iOS SDK (第2回)

～Xcodeの使い方とObjective-C～

my recommendations on research & development tools.

松村将太郎[†], (正会員) 熊澤逸夫[†]

キーワード：モバイルアプリケーション開発, iOS, iOS SDK, Xcode

1. ま え が き

本稿は、iPhoneやiPod touch, iPadなどのiOS搭載のモバイル機器でアプリケーション制作の方法を紹介する連載記事の第2回目である。

前回の第1回では、iOSの概要およびiOSアプリケーションの開発を行うためのツールであるXcodeとiOS SDKのインストールの手順について説明した。第2回となる本稿では、Xcodeおよびプログラミング言語Objective-Cを使って簡単なプログラミングを行う方法について説明する。

まず、Xcodeの画面の説明や操作方法について簡単に説明し、iOSアプリケーションを制作するためのプログラミング言語であるObjective-Cについて説明する。最後に、文字をデバイスの画面に表示する簡単なプログラミングの手順について説明する。

2. Xcodeの操作方法

Xcodeは、Mac OS X上で動作するAppleの統合開発環境であり、追加でiOS SDKをインストールすることで、iOS搭載デバイス上で動作するアプリケーションをプログラミングすることができる。

Xcodeには、バージョン3のXcode3とバージョン4のXcode4の2つがあるが、本稿では、執筆時点で最新のXcode4 (v4.2.1)の操作方法について説明する。

2.1 Xcodeの画面について

Xcodeのプロジェクトを開くと図1のような画面が表示される。Xcodeには上部のツールバーとナビゲーション、編集、ユーティリティ、デバッグの4つエリアが存在する。それぞれのエリアの説明は下記の通り。

(1) ツールバー：プログラムの実行やステータスの表示および画面に表示するエリアの切替えに関する項目を設定できる。ツールバーの使い方については2.2節で詳しく説明する。

(2) ナビゲーションエリア：プロジェクトに含まれるソ

図1 Xcode4の操作画面

ースコードやフレームワークおよびその他のリソースファイルが表示される。

(3) 編集エリア：プロジェクトの設定やソースコードの編集を行うエリアであり、ナビゲーションエリアで選択されたファイルに応じて画面が切替わるようになっている。

(4) ユーティリティエリア：現在編集中のオブジェクトの情報を設定したり、プロジェクトに新しいオブジェクトを追加する際に用いるエリアであり、編集中のオブジェクトに応じて表示される内容が大きく異なる。

(5) デバッグエリア：実行中のログが表示されるコンソール画面や変数の内容の確認が行えるエリアであり、デフォルトでは表示されていないため、表示するにはツールバーにあるデバッグエリアの表示ボタンを選択する必要がある。(表示方法は2.2節を参照のこと)

2.2 ツールバーの操作方法

Xcode上部のツールバーには、プログラムの実行や画面の切替えに関するボタンが設置されている。ツールバーの左側の画面を図2に、右側の画面を図3に示す。各画面に設置されている項目の説明は下記の通り。

(1) RunとStop：アプリケーションを実行したり、実行をやめたい場合にこのボタンを使用する(図2)。

[†] 東京工業大学 像情報工学研究所

"iOS SDK (2): How to use Xcode and introduction of Objective-C" by Shotaro Matsumura and Itsuo Kumazawa (Imaging Science and Engineering Laboratory Tokyo Institute of Technology, Yokohama)

図2 ツールバー左側

図3 ツールバー右側

- (2) Scheme：アプリケーションをPC側のシミュレータで起動するか、デバイス側で起動するかを切替えることができる(図2)。
- (3) Breakpoints：デバッグの際にプログラムを途中で止めることができるブレークポイントのオン・オフを切替えることができる(図2)。
- (4) Editor：ソースコードのエディタを標準エディタとアシスタントエディタ、バージョンエディタの3つの中から選ぶことができる(図3)。3つのエディタの説明は下記の通り。
- ・標準エディタ：1つのソースコードを編集エリアに表示する標準のエディタ。
 - ・アシスタントエディタ：編集エリアを2つに分割し、2つの異なるソースコードを表示するエディタ。
 - ・バージョンエディタ：1つのソースコードをバージョン別に2つの画面に表示するエディタ。
- (5) View：Xcodeの画面上に表示される編集エリア以外の各エリアの表示の切替えを行うことができる(図3)。
- (6) Organizer：PCに接続されているデバイスの情報などを閲覧できるオーガナイザウィンドウを開くことができる(図3)。

3. Objective-C

iOSプログラミングにはObjective-Cというプログラミング言語が採用されており、iOSアプリケーションを制作するためにはObjective-Cの習得が必須となっている。

ここでは、Objective-Cの概要とプログラミングの方法について説明する。ただし、ここでObjective-Cについて詳しく説明することは不可能であるため、必要最低限の項目についてのみ説明する。説明できなかった項目については、次回以降、必要に応じて説明する予定である。

Objective-Cについて詳しく知りたい方は、Appleから配布されているマニュアル¹⁾を参照されたい。

3.1 Objective-Cの概要

Objective-Cは、C言語にオブジェクト指向の拡張をプログラミング言語の一つであり、1983年にBrad Coxによって設計された。

Objective-Cはオブジェクト指向言語の元祖とされるSmalltalkの影響を強く受けており、他の代表的なオブジェクト指向言語であるC++やJavaなどと構文が大きく異なっている。また、Objective-CはC++やJavaと同じように事前にコンパイル作業を行うコンパイラ型言語でありながら、実行時にデータの動的な型付けを行う、インタプリタ型言語に近いという特徴がある。

