ファイルメタデータ検索プログラミングガイド

Carbon > File Management

2011-09-28

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

表、リスト

ファイルメタデータのクエリについて

ファイルのメタデータを扱うAPI（Application Programming Interface）を利用すれば、ファイルやファイルシステムの一部として管理されているデータに基づき、ファイルを検索できます。APIはいくつかありますが、どれを選択するべきかは、多くの場合、検索結果に対してどのような処理が必要か、によって決まります。

Mac OS X用アプリケーションにメタデータの操作機能を組み込む簡便な方法として、Spotlight検索ウインドウがあります。このAPIを利用すると、アプリケーション上に標準的なSpotlight検索ウインドウを表示でき、任意の検索文字列を与えることも可能です。検索結果はそのまま表示されるだけで、アプリケーションが利用することはできません。これは、検索主体のアプリケーションではないけれども、Spotlightを使って関連用語を検索できるようにしたい、という場合に適しています。

クエリを作成し、検索結果をアプリケーション側で利用したい、という場合を考慮して、2つのAPIが用意されています。Metadataフレームワークには低レベルのクエリAPIである`MDQuery`が附属しており、メタデータの値に基づきファイルを検索できるようになっています。`MDQuery`はさまざまな設定が可能で、同期、非同期のどちらのクエリも実行できるほか、（再検索により）結果を更新する頻度もきめ細かく制御できます。

CocoaフレームワークのNSMetadataQueryクラスには、`MDQuery` APIを利用する、高レベルのObjective-Cインターフェイスが組み込まれています。このクラスを使えば、`NSPredicate`クラスの機能の一部を使ってクエリを構築し、非同期に実行できます。また、`NSMetadataQuery`クラスは、検索結果をサブカテゴリに分類する旨の指定も可能です。ただし、同期クエリの機能はなく、データの収集を終えた時点で、最小限の更新通知をアプリケーション側に送るようになっています。Mac OS Xでは、`NSMetadataQuery`はCocoaバインディングに対応しているので、大量のグルーコードを記述しなくても結果を表示できます。

注： iOSにはObjective-Cのメタデータ検索インターフェイスしかありません。Cの手続き的なインターフェイスは、Mac OS Xでのみ使えます。

対象読者

SpotlightはMac OS Xの基本機能なので、開発者は誰でも、その能力を知っておくべきでしょう。少なくとも、Spotlight検索ウインドウを使って文字列検索をする機能は、どのようなアプリケーションにも組み込んでください。

この書類の構成

以下の各節では、Spotlightを使ってメタデータを検索する方法を理解できるよう、重要な概念を説明します。

- 「NSMetadataQueryを利用したファイルメタデータの検索」（11 ページ）では、ファイルのメタデータを用いたファイル検索の概念を概観します。
- 「FinderのSpotlight検索ウインドウの表示」（21 ページ）では、標準的なSpotlight検索ウインドウを表示する手順を示します。
- 「ファイルメタデータのクエリ式の構文」（17 ページ）では、メタデータクエリ言語について解説します。

関連項目

この資料では詳しく説明していませんが、メタデータをアプリケーションで利用するための基本技術はほかにもあります。詳しくは次の資料を参照してください。

- 『*Spotlight Overview*』ではSpotlightのメタデータ活用にまつわる概念的な詳細を解説しています。
- 『*Spotlight Importer Programming Guide*』には、文書ファイルからメタデータを抽出するプラグインについての説明があります。
- 『*File Metadata Attributes Reference*』ではAppleが提供するメタデータ属性について解説します。

Spotlightクエリの生成方法として、次のコード例も参照してください。

- 『*Spotlighter*』はSpotlight検索の使い方を表す例です。
- 『*PredicateEditorSample*』は規則エディタとSpotlightの使い方を表す例です。
- 『*PhotoSearch*』は、名前をもとに画像を検索する例です。複数の検索処理を同時に実行できます。

iCloud上、デスクトップ上のファイル検索

Spotlight検索機能は、iOS v5.0とMac OS Xのどちらでも利用できます。iOSの場合はiCloud上、一方Mac OS Xの場合は、iCloud、デスクトップ、ネットワーク上のファイルを、検索クエリを用いて検索できます。

