iPodライブラリアクセスプログラミングガイド

2011-02-21

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

図、リスト

はじめに

iPodライブラリアクセスによりアプリケーションでユーザの曲、オーディオブック、およびオーディオPodcastを再生できます。APIの設計は基本的な再生を非常に単純にしつつ、高度な検索や再生制御もサポートしています。

iPodライブラリアクセスにより、アプリケーションに対する音楽関連のさまざまな機能強化をiOSでできるようになります。以下に、実現可能ないくつかの例を示します。

- ゲームやエクササイズアプリケーションのサウンドトラックをユーザが組み立てて再生できるようにする
- 友人に送信するために、お気に入りのアーティスト、曲、アルバムの名前をユーザがiPodライブラリから取得できるようにする
- ユーザのiPodライブラリのコンテンツに基づいて、新曲やアーティストのツアースケジュールなどのお勧め情報サービスを提供する

この文書を読むにあたり、『*iOS Application Programming Guide*』および『*iOS Development Guide*』で説明されているiOS開発に慣れている必要があります。iOS向けプログラム作成の紹介チュートリアルについては、『*Your First iOS Application*』を参照してください。

『*iPodライブラリアクセスプログラミングガイド*』（この文書）を読むにあたっては『*Media Player Framework Reference*』が参考になります。この文書は『Media Player Framework Reference』の補足です。

注： iPodライブラリアクセスはデバイス上でのみ動作し、iPhone Simulatorでは動作しません。これは、iPhone SimulatorがデバイスのiPodライブラリにアクセスできないためです。このテクノロジーのクラスを使用してアプリケーションを開発するには、セットアップ済みのiOSデバイスが必要です。

この書類の構成

この文書は次の章で構成されています。

- 「iPodライブラリアクセスについて」（9 ページ）では、このAPIを使用した音楽再生とiPodライブラリアクセスの包括的な概要を示します。
- 「メディアの再生」（19 ページ）では、ミュージックプレーヤーの作成および使用方法について説明します。
- 「メディアアイテムピッカーの使用」（23 ページ）では、ユーザが選択したメディアアイテムを取得するために、ピッカーを起動してデリゲートメソッドを実装する方法を紹介します。

- 「iPodライブラリの使用」（25 ページ）では、デバイスのiPodライブラリからメディアアイテムを取得するための述語およびクエリの作成と使用について詳しく説明します。
- 「用語解説」（33 ページ）では、APIの説明に使用されている用語を定義します。

関連項目

このテクノロジーを最大限に活用するために、次の文書を参照してください。

- 『*Media Player Framework Reference*』。この文書で紹介しているクラスの包括的なリファレンス文書です。
- 『*AddMusic*』。iOSアプリケーション用にダウンロード、学習、改良、および拡張できるiPhoneのサンプルプロジェクトです。
- 『*Audio Session Programming Guide*』。アプリケーションのオーディオ動作全体を構成する方法を説明しています。独自のオーディオとiPodのオーディオ再生を同時に使用するアプリケーションを作成する場合はこの文書を参照してください。
- 『*Core Audio Overview*』。iOS内でのオーディオフォーマットのミキシング方法を説明し、アプリケーションのオーディオで使用できるオーディオフォーマットを解説します。

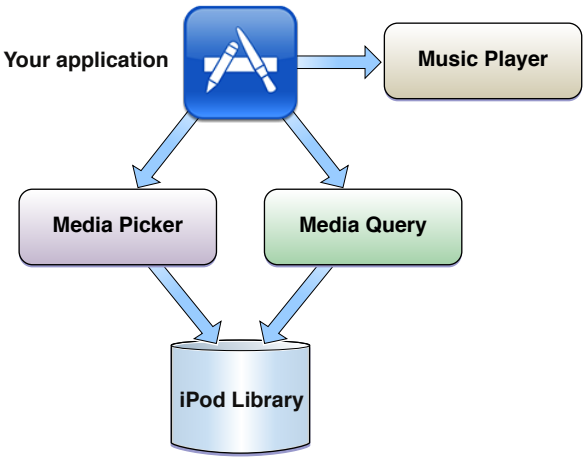
iPodライブラリアクセスについて

iPodライブラリアクセスは、デバイスのiPodライブラリからアイテムを取得して再生するためのiOSのインターフェイスです。iPodライブラリは、ユーザがデスクトップのiTunesから同期させたデバイス上のメディアアイテムの集まりです。

図 1-1に示すように、アプリケーションには2通りのアイテムの取得方法があります。左側のメディアピッカーは、簡単に使用できるパッケージ済みのView Controllerで、組み込みのiPodアプリケーションのミュージック選択インターフェイスのように動作します。多くのアプリケーションにはこれで十分です。

要求される特別なアクセス制御をピッカーで提供できない場合、図のアプリケーションの右下に示すようなメディアクエリインターフェイスを使用できます。これを使用して、デバイスのiPodライブラリからアイテムを述語ベースで指定できます。

図 1-1 iPodライブラリアクセスの使用



図でアプリケーションの右側に描かれているとおり、次に、APIが提供するミュージックプレーヤーを使用して、取得したメディアアイテムを再生します。

図に示されているとおり、このAPIのクラスは、通常は開発における異なる2つの領域（再生するメディアアイテムを検索するためのデータベースアクセス、およびミュージックの再生）に対応しています。ただし、このAPIを効果的に使用するためにどちらの分野のエキスパートでもある必要はありません。iPodライブラリアクセスが代わりに「重労働」をこなしてくれます。

たとえば、再生するのがDRM暗号化されたAACファイル、CDからリッピングしたApple Losslessトラック、iLBCでエンコードされたオーディオPodcast、またはオーディオブックのどれでも、ミュージック再生のためのコードは全部で1行で済みます。システムが自動的にバッファの設定、適切なコーデックの選択を行い、オーディオをデバイスの出力ハードウェアに直接送信します。

ミュージックプレーヤーとメディアアイテムピッカーを使用すると、iPodライブラリにアクセスするコードをまったく書くことなくミュージックの選択と再生を実装できます。データベースアクセスクラスは必要な時は完全なクエリシステムを提供し、必要でなければ目にすることはありません。

注： iPodライブラリアクセスは、オーディオベースのメディアアイテムにのみ適用されます。iPodライブラリのビデオPodcast、ムービー、またはテレビ番組を再生することはできません。

ミュージックプレーヤーについて

アプリケーションは、エンドユーザが組み込みのiPodアプリケーションを使用するのと同様の方法でミュージックプレーヤーを使います。プログラミングによって、再生、一時停止、シークなどが行えます。

