえば、シミュレータは、自身の実装の一部にMac OS Xのフレームワークをいくつか使用しています。さらに、デバイスのフレームワークとシミュレータのフレームワークで利用可能な同じインターフェイスでも、システムの制約によって多少異なる場合があります。フレームワークのリストやデバイスのフレームワークとシミュレータのフレームワークの特定の違いの詳細については、『*iPhone Development Guide*』を参照してください。

システムライブラリ

Core OSおよびCore Servicesレベルの特殊なライブラリの中には、フレームワークとしてパッケージ化されていないものもあります。その代わりに、iPhone OSではシステムの`/usr/lib`ディレクトリにたくさんのダイナミックライブラリが含まれています。共有ダイナミックライブラリは、`.dylib`拡張子で識別されます。これらのライブラリのヘッダファイルは、`/usr/include`ディレクトリにあります。

iPhone SDKの各バージョンには、この共有ダイナミックライブラリのローカルコピーが含まれており、これらはシステムと一緒にインストールされます。これらのコピーは、Xcodeプロジェクトからリンクできるように開発用のシステムにインストールされます。特定のバージョンのiPhone OS用のライブラリのリストを調べるには、*<Xcode>*`/Platforms/iPhoneOS.platform/Developer/SDKs/`*<iPhoneSDK>*`/usr/lib`を参照してください。*<Xcode>*は、Xcodeのインストールディレクトリ、*<iPhoneSDK>*は、ターゲットになっている特定のSDKバージョンを表します。たとえば、iPhone OS 3.0 SDK用の共有ライブラリは、`/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS3.0.sdk/usr/lib`にあります。それに対応するヘッダは、`/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS3.0.sdk/usr/include`にあります。

iPhone OSでは、シンボリックリンクを使用してほとんどのライブラリの最新バージョンを参照できます。共有ダイナミックライブラリにリンクする場合は、特定のバージョンのライブラリへのリンクではなくこのシンボリックリンクを使用します。ライブラリのバージョンは、iPhone OSの今後のバージョンで変更される可能性があります。したがって、ソフトウェアを特定のバージョンにリンクした場合、そのバージョンがユーザのシステム上で常に利用できるとは限りません。

書類の改訂履歴

この表は「*iPhone OSテクノロジーの概要*」の改訂履歴です。

日付	メモ
2009-05-27	iPhone OS 3.0向けに更新しました。
2008-10-15	iPhone OSとそのテクノロジーを説明する新規文書。

書類の改訂履歴