Mac OS Xにおける検索機能

Spotlightは、Mac OS Xではさまざまな箇所で使われています。Finderにも組み込まれていて、ファイル検索に利用できます。MailはSpotlightを使ってメールメッセージを検索できるようになっています。それ以外のアプリケーションも、ユーザのホームディレクトリ、ローカルファイルシステム、マウントされたネットワーク上のファイルシステム、iCloudを検索できます。

Mac OS XにはSpotlight APIを呼び出すObjective-Cのインターフェイスが用意されています。検索機能をまとめた`NSMetadataQuery`クラス、ファイルのメタデータを調べる`NSMetadataItem`クラスがあるほか、`NSMetadataQuery`クラスを`NSMetadataQueryAttributeValueTuple`クラスと併用すれば、より複雑な条件を指定して検索できます。さらに、Objective-Cのクラスにそれぞれ対応する、Core FoundationのAPIもあります（実際にはクラスがCore Foundationという基盤の上に構築されています）。

注： ファイルのメタデータは非常に役に立ちます。ファイル属性に関するさまざまな情報（ビデオのフレームサイズ、音声ファイルの圧縮方式、Finderの注釈、ファイルの作成者など）を、ファイルの内容を読み込むことなく取得できるのです。なお、ユーザは、ボリューム単位またはフォルダ単位で、Spotlightを無効にすることができます。

iOSにおける検索機能

iOSでは、iCloud上のメタデータ検索により、あるファイルに付随するファイルを検索できるようになっています。ファイルのメタデータを問い合わせるObjective-Cのインターフェイスは`NSMetadataQuery`と`NSMetadataItem`だけです。これは検索範囲がiCloud上だけだからです。

デスクトップ上の検索と違い、iOSアプリケーションのサンドボックス内を、メタデータクラスを使って検索することはできません。この中を検索するためには、サンドボックスのファイルシステム内を、`NSFileManager`クラスで再帰的にたどっていく必要があります。条件に合致するファイルが見つかったら、必要に応じたやり方でファイルにアクセスできます。当該ファイルのメタデータは、`NSMedatataItem`クラスを使って調べることもできます。

NSMetadataQueryを利用したファイルメタデータの検索

アプリケーションがSpotlightメタデータを検索するためには、Foundationフレームワークが提供するNSMetadataQueryクラスを使って、クエリを作る必要があります。クエリの動作モードには、非同期とライブ更新つき非同期の2通りがあります。非同期モードの場合は、検索開始時点で存在するファイルを対象として普通に検索します。ライブ更新つき非同期モードであれば、検索を続けながら、検索条件を満たす/満たさないファイルが入れ替わっていれば随時データを更新します。

非同期のメタデータクエリは、大きく4つの段階に分けて実行します。

- 検索条件の定義と初期化。
- 検索の開始。
- バッチ通知の監視。
- 完了通知の監視。
- クエリの停止。
- 結果の処理。

ライブ非同期モードのSpotlightクエリを利用すれば、「処理進行中に」状態が変わっていないか、所定の範囲を監視することができます。コード上の違いは、結果が得られてもクエリを停止せず、一時停止するだけにして、検索結果の処理が終わったら検索を再開する、という点だけです。

静的なファイルメタデータ検索の実行

静的なSpotlight検索とは、単純に検索を実行し、結果を返して終了する処理のことです。一度検索するだけで、状態が変わっていないか監視することはしません（これはライブ検索により実行可能、「ライブ検索の実行」を参照）。

検索条件の定義

クエリ作成手順の最初は、求める結果を得るための検索式を定義することです。`MDMetadataQuery`を使って検索する場合は、検索式の述語を、「ファイルメタデータのクエリ式の構文」（17 ページ）に示す構文で記述します。検索結果のデータが返され、検索が終了した時点で通知するよう、登録する必要があります。また、検索範囲、整列順序、デリゲートも登録できます。

1. `NSMetadataQuery`のインスタンスを作成します。
2. `NSMetadataQueryDidUpdateNotification`通知を受信するよう登録します。この通知は、検索結果が得られた時点で送信されます。

この通知は、検索結果によっては生成されないかも知れません。

3. NSMetadataQueryDidFinishGatheringNotification通知を受信するよう登録します。これは初回の検索が終了した時点で送信されます。

クエリ検索の設定

検索述語は、「ファイルメタデータのクエリ式の構文」（17 ページ）に示す構文で記述します。検索対象とすることができるフィールドは、『*File Metadata Attributes Reference*』に定義されています。属性リファレンスには、当該属性を検索するために指定できる、メタデータキーやデータ型（文字列、数値、文字列の配列、日付、UTI（Uniform Type Identifier）など）が載っています。