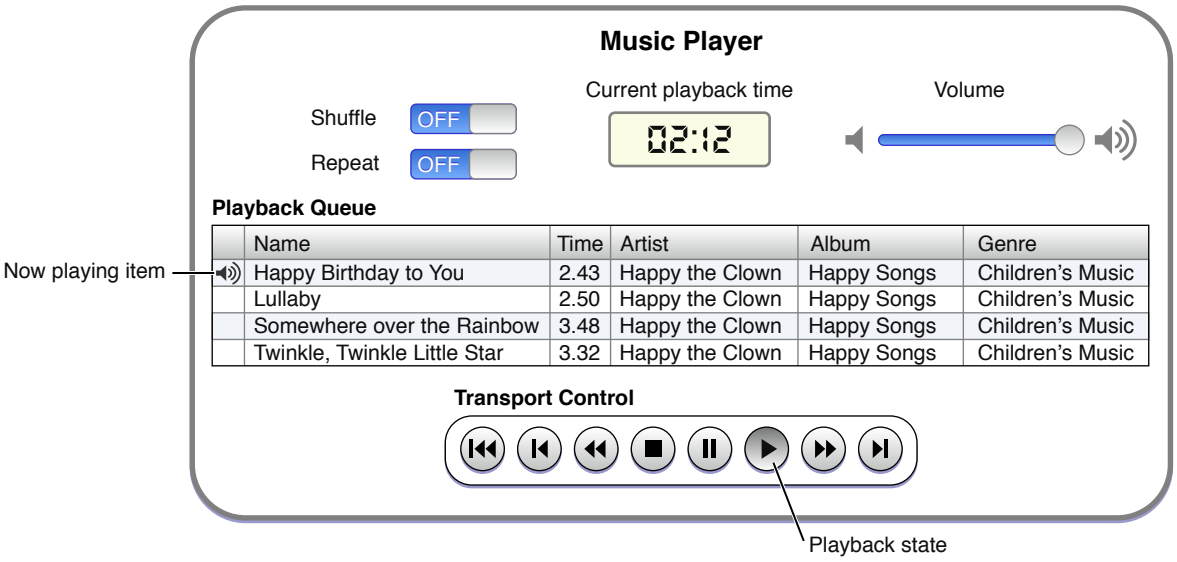
ミュージックプレーヤーの基礎と用語

ミュージックプレーヤー(Music Player)はメディアアイテムの再生に使用するオブジェクトです。再生するメディアアイテムのリストである**再生キュー**を持っています。**メディアアイテム(Media Item)**は、曲、オーディオPodcast、またはオーディオブックです。

メディアアイテムは、ユーザがデスクトップのiTunesと同期した際にデバイス上にコピーされます。デバイス上のメディアアイテムの一式を**iPodライブラリ(iPod library)**と呼びます。

ミュージックプレーヤーは現在再生中のアイテムや再生するよう指定されているアイテムを知っており、ある時点でのアイテムのタイムラインの再生ポイントを知っています。これらの属性を、図1-2に図式的に示します（この図は説明のためだけのものです。ミュージックプレーヤーはユーザインターフェイスを提供しません）。

図 1-2 ミュージックプレーヤーオブジェクトの図式的表現



再生中(Now Playing)のアイテムは特別なステータスを持っています。たとえば、ユーザが組み込みのiPodアプリケーションで曲を一時停止し、開発したアプリケーションを起動した場合、ミュージックプレーヤーはそのアイテムの再生を同じポイントから続けられます。

さらなるプロパティがミュージックプレーヤーに存在し、非常に柔軟性を高めています。図に示すように、ミュージックプレーヤーにはモード、再生状態(Playback State)、および音量(Volume)があります。

- **シャッフル(Shuffle)モード**と**リピート(Repeat)モード**は、組み込みのiPodアプリケーションと同じように動作します。
- **再生状態(Playback State)**は通常のオーディオ再生システムで使われるもので、再生中、停止、一時停止、早送り、または巻き戻しがあります。さらに、次のメディアアイテムや前のメディアアイテムの先頭までスキップしたり、再生キューの先頭に戻ることもできます。
- **音量(Volume)**はデフォルトでは最大で、サイレントまで任意に下げることができます。

ミュージックプレーヤーは、アプリケーションの目的に応じて2種類使用できます。

- **アプリケーションミュージックプレーヤー**は、アプリケーション内でローカルに音楽を再生します。組み込みのiPodアプリケーションの状態に影響を受けたり与えたりすることはありません。具体的に、ミュージックプレーヤーは再生中アイテム、再生状態、モードをそれぞれ保持しています。ユーザがアプリケーションを終了すると、再生していたミュージックは停止します。
- **iPodミュージックプレーヤー**は、実質的に組み込みのiPodアプリケーションを使用し、iPodの状態を共有します。ユーザがこのプレーヤーを使用していたアプリケーションを停止した場合、音楽は再生し続けます。

最後に、ミュージックプレーヤーの使用について、2つの重要なポイントを以下に示します。

- オーディオを再生できるのは一度に1つのミュージックプレーヤーのみです。
- ミュージックプレーヤーは、アプリケーションのメインスレッドでのみ使用できます。

Hello Music Player

ここに、ライブラリアクセスとミュージック再生を行う基本的な*hello world*スタイルの例があります。最小限ながら、動作するミュージックプレーヤーが数分で手に入ります。ユーザインターフェイスがないため、このコードはデバイスのiPodライブラリ全部をキューイングし、起動後すぐに再生を開始します。

注： iPhone Simulatorがデバイスの iPodライブラリにアクセスできないため、以下のステップを実行するには設定済みのデバイスが必要です。

1. **新規のXcodeプロジェクトを作成します。**

   Xcodeで、「Window-based Application」テンプレートから新規プロジェクトを作成します。プロジェクトウインドウで、「Frameworks」グループに`MediaPlayer`フレームワークを追加します。プロジェクトを保存します。

2. **Media Playerフレームワーク用のアンブレラヘッダファイルをインポートします。**

   AppDelegate.hファイルにある既存の#import行の後に、次の行を追加します。

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

3. **ミュージックプレーヤーを作成するコードを追加し、再生するミュージックを割り当てて、再生を開始します。**

   プロジェクトのAppDelegate.m実装ファイルを開きます。applicationDidFinishLaunching:ブロックの終端の前に、リスト 1-1に示す3行のコードを追加します。

リスト 1-1　最小限のミュージックプレーヤー

```
// ミュージックプレーヤーをインスタンス化する
MPMusicPlayerController *myPlayer =
    [MPMusicPlayerController applicationMusicPlayer];

// デバイス上のすべてのメディアアイテムを含む再生キューを割り当てる
[myPlayer setQueueWithQuery: [MPMediaQuery songsQuery]];

// キューの先頭から再生を開始する
[myPlayer play];
```

次に、コード署名IDとバンドル識別子を含め、開発デバイスに合わせてプロジェクトを設定します。デバイスのiPodライブラリに少なくとも曲が1つ存在することも確認します。プロジェクトをビルドして実行します。デバイス上でアプリケーションを起動すると、iPodライブラリ内の最初の曲が再生されます。アプリケーションを停止しない限り、プレーヤーはiPodライブラリのすべてのアイテムを再生し続けます。

ミュージックプレーヤー変更通知について

ミュージックプレーヤーの動作を追跡するためには、ミュージックプレーヤー変更通知を登録します。アプリケーションの状態とミュージックプレーヤーの状態が正しく連携していることを確認するためにこれは必須です。

たとえば、ミュージックの再生を正しく開始するためのイベントシーケンスを次に示します。ミュージックプレーヤーは独自のスレッド上で動作するため、適切な通知を受信するまでユーザインターフェイスを更新しないことに注意してください。

1. ユーザが「再生(Play)」をタップします。
2. アプリケーションがミュージックプレーヤーの再生を起動します。
3. ミュージックプレーヤーが再生を開始し、Playback-State-Change通知を発行します。
4. アプリケーションが通知を受信してミュージックプレーヤーの状態を問い合わせ、確かに再生中であることを確認します。
5. アプリケーションが、「再生(Play)」ボタンを「一時停止(Pause)」に変更するなど、ユーザインターフェイスを必要に応じて更新します。

ミュージックプレーヤー変更通知は、再生状態、再生中アイテム、およびミュージックプレーヤーの音量の追跡をサポートします。「メディアの再生」（19 ページ）でそれらの使いかたを説明します。

ホームシェアリングとiPodミュージックプレーヤー

iOS 4以降、組み込みのiPodアプリケーションとビデオアプリケーションでは、ホームシェアリングを使用して共有ライブラリからメディアを再生できます。ただし、Media Playerフレームワークを使用したサードパーティ製のアプリケーションからは、デバイスのiPodライブラリにしかアクセスできません。つまり、そのアプリケーションでは、ホームシェアリングされた曲のタイトルをユーザインターフェイスで表示することはできません。ただし、共有メディアの再生時に、現在の再生時間や再生状態など、その他の再生情報は利用できます。

メディアアイテムとiPodライブラリについて

メディアアイテム（iPodライブラリ内の曲、オーディオブック、およびオーディオPodcast）はさまざまなメタデータを保持できます。このメタデータを使用して、クエリを組み立てたり、ユーザインターフェイスにメディアアイテムの魅力的な表示を作成したりできます。図 1-3にメディアアイテムの文字列を示します。

図 1-3　メディアアイテムとそのメタデータ

Type: MUSIC

Artwork:

Title:	Happy Birthday to You
Album title:	Happy Songs
Artist:	Happy the Clown
Genre:	Children's Music
Playback duration:	60 s
Persistent ID:	0X98ab4207d30
Rating:	★★★★
Last played date:	June 8, 2009

メディアアイテムのメタデータはすべて読み取り専用です。ただし、メディアアイテムの永続IDの値を使用することで、アプリケーションで管理する追加のメタデータを関連付けることができます。

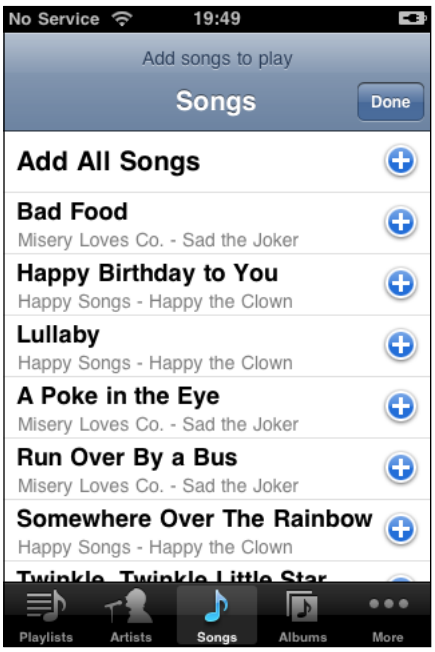
図1-3でわかるとおり、アイテムのメタデータには複数のカテゴリの情報を含めることが可能です。いわゆる**一般プロパティ**はあらゆるメディアアイテムに適用できます。これらには、「タイトル(Title)」、「アーティスト(Artist)」、「アートワーク(Artwork)」、その他多数のものが含まれます。これらのプロパティの値は通常、時間の経過とともに変更されることはありません。

メディアアイテムは、レーティング(Rating)や最終再生日(Last played date)といった**ユーザ定義プロパティ**を持つこともできます。これらのプロパティはデスクトップ上と同じようにユーザの行動に従って更新されます。

メディアアイテムピッカー

ユーザが音楽を選択できるようにする最も簡単な方法は、完全に設定済みのModal View Controllerである**メディアアイテムピッカー**を使用することです。図1-4に示すように、このユーザインターフェイスは組み込みのiPodアプリケーションの「On-The-Go」のインターフェイスに似ています。

図 1-4 メディアアイテムピッカーのユーザインターフェイス



ユーザが「完了(Done)」をタップすると、実装したデリゲートメソッドが選択されたメディアアイテムのリストを受信し、ピッカーのビューを閉じます。

重要なのは、メディアアイテムピッカーと組み込みのiPodアプリケーションが似ているのは表面上だけだということを理解しておくことです。iPodを使用してユーザは、「On-The-Go」プレイリストを作成できます。このプレイリストはその他のプレイリストと同じように動作します。たとえば、iPodアプリケーションの「プレイリスト(Playlists)」タブに表示され、ユーザが変更するか削除するまで存続します。

ピッカーを使用した場合、ユーザは代わりにメディアアイテムのコレクションを指定します。コレクションはプレイリストのステータスを保持しません。アーカイブしない限り、コレクションはユーザのアプリケーション終了後は存続しません。またどのような状況でもコレクションがピッカーの「プレイリスト(Playlists)」タブに表示されることはありません。

プログラムによるメディアアイテムの取得

メディアアイテムピッカーでは必要とする制御が得られない場合、APIのデータベースアクセスクラスを使用できます。これらのクラスは、任意の複雑なクエリを作成できるように設計されています。たとえば、特定のジャンル内でタイトルに特定の単語や語句を含むすべての曲を取得できます。

プログラムによるアクセスは次の2つの処理ステップからなります。

1. クエリを作成します。
2. 一致するメディアアイテムを取得するようクエリに命令します。

メディアクエリは、デバイスのiPodライブラリから何を取得し、それらをどのように配置すべきかを記述したものです。次の2つのプロパティを設定します。

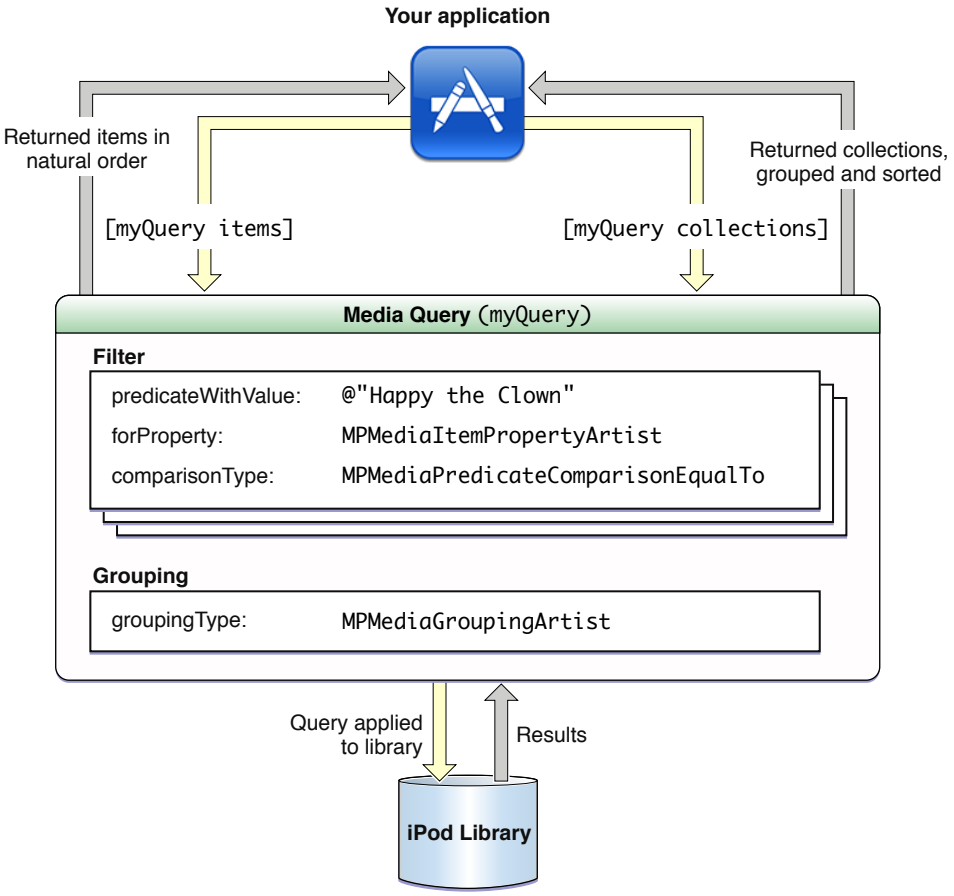
- **フィルタ(filter)**に何を取得するか記述します。フィルタはオプションで、フィルタのないクエリはiPodライブラリ全体と一致します。
- **グルーピングタイプ(grouping type)**は取得したメディアアイテムのコレクションに対して使用する、配置を指定するオプションのキーです。