- 適当なSpotlightクエリ式を指定して、NSPredicateのインスタンスを生成します。

整列順序の設定

NSMetadataQueryを使う場合、整列記述子の配列を与えることにより、結果の整列順序を指定できます。整列は、返された各NSMetadataItemオブジェクトのメタデータ属性キーに基づいて行います。

- 整列に用いるメタデータキー（後述の例ではkMDItemDisplayName）を指定してNSSortDescriptorを生成します。

検索範囲の制限

どの範囲から検索結果を収集するか、検索範囲を指定して制限できます。検索範囲は、場所を示す定義済み定数、URL、ディレクトリパスの配列の形で、クエリに指定します。このうち定義済み定数は、ユーザのホームディレクトリ、ローカルにマウントされたボリュームとユーザのホームディレクトリ、遠隔マウントされたボリュームなどに制限する場合に便利です。

検索範囲は、メタデータクエリがどの範囲のファイルを検索するか、を指定するものです。表 2-1 に、検索範囲を表す定数を示します。

表 2-1　検索範囲を指定するための定数

検索範囲を表す定数	対応しているオペレーティングシステム	説明
NSMetadataQuery-UbiquitousDocuments-Scope	iOSおよびMac OS X	アプリケーションのiCloudコンテナディレクトリ以下、「Documents」ディレクトリ内の全ファイル。
NSMetadataQuery-UbiquitousDataScope	iOSおよびMac OS X	アプリケーションのiCloudコンテナディレクトリ以下、「Documents」ディレクトリ以外の全ファイル。

検索範囲を表す定数	対応しているオペレーティングシステム	説明
`NSMetadataQueryNetworkScope`	Mac OS X	ユーザがマウントした遠隔ボリューム全体。
`NSMetadataQueryLocalComputerScope`	Mac OS X	ローカルにマウントしたボリューム全体。ユーザのホームディレクトリを含み、これは遠隔ボリュームであっても検索対象。
`NSMetadataQueryUserHomeScope`	Mac OS X	ユーザのホームディレクトリ。

検索範囲は、上記の定数の配列として指定します。

- `metadataSearch`オブジェクトに、範囲を表す定数の配列を指定して、`setSearchScopes:`メッセージを送ります。

  後述の例では、コンピュータ上のユーザディレクトリと、iCloudの「Documents」フォルダ以下を検索します。同じ検索コードをiOSでも実行できます。ただし、iOSが対応していない定数`NSMetadataQueryUserHomeScope`を削除する必要があります。

注： Mac OS Xの場合、ファイルシステムのメタデータはどのボリュームにもありますが、それ以外のメタデータ属性は使えないことに注意してください。Spotlightは、CD、DVD、ディスクイメージ、「System」ディレクトリについては索引づけをしません。

ユーザは意識的に、Spotlightの環境設定により、特定のディレクトリや文書タイプを、検索結果から除外することもできます。iOSの場合、「General」>「Spotlight Search」以下の「Settings」アプリケーションで、「Spotlight Search」オプションとして検索結果から除外できるのは文書タイプだけです。

検索の実行

クエリオブジェクトを作成、設定すると、いよいよ実行できるようになります。クエリの実行は通常、最初の結果収集フェーズと、ライブ更新フェーズの2つに分かれます。

最初の結果収集フェーズでは、既存のSpotlightシステムストアを対象に、検索式に合致するファイルを検索します。検索結果が返されると、クエリはNSMetadataQueryDidUpdateNotificationを使って通知を送ります。この通知は、単発のクエリであっても検索の進捗状況を示すために役立ちますが、ライブ検索の場合はより重要です。

クエリは、最初の結果収集フェーズが終了した時点で、アプリケーションに`NSMetadataQueryDidFinishGatheringNotification`通知を送ります。

- `metadataSearch`プロパティに`startQuery`メッセージを送ります。

検索結果へのアクセス

アプリケーションが検索結果にアクセスするためには、まずクエリを停止しなければなりません。最初の結果収集フェーズ中、あるいはライブ更新フェーズ中であっても、更新を無効にすることができます。