もう少し詳しく見ると、フィルタはアプリケーションの要求に合わせて単純にも複雑にもできます。フィルタはメディアプロパティ述語の1つ以上のインスタンスで構成されます。**メディアプロパティ述語**は各メディアアイテムに対して評価する、論理条件文です。クエリを実行すると、フィルタを満たすアイテムがiPodライブラリから取得されます。

オプションのグルーピングタイプはコレクションの配置およびソート方法、さらに各コレクション内のメディアアイテムのソート方法を指定します。たとえば、“album”グルーピングタイプを使用すると、返されたメディアアイテムはアルバム別にグループ化され、各アルバムの曲はトラック順にソートされます。

図1-5に設定済みのメディアクエリおよびアプリケーションとiPodライブラリ間でのその位置について示します。

図 1-5 メディアクエリを使用したデバイスのiPodライブラリのアクセス



図に示すように、クエリはアイテムまたはコレクションを取得できます。

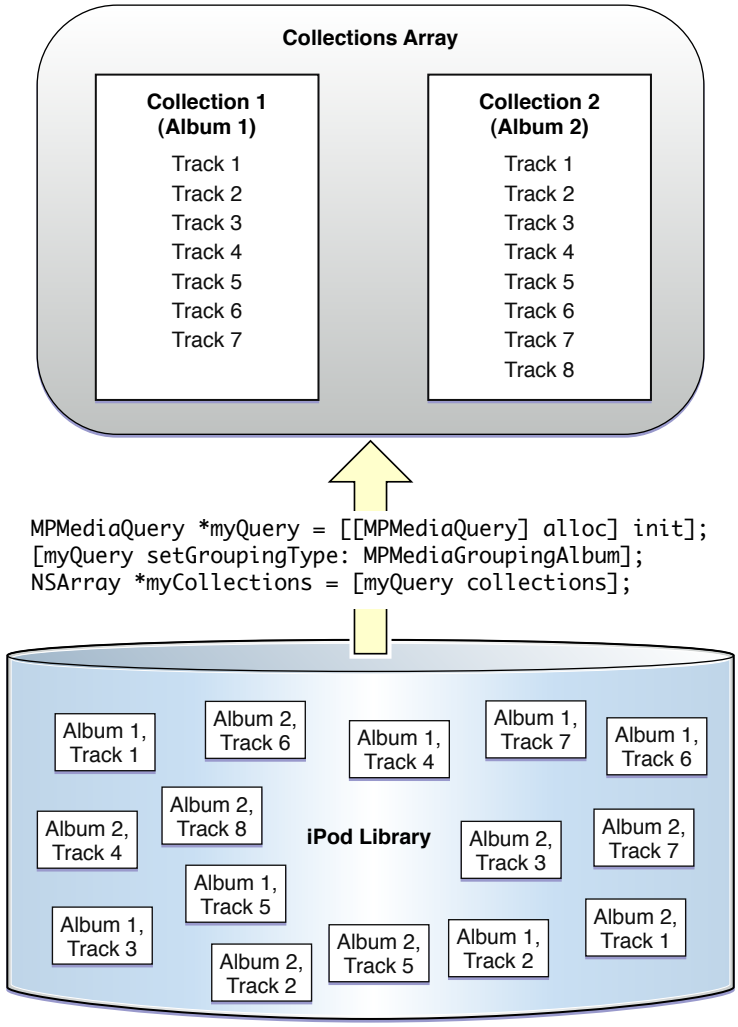
- アイテムを要求すると、クエリはフィルタに一致するすべてのアイテムを含むコレクションを返します。アイテムは「自然な」順番、つまりデスクトップ上でiTunesが表示する順番になっています。
- コレクションを要求すると、メディアクエリはフィルタだけでなくグルーピングタイプも使用します。

コレクションとプレイリストについて

メディアアイテムコレクションはメディアアイテムの配列です。メディアクエリの`collections`プロパティにアクセスすることによって、デバイスのiPodライブラリからコレクションを取得できます。戻り値はソートされたコレクションの配列で、各コレクションはクエリのグルーピングタイプのインスタンスです。

たとえば、`MPMediaGroupingAlbum`キーをメディアクエリに割り当てて、“album”グルーピングタイプを指定し、クエリにフィルタはないとします。`collections`プロパティの値には、iPodライブラリのすべてのオーディオコンテンツが、コレクションごとに1アルバムの状態で配置されます。コレクション（このケースではアルバム）はアルファベット順にソートされます。各コレクション内の曲（各メディアアイテム）はトラック番号順にソートされます。図 1-6にこれを示します。

図 1-6 デバイスのiPodライブラリからのコレクションの取得



図の下部にデバイスのiPodライブラリが表示されています。この図の場合、ライブラリには2つだけミュージックアルバムがあり、1つは7トラック、もう1つには8トラックが含まれています。アイテムは「自然な」順番、つまりデスクトップ上でiTunesが表示する順番になっています。

図の中央の上向きの矢印は、汎用（フィルタのない）クエリの定義、“album”グルーピングタイプの適用、およびクエリの`collections`プロパティへのアクセスを示しています。

図の上部は`collections`呼び出し結果を表しています。この配列は、要素がさらにメディアアイテムの配列になっています。コレクションの配列はアルバム名でソートされています。各コレクションはトラック番号でソートされています。

独自のコレクションを構築することも可能です。たとえば、ユーザがメディアアイテムピッカーを使用して作成したセレクションを管理するような場合に便利です。ただし、コレクションは可変ではないことに注意してください。

プレイリストは特別な種類のメディアアイテムコレクションです。名前や永続IDなど、通常のコレクションにはないプロパティを持っています。プレイリストはデスクトップ上でユーザによって作成され、デバイス上では読み取り専用です。

iPodライブラリ変更通知の使用

`MPMediaLibrary`オブジェクトは、デバイスのiPodライブラリの状態を表します。これを使用して、ライブラリコンテンツのキャッシュが最新であることを確認できます。これは、アプリケーションの実行中に、ユーザがデバイスを同期させてiPodライブラリのコンテンツを変更することが考えられるため便利です。

独自のサウンドとiPodライブラリサウンドのミキシング

注： アプリケーションのサウンドをデバイスのiPodライブラリのサウンドとミキシングするには、オーディオセッション、オーディオセッションカテゴリ、オーディオハードウェア経路の変化、およびオーディオ割り込みについて理解している必要があります。これらについて学習するには、『*Audio Session Programming Guide*』を参照してください。

オーディオフォーマットの選択が、デバイス上のサウンドの同時再生にどのように影響を与えるかも理解している必要があります。『*Core Audio Overview*』の「Playing Multiple Sounds Simultaneously」in *Core Audio Overview*を参照してください。

ミュージックプレーヤーは、Media Playbackオーディオセッションカテゴリを自動的に採用します。アプリケーションでミュージックプレーヤーを使用し、ほかのサウンドは使用しない場合、オーディオセッションのコードを一切書いてはいけません。具体的には、オーディオセッションを初期化、設定、またはアクティブ化すべきではありません。システムが自動的にミュージックプレーヤー用に、再生、オーディオハードウェア経路の変化、およびオーディオ割り込みを処理します。

一方で、アプリケーションのサウンドとiPodライブラリサウンドをミキシングする場合、アプリケーションのオーディオセッションを設定して使用する必要があります。ミキシングをサポートするためAmbientカテゴリを使用します。『*Audio Session Programming Guide*』に説明されているとおり、オーディオ割り込み、オーディオハードウェア経路の変化、およびオーディオセッションの再アクティブ化を処理します。アプリケーションサウンドとiPodサウンドのミキシングの例については、*AddMusic*』サンプルを参照してください。

メディアの再生

デバイスのiPodライブラリから曲、オーディオブック、およびオーディオPodcastを再生するには、`MPMusicPlayerController`クラスのインスタンスであるミュージックプレーヤーを使用します。簡単に言うと、実行するステップは次のとおりです。

1. ミュージックプレーヤーの通知を受信するための登録をし、通知を有効にする
2. ミュージックプレーヤーを作成する
3. ミュージックプレーヤーの再生キューを設定する
4. 再生オプションを設定する
5. 再生を実行する

この章では、これらのステップを実装する方法を説明します。また、音楽の再生中に既存の再生キューへの追加が行われるなど、ミュージックプレーヤーの使用時に生じるいくつかのシナリオに対する対処方法も提供します。この章のコード例の拡大版が、『*AddMusic*』サンプルアプリケーションにあります。

重要： オーディオを再生できるのは一度に1つのミュージックプレーヤーのみです。ミュージックプレーヤーは、アプリケーションのメインスレッドでのみ使用できます。

ミュージックプレーヤー通知の登録

ミュージックプレーヤーの再生制御を提供したり、再生中のアイテムの情報を表示したりするには、リスト 2-1に示すようにミュージックプレーヤー通知に登録する必要があります。

リスト 2-1 ミュージックプレーヤー通知の登録とアクティブ化

```
NSNotificationCenter *notificationCenter = [NSNotificationCenter defaultCenter];

[notificationCenter
    addObserver: self
    selector:    @selector (handle_NowPlayingItemChanged:)
    name:        MPMusicPlayerControllerNowPlayingItemDidChangeNotification
    object:      musicPlayer];

[notificationCenter
    addObserver: self
    selector:    @selector (handle_PlaybackStateChanged:)
    name:        MPMusicPlayerControllerPlaybackStateDidChangeNotification
    object:      musicPlayer];

[musicPlayer beginGeneratingPlaybackNotifications];
```

ここで示されている*selector:*パラメータに提供されるメソッドを実装します。実装例については、『*AddMusic*』サンプルを参照してください。

ミュージックプレーヤーを割り当て解除する前に、リスト 2-2に示すように、通知の登録解除と無効化を行います。

リスト 2-2　ミュージックプレーヤー通知の登録解除と非アクティブ化

```
[[NSNotificationCenter defaultCenter]
    removeObserver: self
    name:           MPMusicPlayerControllerNowPlayingItemDidChangeNotification
    object:         musicPlayer];

[[NSNotificationCenter defaultCenter]
    removeObserver: self
    name:           MPMusicPlayerControllerPlaybackStateDidChangeNotification
    object:         musicPlayer];

[musicPlayer endGeneratingPlaybackNotifications];
```

ミュージックプレーヤーの作成と設定

アプリケーションミュージックプレーヤーを作成するには、リスト 2-3の1行目に示すように、単に`applicationMusicPlayer`クラスメソッドを呼び出します。

リスト 2-3　アプリケーションミュージックプレーヤーの作成

```
MPMusicPlayerController* appMusicPlayer =
    [MPMusicPlayerController applicationMusicPlayer];

[appMusicPlayer setShuffleMode: MPMusicShuffleModeOff];
[appMusicPlayer setRepeatMode: MPMusicRepeatModeNone];
```

リストに示すように、シャッフルモードとリピートモードも設定し、iPodアプリケーションのモード（標準設定）を引き継がないようにしたい場合もあるかもしれません。

各種のシャッフルモードおよびリピートモードについては『*MPMusicPlayerController Class Reference*』に解説されています。

iPodミュージックプレーヤーの作成には追加のステップがあります。その理由は、組み込みiPodアプリケーションの状態にアクセスでき、アプリケーションの起動時に再生中のアイテムが存在する可能性があるためです。リスト 2-4に、iPodアプリケーションがデバイスのiPodライブラリからアイテムを再生しているときに、再生中のアイテムを確認する方法を示します。

リスト 2-4　iPodミュージックプレーヤーの作成

```
MPMusicPlayerController* iPodMusicPlayer =
    [MPMusicPlayerController iPodMusicPlayer];

if ([iPodMusicPlayer nowPlayingItem]) {
    // UI（アートワーク、曲名、音量表示など）を更新し
    //        iPodの状態を反映する
}
```

リストに示すように、アプリケーションは再生中のアイテムに基いてユーザインターフェイスと状態を更新することができます。

重要： iPodアプリケーションがホームシェアリングを使用して共有されている曲を再生している場合、iPodミュージックプレーヤーの`nowPlayingItem` (? ページ) プロパティの値は`nil`になります。ただし、iPodミュージックプレーヤーの`playbackState`プロパティは、常にiPodアプリケーションの実際の再生状態を反映します。

再生キューの設定

ミュージックプレーヤー用の再生キューを設定するには主に次の2通りの方法があります。

- メディアアイテムコレクションを使用する
- コレクションを暗黙に定義する、メディアクエリを使用する

データベースアクセスクラスを使用するか、メディアアイテムピッカーを使用してコレクションを取得します。どちらの方法でも、ミュージックプレーヤーにコレクションを適用するには次のメソッドを呼び出します。

```
[musicPlayer setQueueWithItemCollection: userMediaItemCollection];
```

または、メディアクエリを使用してキューを設定し、次に示すように暗黙的にコレクションを定義します。

```
[musicPlayer setQueueWithQuery: [MPMediaQuery songsQuery]];
```

もちろん、`setQueueWithQuery:`メソッドの引数としてより複雑なクエリを指定することもできます。クエリの詳細については、「iPodライブラリの使用」 (25 ページ) を参照してください。

リスト 2-5は、プレーヤーのキューを新しいコレクションで置き換えるのか、または追加するのかといった、再生の状態を考慮した実践的な再生キューの設定を示しています。

リスト 2-5 コンテキスト内の再生キューの設定

```
- (void) updateQueueWithCollection: (MPMediaItemCollection *) collection {

    // コレクションをミュージックプレーヤーの再生キューに追加する、ただし
    //    ユーザが再生する曲を少なくとも1つ選んだ場合に限る
    if (collection) {

        // まだ再生キューが存在しない場合
        if (userMediaItemCollection == nil) {
            [self setUserMediaItemCollection: collection];
            [musicPlayer setQueueWithItemCollection: userMediaItemCollection];
            [musicPlayer play];

        // ミュージックプレーヤーの状態を取得し、
        //    再生キューの更新後に復元できるようにする
        } else {
            BOOL wasPlaying = NO;
            if (musicPlayer.playbackState == MPMusicPlaybackStatePlaying) {
```

```
            wasPlaying = YES;
        }

        // 再生中のアイテムと現在の演奏時間を保存する
        MPMediaItem *nowPlayingItem        = musicPlayer.nowPlayingItem;
      NSTimeInterval currentPlaybackTime = musicPlayer.currentPlaybackTime;

        // 前から存在するメディアアイテムコレクションを
        //    新しいものと組み合わせる
        NSMutableArray *combinedMediaItems =
            [[userMediaItemCollection items] mutableCopy];
        NSArray *newMediaItems = [mediaItemCollection items];
        [combinedMediaItems addObjectsFromArray: newMediaItems];

        [self setUserMediaItemCollection:
            [MPMediaItemCollection collectionWithItems:
                (NSArray *) combinedMediaItems]];

        [musicPlayer setQueueWithItemCollection: userMediaItemCollection];

        // 再生中のアイテムと現在の演奏時間を復元する
        musicPlayer.nowPlayingItem      = nowPlayingItem;
        musicPlayer.currentPlaybackTime = currentPlaybackTime;

        if (wasPlaying) {
            [musicPlayer play];
        }
    }
}
```