アプリケーションは、返された結果の個数を、NSMetadataQueryのインスタンスメソッドresultCountで調べます。次に、個々の結果項目を、インデックス（番号）を指定してアクセスします。resultsを順にたどる方法（Cocoa Bindingsでの利用を想定した使い方）よりも、インデックスを指定し、resultAtIndex:メソッドで結果を参照する方が便利でしょう。

結果項目はNSMetadataItem型のインスタンスの形で返されます。各オブジェクトには、該当するファイルのメタデータ属性がカプセル化されています。ファイルのメタデータ属性は、その属性名を指定して該当するインスタンスにvalueForAttribute:メッセージを送ることにより取得できます。

1. 進行中のクエリを停止します。
2. 結果について順次、アプリケーションの目的に応じた処理を施します。
3. 通知を監視する登録を削除します。

   繰り返しクエリを実行する場合、この手順は省略しても構いません。しかし、同じセットアップコードを使う予定であれば、いずれにしても登録を削除する方がよいでしょう。

静的検索のコード（最終形）

「静的なファイルメタデータ検索の実行」に、静的検索を実装したコードの例を示します。

リスト 2-1　静的Spotlight検索の実装

```
// 検索メソッドの初期化
- (void)initiateSearch
{
    // メタデータクエリのインスタンスを作成。metadataSearch @propertyは
    // retainとして宣言
    self.metadataSearch=[[[NSMetadataQuery alloc] init] autorelease];

    // 検索結果の更新通知、初回の検索終了通知を登録
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(queryDidUpdate:)

name:NSMetadataQueryDidUpdateNotification
                                               object:metadataSearch];

    [[NSNotificationCenter defaultCenter] addObserver:self

selector:@selector(initalGatherComplete:)

name:NSMetadataQueryDidFinishGatheringNotification
                                               object:metadataSearch];

    // UTIがpublic.imageである画像をすべて検索する旨の
```

```
    // 検索述語を設定
    NSPredicate *searchPredicate;
    searchPredicate=[NSPredicate predicateWithFormat:@"kMDItemContentTypeTree
== 'public.image'"];
    [metadataSearch setPredicate:searchPredicate];

    // 検索範囲を設定。この場合、ユーザのホームディレクトリ
    // およびiCloudの「Documents」以下を検索
    NSArray *searchScopes;
    searchScopes=[NSArray arrayWithObjects:NSMetadataQueryUserHomeScope,
                  NSMetadataQueryUbiquitousDocumentsScope,nil];
    [metadataSearch setSearchScopes:searchScopes];

    // 表示名を基準に結果を並べ替えるよう、
    // 整列順序を設定
    NSSortDescriptor *sortKeys=[[[NSSortDescriptor alloc]
initWithKey:(id)kMDItemDisplayName
                                                              ascending:YES]
autorelease];
    [metadataSearch setSortDescriptors:[NSArray arrayWithObject:sortKeys]];

    // 非同期クエリの開始
    [metadataSearch startQuery];

}

// 検索結果が更新された旨の通知を受信したときに起動されるメソッド
- (void)queryDidUpdate:sender;
{
    NSLog(@"A data batch has been received");
}


// 初回のクエリ収集処理が終了したときに起動されるメソッド
- (void)initalGatherComplete:sender;
{
    // クエリの停止、1回の検索処理が完了。
    [metadataSearch stopQuery];

    // 検索結果に対する処理。この場合、アプリケーションは単に、
    // 検索の結果得られた各画像の
    // 表示名キーを表示
    NSUInteger i=0;
    for (i=0; i < [metadataSearch resultCount]; i++) {
        NSMetadataItem *theResult = [metadataSearch resultAtIndex:i];
        NSString *displayName = [theResult valueForAttribute:(NSString
*)kMDItemDisplayName];
        NSLog(@"result at %lu - %@",i,displayName);
    }

    // 通知の登録を解除。
    // さらにmetadataQueryも解放。
    // クエリを削除すると、その結果も失われる。
    [[NSNotificationCenter defaultCenter] removeObserver:self

name:NSMetadataQueryDidUpdateNotification
                                                    object:metadataSearch];
    [[NSNotificationCenter defaultCenter] removeObserver:self
```

```
name:NSMetadataQueryDidFinishGatheringNotification
                                                    object:metadataSearch];
    self.metadataQuery=nil;
}

@end
```