このメソッドは既存の再生キューの有無に応じて分岐し、再生キューに追加する際に再生状態が確実に保存されるようにします。

再生の制御

『*MPMusicPlayerController Class Reference*』に説明されているとおり、ミュージックプレーヤーの再生制御は簡単です。再生、一時停止、停止、早送りと巻き戻し、そして前後への頭出しが可能です。

さらに、`currentPlaybackTime`プロパティは読み書きが行えます。これを使用して再生中のアイテムのタイムライン内に再生ポイントを設定できます。

たとえば、アプリケーションインターフェイスに`UISlider`オブジェクトを提供すると、ユーザはメディアアイテムのタイムラインに再生ポイントを設定できます。次の例では、そのようなスライダが`timelineSlider`インスタンス変数に関連付けられていると想定しています。

```
- (IBAction) setTimelinePosition: (id) sender {
    [musicPlayer setCurrentPlaybackTime: [timelineSlider value]];
}
```

メディアアイテムピッカーの使用

メディアアイテムピッカーは、ユーザがデバイスのiPodライブラリからメディアアイテムを選択できるようにするためのパッケージ済みView Controllerです。ピッカーの使用はとても簡単です。

1. Controllerオブジェクトをピッカーのデリゲートとして指定します。
2. Controllerからピッカーを起動します。
3. ユーザが終了を知らせると、デリゲートは選択されたメディアアイテムのコレクションを受信し、ピッカーを閉じます。

メディアアイテムピッカーのデリゲートの設定

Controllerオブジェクトをメディアアイテムピッカーのデリゲートとして設定するには、まず次のようにControllerのヘッダファイルのインターフェイス宣言にプロトコルの名前を追加します。

```
@interface myController : UIViewController <MPMediaPickerControllerDelegate> {
    // インターフェイス宣言
}
```

次に、そのプロトコルから2つのデリゲートメソッドを実装します。リスト 3-1に示すように、最初のメソッドはいくつかのメディアアイテムを選択したユーザに応答します。ピッカーを解放し、Controllerの再生キュー更新メソッドを実行します。

リスト 3-1　ピッカーからのメディアアイテムの新規コレクションへの応答

```
- (void) mediaPicker: (MPMediaPickerController *) mediaPicker
           didPickMediaItems: (MPMediaItemCollection *) collection {

    [self dismissModalViewControllerAnimated: YES];
    [self updatePlayerQueueWithMediaCollection: collection];
}
```

再生キューを更新する方法を示したコードの例は、リスト 2-5 （21 ページ）を参照してください。

2つ目のピッカーデリゲートメソッドは、ユーザが再生するアイテムを1つも選択せずに「完了(Done)」をタップした場合の処理を行います。基本的な実装については、リスト 3-2を参照してください。

リスト 3-2　ユーザがピッカーをキャンセルした場合の応答

```
- (void) mediaPickerDidCancel: (MPMediaPickerController *) mediaPicker {

    [self dismissModalViewControllerAnimated: YES];
}
```

メディアアイテムピッカーの表示

リスト 3-3は、Controllerオブジェクトをピッカーのデリゲートとして確立する方法を含め、メディアアイテムピッカーを設定し表示する方法を示します。

リスト 3-3　メディアアイテムピッカーの表示

```
MPMediaPickerController *picker =
    [[MPMediaPickerController alloc]
        initWithMediaTypes: MPMediaTypeAnyAudio];                   // 1

[picker setDelegate: self];                                         // 2
[picker setAllowsPickingMultipleItems: YES];                        // 3
picker.prompt =
    NSLocalizedString (@"Add songs to play",
                       "Prompt in media item picker");

[myController presentModalViewController: picker animated: YES];    // 4
[picker release];
```

このコードは次のように動作します。

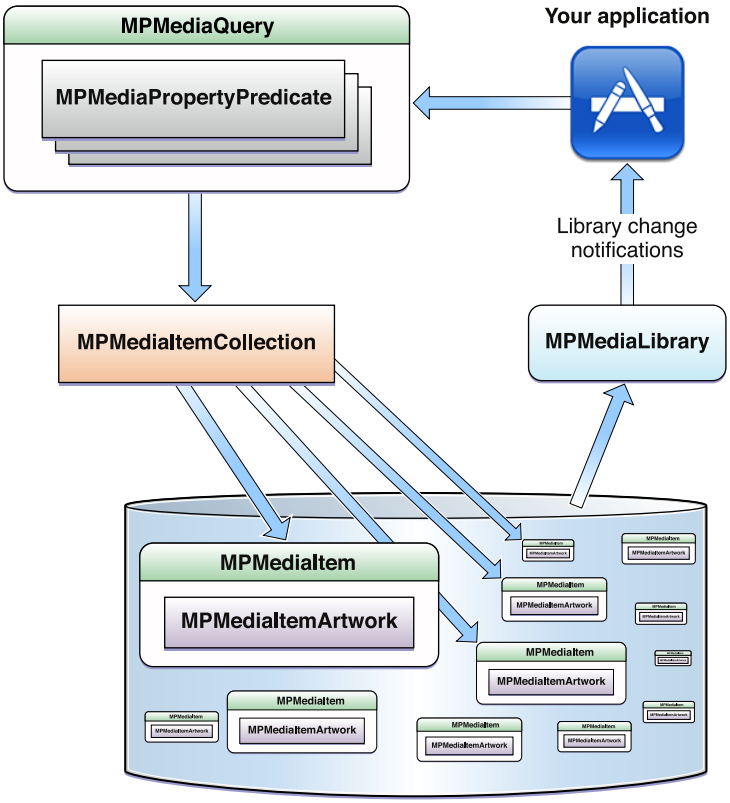
1. メディアアイテムピッカーを作成します。パラメータは、表示するメディアアイテムの種類を示します。オプションについては、`Media Item Type Flags`列挙を参照してください。
2. Controllerオブジェクトをデリゲートとして確立します。
3. ユーザは複数のアイテムを選択できることを指定します。このステートメントを除外することで、代わりにシングルアイテムピッカーを表示することもできます。デフォルトの動作は複数の選択を許可しません。
4. ピッカーを表示します。`myController`オブジェクトがそれを保持するため、次にそれを解放して`alloc`呼び出しのバランスを保ちます。

iPodライブラリの使用

メディアアイテムの管理および選択に、メディアアイテムピッカーで得られる以上の制御がアプリケーションに必要な場合があります。デバイスのiPodライブラリとのカスタムユーザインターフェイスを提供したり、特定のクエリを実行したり、メディアアイテムにカスタムメタデータを関連付けたりする場合、APIのデータベースアクセスクラスが必要です。

図 4-1に、アプリケーションとデータベースアクセスクラスがメディアアイテムの取得時にどのようにやり取りするのかを示します。

図 4-1 iPodライブラリデータベースアクセスクラスの使用



最初に、少し時間をとってこの図と図 1-5 （16 ページ）の類似性に注目してください。前の図はクエリが何であるかを説明するのに役立ちました。この図ではメディアクエリクラスを集団の中に配置し、すべてのデータベースアクセスクラスがお互いにどのように関連しているかを表しています。