ライブ検索の実行

ごくわずかの修正だけで、ライブ検索をする設定も可能です。

- 更新されたデータにアクセスする際には、`disableUpdates`でクエリを一時停止しなければなりません。
- データは通常、`queryDidUpdate:`メソッド内で処理します。
- その後、`enableUpdates`でクエリを再起動し、検索を続行します。

ファイルメタデータのクエリ式の構文

ファイルメタデータのクエリは、述語文字列書式のサブセットである、クエリ言語を使って組み立てます。メタデータ検索式は、実行時に検索式を組み立てることができるような構文になっています。そのため、習熟したユーザは独自のクエリを組み立てることができますし、一度使ったクエリを保存しておいて後で再利用することも可能です。構文は直感的に分かりやすい形式で、比較演算子、言語に依存しないオプション、時刻や日付を表す変数などが使えます。

比較構文

ファイルメタデータのクエリ式の構文は、シェルを使い慣れた人にはおなじみの、ファイル名の展開に用いる書式を簡便にしたものです。クエリは次の形式です。

```
attribute == value
```

ここで`attribute`は、標準的なメタデータ属性（『*File Metadata Attributes Reference*』を参照）、またはSpotlight Importerが定義した独自のメタデータ属性です。

たとえば、著者が「Steve」であるファイルをすべて検索する場合、クエリは次のようになります。

```
kMDItemAuthors ==[c] "Steve"
```

比較演算子を表 3-1に示します。

表 3-1　比較演算子

演算子	説明
`==`	等しい
`!=`	等しくない
`<`	より小さい（数値と日付のみ）
`>`	より大きい（数値と日付のみ）
`<=`	より小さいか等しい（数値と日付のみ）
`>=`	より大きいか等しい（数値と日付のみ）
`InRange(attribute-Name,minValue,maxValue)`	指定されたattributeNameの値（数値）がminValueからmaxValueまでの範囲内である

値の文字列中に「`"`」や「`'`」がある場合は、「`\`」を前置してエスケープしてください。

例では、検索値に修飾子「c」がついています。これは比較方法を表す修飾子です。表 3-2に、指定可能な修飾子を示します。

この修飾子は、比較演算子の直後に、角括弧[…]で囲んで指定します。

表 3-2　値比較の修飾子

修飾子	説明
c	大文字と小文字を区別しない。
d	分音符の有無を区別しない。

表 3-3に、比較修飾子の使用例をいくつか示します。

表 3-3　比較修飾子の使用例

クエリ文字列	結果
kMDItemTextContent == "Paris"	「Paris」に合致するが「paris」には合致しない。
kMDItemTextContent ==[c] "Paris"	「Paris」にも「paris」にも合致する。
kMDItemTextContent ==[c] "*Paris*"	「Paris」、「paris」、「I love Paris」、「paris-france.jpg」などに合致する。
kMDItemTextContent == "Frédéric"	「Frédéric」に合致するが「Frederic」には合致しない。
kMDItemTextContent ==[d] "Frédéric"	分音符の有無にかかわらず、「Frédéric」にも「Frederic」にも合致する。

ワイルドカード文字（「*」と「?」）を使って、文字列の先頭や末尾、中間の部分文字列に合致させることができます。表 3-4に典型的な使い方をいくつか示します。

「*」は任意の個数の文字に、「?」は1文字に合致します。

表 3-4　ワイルドカードの使い方

クエリ文字列	結果
kMDItemTextContent == "paris*"	先頭が「paris」である属性値に合致する。たとえば「paris」に合致するが、「comparison」には合致しない。
kMDItemTextContent == "*paris"	末尾が「paris」である属性値に合致する。
kMDItemTextContent == "*paris*"	位置にかかわらず、部分文字列として「paris」が含まれる属性値に合致する。たとえば「paris」や「comparison」に合致する。
kMDItemTextContent == "paris"	「paris」と完全に一致する属性値に合致する。

クエリは、ANDやORを表すC風の構文、すなわち「&&」や「||」で組み合わせることも可能です。たとえば、オーディオファイルであり、かつ、作者が「Steve」であるものを検索する場合、クエリは次のようになります。

```
kMDItemAuthors ==[c] "Steve" && kMDItemContentType == "public.audio"
```

括弧を使って演算子の優先度を制御できます。たとえば、オーディオファイルであって、作者が「Steve」または「Daniel」であるものを検索する場合、クエリは次のようになります。

```
(kMDItemAuthors ==[c] "Daniel" || kMDItemAuthors[c] == "Steve") &&
 kMDItemContentType == "public.audio"
```

さらに、ビデオファイルも検索対象に含める場合は、次のようになります。

```
(kMDItemAuthors ==[c] "Daniel" || kMDItemAuthors ==[c] "Steve" ) &&
 (kMDItemContentType == "public.audio" || kMDItemContentType == "public.video")
```

時刻や日付を表す変数

また、日付や時刻を検索値として用いるクエリも生成できます。日付や時刻の値は浮動小数点数で表され、これはCFDate（2001年1月1日からの経過秒数）とも互換性があります。

さらに、現在時刻を基準として値を指定するために、$timeという変数も使えます（表3-5を参照）。

表 3-5　$time変数式

時刻を表す変数	説明
$time.now	現在の日付と時刻。
$time.today	現在の日付。
$time.yesterday	昨日の日付。
$time.this_week(-1)	前週の各日。
$time.this_week	今週の各日。
$time.this_month	今月の各日。
$time.this_year	今年の各日。
$time.now(NUMBER)	現在の時刻に正または負の値（秒単位）を加えて得られる日付と時刻。
$time.today(NUMBER)	現在の日付に正または負の値（日単位）を加えて得られる日付。
$time.this_week(NUMBER)	現在の週に正または負の値（週単位）を加えて得られる週の各日。
$time.this_month(NUMBER)	現在の月に正または負の値（月単位）を加えて得られる月の各日。

時刻を表す変数	説明
`$time.this_year(NUMBER)`	現在の年に正または負の値（年単位）を加えて得られる年の各日。
`$time.iso(ISO-8601-STR)`	ISO-8601-STRに準拠した文字列を解析して得られる日付。

`$time`変数を使うと、たとえば先週中に修正したファイルは、次のクエリで検索できます。

```
((kMDItemAuthors ==[c] "Daniel" || kMDItemAuthors[c] == "Steve") &&
 (kMDItemContentType == "public.audio" || kMDItemContentType == "public.video"))
 &&
 (kMDItemFSContentChangeDate == $time.this_week(-1))
```

注： `$time`変数の値は、クエリを最初に実行する時点で設定されます。実行中に現在時刻が更新されることはありません。

FinderのSpotlight検索ウインドウの表示

アプリケーション画面上に、Finderの標準的な検索インターフェイスを表示することにより、ユーザはSpotlightを直接使って検索できるようになります。

NSWorkspaceのshowSearchResultsForQueryString:メソッドを使うと、簡単にFinderの検索ウインドウを実装できます。プログラム的には、ユーザがFinderに切り替え、新規ウインドウを開き、検索フィールドに検索文字列を入力したのと同等の処理になります。

「Finderの検索ウインドウを表示」に、文字列値を取り出して検索インターフェイスに表示するコード例を示します。

リスト 4-1　Finderの検索ウインドウを表示

```
resultCode=[[NSWorkspace sharedWorkspace] showSearchResultsForQueryString:[sender
 stringValue]];

if (resultCode == NO) {
    // パネルを開く処理に失敗したので、
    // エラー状況を表示する
    }
```

書類の改訂履歴

この表は「*ファイルメタデータ検索プログラミングガイド*」の改訂履歴です。

日付	メモ
2011-09-28	iCloudおよびiOSについての情報を追加しました。
2010-04-12	クエリの表記を訂正し、クエリの例をいくつか更新しました。
2009-07-14	クエリ構文を訂正しました。
2009-05-27	HISearchWindowShow()関数に代えて、NSWorkspaceの新しいshowSearchResultsForQueryString:メソッドを推奨するように変更しました。
2006-03-08	クエリ式のテーブルに$time.last_weekを追加しました。
2005-08-11	inRangeクエリ式に関する使用上の注意を追加しました。細かな誤字を訂正しました。
2005-04-29	Spotlightクエリの作成方法の考え方に関する内容を追加しました。「クエリ式の形式」に、$timeに関する使用上の注意を追加しました。
	Spotlight検索をアプリケーションに追加する方法について解説した新規文書。

書類の改訂履歴