図では、アプリケーションとデバイスのiPodライブラリ間のやり取りの2つのシナリオを示しています。最初に、アプリケーション（“Your application”）から反時計周りに進むと、図ではクエリの作成と設定が描かれ、それがメディアアイテムコレクションを定義していることが描かれています。コ

レクションはライブラリ内の特定のアイテムのサブセットをポイントしています。図には示されていませんが、コレクションはクエリを実行したときの戻り値です。図に示すように、各メディアアイテムは複数のプロパティの中にメディアアイテムアートワークオブジェクトを持っています。

図4-1の2つ目のシナリオでは、`MPMediaLibrary`クラスを使用してアプリケーションがiPodライブラリから変更通知を受信しています。これらの変更通知を受信するように登録することにより、ユーザがアプリケーションの実行中にデバイスを同期させた場合、アプリケーションではライブラリコンテンツのキャッシュを更新することができます。

グループ化されていないメディアアイテムの取得

デバイスのiPodライブラリからメディアアイテムを取得するには、メディアクエリの作成から始めます。最も単純なクエリは、リスト 4-1に示す`everything`汎用クエリで、ライブラリのすべてのコンテンツを照合します。例では次に、メディアアイテムのタイトルをXcodeデバッガコンソールにログ出力し、クエリを実行して一致するメディアアイテムを取得する`items`プロパティの使用方法を示しています。

リスト 4-1　汎用メディアクエリの作成と使用

```
MPMediaQuery *everything = [[MPMediaQuery alloc] init];

NSLog(@"Logging items from a generic query...");
NSArray *itemsFromGenericQuery = [everything items];
for (MPMediaItem *song in itemsFromGenericQuery) {
    NSString *songTitle = [song valueForProperty: MPMediaItemPropertyTitle];
    NSLog (@"%@", songTitle);
}
```

注： この文書のすべてのコード例と同様に、この例は設定済みのiPhoneデバイス上でのみ正常に実行し、iPhone Simulatorでは使用できません。

より具体的なクエリを作成するには、汎用クエリに1つ以上のメディアプロパティ述語を適用します。述語は、メディアアイテムに対して判定する1つの論理条件を指定します。

リスト 4-2は、メディアアイテムの「artist（アーティスト）」プロパティが特定の値でなければならないという条件を含んだ述語を作成しています。次に述語をクエリに加えています。

リスト 4-2　メディアプロパティ述語の作成と適用

```
MPMediaPropertyPredicate *artistNamePredicate =
    [MPMediaPropertyPredicate predicateWithValue: @"Happy the Clown"
                                     forProperty: MPMediaItemPropertyArtist];

MPMediaQuery *myArtistQuery = [[MPMediaQuery alloc] init];
[myArtistQuery addFilterPredicate: artistNamePredicate];

NSArray *itemsFromArtistQuery = [myArtistQuery items];
```

このコードを実行すると、`itemsFromArtistQuery`配列にiPodライブラリ内の「Happy the Clown」によるアイテムのみが格納されます。

クエリが一致するものを絞り込むために、複数の述語を作成してクエリに加えることができます。リスト 4-3に、2つの述語を使用したこのテクニックを示します。

リスト 4-3　既存のメディアクエリへの複数述語の適用

```
MPMediaPropertyPredicate *artistNamePredicate =
    [MPMediaPropertyPredicate predicateWithValue: @"Sad the Joker"
                                     forProperty: MPMediaItemPropertyArtist];

MPMediaPropertyPredicate *albumNamePredicate =
    [MPMediaPropertyPredicate predicateWithValue: @"Stair Tumbling"
                                    forProperty: MPMediaItemPropertyAlbumTitle];

MPMediaQuery *myComplexQuery = [[MPMediaQuery alloc] init];

[myComplexQuery addFilterPredicate: artistNamePredicate];
[myComplexQuery addFilterPredicate: albumNamePredicate];
```

各述語は、任意の選択（アーティスト名など）の値を適切なメディアアイテムプロパティキーと共に使用して作成します。これらのキーについては、`General Media Item Property Keys`および`Podcast Item Property Keys`の『*MPMediaItem Class Reference*』で説明しています。

クエリに複数の述語を適用すると、クエリはそれらを論理演算子ANDを使用して組み合わせます。

> **重要：** 特定のメディアアイテムプロパティに複数の述語を適用しないでください。たとえば、クエリに2つの`MPMediaItemPropertyArtist`述語を指定してはいけません。指定した場合、そのクエリの使用時の動作結果は不確定です。

リスト 4-4に示すように、クエリの初期化時に述語を追加することもできます（この例の2つの述語はすでに定義済みという想定です）。

リスト 4-4　メディアクエリの初期化時に複数の述語を適用

```
NSSet *predicates =
    [NSSet setWithObjects: artistNamePredicate, albumNamePredicate, nil];

MPMediaQuery *specificQuery =
    [[MPMediaQuery alloc] initWithFilterPredicates: predicates];
```

リスト 4-4は、2つの述語を含む`NSSet`オブジェクトを最初に定義し、次に`initWithFilterPredicates:`クラスメソッドを使用してメディアクエリを割り当てて、初期化しています。

特定のプロパティキーのみが正しい述語の作成に使用できます。これらのキーは、『*MPMediaItem Class Reference*』で「filterable」と記されています。無効な述語を含むクエリを使用しようとした場合、結果として生じる動作は不確定です。

重要： システムは、述語と共に使用しても意味のないプロパティキーを使用した述語の作成を許可します。そのような述語をクエリに適用しようとした場合、結果として生じる動作は不確定です。述語の問題を防ぐには以下の点に従います。

- 『*MPMediaItem Class Reference*』をよく読みます。
- `canFilterByProperty:`メソッドを使ってコードを記述し、メディアアイテムプロパティキーをその使用前に検証します。
- ユーザにメディアプロパティ述語を作成させないようにします。
- コードが想定通りに動作することをさまざまな条件下で十分にテストして確認します。

リスト 4-5に、述語で使用する前にメディアアイテムプロパティを検証する`canFilterByProperty:`クラスメソッドの使用方法を示します。

リスト 4-5 プロパティキーがメディアプロパティ述語に使用できるかのテスト

```
if ([MPMediaItem canFilterByProperty: MPMediaItemPropertyGenre]) {

    MPMediaPropertyPredicate *rockPredicate =
        [MPMediaPropertyPredicate predicateWithValue: @"Rock"
                                         forProperty: MPMediaItemPropertyGenre];

    [query addFilterPredicate: rockPredicate];
}
```

リスト 4-5では、示されたメディアアイテムプロパティキー（この例では、`MPMediaItemPropertyGenre`）が正しい述語を作成できる場合のみ、if文の本体が実行されます。Appleは、メディアプロパティ述語を作成する前に必ずこのようなチェックを行うことをお勧めします。

メディアアイテムコレクションの取得

メディアクエリは、グループ化されていないメディアアイテムを取得する場合にのみ役立ちます。メディアクエリを使用して、ソートおよび整理されたメディアアイテムの*コレクション*を取得することもできます。どのように整理されるかは、メディアクエリの`grouping`プロパティに設定する値に依存します。

リスト 4-6に、特定のアーティストのすべての曲を、アルバムごとに整理して取得する方法を示します。例では結果をXcodeデバッガコンソールにログ出力します。

リスト 4-6 グルーピングタイプを使用したメディアアイテムコレクションの指定

```
MPMediaQuery *query = [[MPMediaQuery alloc] init];

[query addFilterPredicate: [MPMediaPropertyPredicate
                               predicateWithValue: @"Moribund the Squirrel"
                                      forProperty: MPMediaItemPropertyArtist]];
// メディアクエリのグルーピングタイプを設定
[query setGroupingType: MPMediaGroupingAlbum];
```

```
NSArray *albums = [query collections];
for (MPMediaItemCollection *album in albums) {
    MPMediaItem *representativeItem = [album representativeItem];
    NSString *artistName =
        [representativeItem valueForProperty: MPMediaItemPropertyArtist];
    NSString *albumName =
        [representativeItem valueForProperty: MPMediaItemPropertyAlbumTitle];
    NSLog (@"%@ by %@", albumName, artistName);

    NSArray *songs = [album items];
    for (MPMediaItem *song in songs) {
        NSString *songTitle =
            [song valueForProperty: MPMediaItemPropertyTitle];
        NSLog (@"\t\t%@", songTitle);
    }
}
```

リスト4-6では、メディアクエリのcollectionsプロパティの値が配列の配列であることがわかります。外側のforループは指定したアーティストによるアルバムごとに繰り返しています。内側のforループは現在のアルバム内の曲ごとに繰り返しています。

MPMediaQueryクラスには、グルーピングタイプが事前設定済みのクエリを作成するためのいくつかの便利なコンストラクタが含まれています。たとえば、次のステートメントは"album"グルーピングタイプを新しく割り当てたクエリにアタッチしています。

```
MPMediaQuery *query = [MPMediaQuery albumsQuery];
```

すべての簡易コンストラクタについての説明は、『*MPMediaQuery Class Reference*』を参照してください。

メディアアイテムアートワークの使用

メディアアイテムの最も便利でインパクトの高いプロパティの1つがアートワークです。アートワークを表示するには、Interface BuilderおよびXcodeを使用します。その手順は以下のとおりです。

1. UIImageViewオブジェクトをInterface Builderのビューレイアウトに追加します。
2. IBOutletインスタンス変数をView Controllerクラスに追加して、UIImageViewオブジェクトに接続します。
3. アートワークを所持するメディアアイテムからアートワークを取得します（この章で説明したとおり、メディアアイテムはすでに取得済みです）。
4. アートワークをUIImageオブジェクトに変換し、それをレイアウトのUIImageViewオブジェクトに割り当てます。

リスト 4-7に、ステップ3および4の実行方法を示します。

リスト 4-7　メディアアイテムのアルバムアートワークの表示

```
MPMediaItemArtwork *artwork =
    [mediaItem valueForProperty: MPMediaItemPropertyArtwork];
```

```
UIImage *artworkImage =
    [artwork imageWithSize: albumImageView.bounds.size];

if (artworkImage) {
    albumImageView.image = artworkImage;
} else {
    albumImageView.image = [UIImage imageNamed: @"noArtwork.png"];
}
```

リスト 4-7の最後の行で使用したnoArtwork.pngファイルは、メディアアイテムに関連付けられたアートワークがない場合にXcodeプロジェクトに追加する代替画像です。

書類の改訂履歴

この表は「*iPod*ライブラリアクセスプログラミングガイド」の改訂履歴です。

日付	メモ
2011-02-21	「ホームシェアリングとiPodミュージックプレーヤー」（13 ページ）と「ミュージックプレーヤーの作成と設定」（20 ページ）のセクションで、共有アイテムの再生について説明しています。
2010-11-30	ミュージックプレーヤーはアプリケーションのメインスレッドでのみ使用できることを追記しました。
2010-05-21	細かな変更を行いました。
2009-11-18	Media Playerフレームワークを使用してiOS内のメディアアイテムを検索して再生する方法を説明する新規文書。

書類の改訂履歴

用語解説

アプリケーションミュージックプレーヤー(application music player) iPodアプリケーションの状態に影響を与えることなくメディアアイテムを再生するオブジェクト。iPodミュージックプレーヤー(iPod music player)、ミュージックプレーヤー(music player)も参照。

デフォルトメディアライブラリ(default media library) アプリケーションに対してライブラリ変更通知を提供するオブジェクト。

filterable 有効な検索述語(predicate)を作成するために使用できるメディアアイテムプロパティキーについて記述します。

フィルタ(filtering) 1つ以上のメディアプロパティ述語(media property predicate)オブジェクトに基いた検索を使用することにより、iPodライブラリ(iPod library)からのメディアアイテムのサブセットの取得を記述します。

Geniusプレイリスト(Genius Playlist) ミュージックの特徴に基づいたアルゴリズムで作成されたミュージックメディアアイテムの並び。On-The-Goプレイリスト(on-the-go playlist)、Smartプレイリスト(Smart Playlist)も参照。

iPodライブラリ(iPod library) ユーザがデスクトップのiTunesから同期させたデバイス上のメディアアイテムの集まり。

iPodミュージックプレーヤー(iPod music player) 組み込みのiPodアプリケーションを制御して、その状態を共有しながらメディアアイテムを再生するオブジェクト。アプリケーションミュージックプレーヤー(application music player)、ミュージックプレーヤー(music player)も参照。

メディアアイテム(media item) iPodライブラリ(iPod library)内の個々のメディア（1つの曲など）。メディアアイテムには、オーディオ、オーディオブック、およびPodcastを含むさまざまな種類があります。

メディアアイテムアートワーク(media item artwork) メディアアイテム(media item)に関連付けられたグラフィックイメージ（アルバムのカバーアートワークなど）を表すオブジェクト。

メディアアイテムコレクション(media item collection) iPodライブラリ(iPod library)内の関連したメディアアイテムの集まり。

メディアアイテムピッカー(media item picker) メディアアイテムを選択するグラフィカルインターフェイスを提供するために使用する、特別なView Controller。

メディアプロパティ述語(media property predicate) iPodライブラリ(iPod library)から指定したメディアアイテムのサブセットを取得するためのメディアクエリ(media query)内のフィルタ。

メディアクエリ(media query) iPodライブラリから何を取得し、それらをどのように配置すべきかの記述が含まれているオブジェクト。

ミュージックプレーヤー(music player) メディアアイテムを再生するオブジェクト。アプリケーションミュージックプレーヤー(application music player)、iPodミュージックプレーヤー(iPod music player)も参照。

On-The-Goプレイリスト(on-the-go playlist) ノートブックやデスクトップコンピュータ上ではなく、デバイス上でユーザが作成したメディアアイテムの並び。Geniusプレイリスト(Genius Playlist)、Smartプレイリスト(Smart Playlist)も参照。

再生キュー(playback queue) ミュージックプレーヤー(music player)で再生するためのメディアアイテムのリスト。

再生状態(playback state) iPodライブラリアクセスのためにミュージックプレーヤーの再生状態を記述したもの。再生中、停止、一時停止、早送り、または巻き戻し。

述語(predicate) iPodライブラリアクセスにおける、iPodライブラリ(iPod library)を検索した際にメディアアイテムに対して判定する1つの論理条件の指定。

リピートモード(repeat mode) iPodライブラリアクセスの場合、現在の再生キューを繰り返すか、または1回だけ再生するかを表す。

シャッフルモード(shuffle mode) iPodライブラリアクセスの場合、現在の再生キューをユーザ定義の順序で再生するか、シャッフルされた順番で再生するかを表す。

Smartプレイリスト(Smart Playlist) ユーザが指定した条件に従って自動的に作成されたメディアアイテムの並び。Geniusプレイリスト(Genius Playlist)、On-The-Goプレイリスト(on-the-go playlist)も参照。