3.2 クラスの定義と実装

本節からObjective-Cの説明を行っていくが、Xcodeに用意されているテンプレートを参照しながら説明を行っていくため、読者の方は下記の手順を参考にしつつ、実際に画面を見ながら確認していただきたい。

(1) プロジェクトを作成する

Xcodeを起動し、上部メニューバーから「File」⇒「New」⇒「New Project...」を選択し、iOS Applicationの「Single View Application」のプロジェクトを作成する。

(2) クラス定義の確認

Xcodeのナビゲーションエリアにある、View Controller.hを選択するとViewControllerというクラスの定義が編集エリアに表示される。この、hファイルの@interface～@end部分がクラスの定義を行う部分であり、クラスのインスタンス変数やメソッドの定義を行う必要がある場合には、この中に記述する。

また、@interface隣にあるViewController: UIViewControllerという記述は、ViewControllerクラスがUIViewControllerというクラスを継承していることを示している。

以上をまとめるとObjective-Cでは次のようにクラスを定義する。

@interfaceクラス名：継承するクラス名

ここに、インスタンスの定義やメソッドの定義を記述する。

@end

(3) クラスの実装の確認

次にXcodeのナビゲーションエリアにある、View Controller.mを選択するとViewControllerクラスの実装が編集エリアに表示される。この、mファイルの@implementation～@end部分がクラスの定義を行う部分であり、クラスのメソッドの実装はここで行う必要がある。

また、開いたViewControllerクラスの実装部分には、クラス定義で定義されていないメソッドが数多く実装されているが、これは親クラスであるUIViewControllerクラスのメソッドをオーバーライドしていることを示している。

以上をまとめるとObjective-Cでは次のようにクラスの実装を行う。

@implementationクラス名：継承するクラス名

ここに、メソッドの実装を記述する。

@end

3.3 インスタンス変数の定義

Objective-Cでは、インスタンス変数の定義は@interface～@endの間に記述する。例えば、int型の変数answerをインスタンス変数として定義する場合は次のように記述する。

読者の方は、ViewController.hに実際に下記のコードを記述していただきたい。

```
@interface ViewController : UIViewController {  
    int answer;  
}  
@end
```

このように、クラス定義部分の上部に {} を記述し、その中にインスタンス変数を定義することができる。

3.4 メソッドの定義と実装

Objective-Cには、メソッドの種類として、クラスメソッドとインスタンスメソッドが存在する。前者は、クラスから呼び出すことができるメソッドであり、後者はインスタンス化されたオブジェクトから呼び出すことができるメソッドである。ただし、クラスメソッドはメソッドの実装でインスタンス変数を使用することはできないので注意する必要がある。

(1) メソッドの定義

メソッド定義の例として、引数と返値がなくコンソールに"Hello World!!"と表示するクラスメソッドhelloWorldを考える。この場合、メソッドの定義は下記のように記述する。

```
+ (void) helloWorld;
```

最初の+は定義するメソッドがクラスメソッドであることを示しており、隣の()の中にメソッドの返値の型、最後にメソッドを識別するためのキーワード(helloWorld)を記述する。

次に、2つのint型の引数があり、2つの引数の和を3.3節で定義した変数answerに格納するインスタンスメソッドaddを考える。この場合、メソッドの定義は下記のように記述する。

```
- (void) add:(int)n1 to:(int)n2;
```

このように、引数があるメソッドを定義する場合、キーワード(add)の後に:と1つ目の引数を記述し、2つ目以降の引数に対しても引数の前にキーワードと:を記述していくことで、複数の引数を持つメソッドを定義することができる。

(2) メソッドの実装

次に、(1)で定義した2つのメソッドの実装例を下記に示す。

```
+ (void) helloWorld{  
    NSLog(@"Hello World!!");  
}
```

NSLog関数は、C言語のprintf関数のようなもので、コンソールに文字を表示することができ、デバッグにおいてよく用いられる関数である。

```
- (void) add:(int)n1 to:(int)n2{  
    answer = n1 + n2;  
    NSLog(@"Answer = %d", answer);  
}
```

このように、メソッドの実装は@implementation～@endの間に、(1)で定義したメソッドを記述し{}の中に、メソッドの処理を記述する。

3.5 メソッドの呼び出し

3.2節(3)のViewController.mファイルの実装部分の中に[]で括られた記述が何か所も見受けられるが、この[]で括られた部分がObjective-Cのメソッド呼び出しの記述となる。

Objective-Cのメソッド呼び出しでは、[]内の左側にメソッドが呼び出される側のオブジェクト名を指定し、右側に呼び出すメソッドを記述する。

(1) クラスメソッドの呼び出し

例えば、3.4節でViewControllerクラスに定義したクラスメソッドhelloWorldをViewController.m内のviewWillAppear:メソッド内で呼び出す場合、viewWillAppear:メソッドのfg内に下記のコードを記述する。

ここでコードを記述するviewWillAppear:メソッドとは、Viewが表示される直前に自動で呼び出されるメソッドである。

```
[ViewController helloWorld];
```

このように、クラスメソッドを呼び出す場合は、[]の左側にクラスの名前を記述する。Objective-Cではクラスもオブジェクトとして扱われるためこのような記述となる。

(2) インスタンスメソッド呼び出し

次に、3.4節でViewControllerクラスに定義したインスタンスメソッドadd:to:をViewController.m内のviewWillAppear:メソッド内で呼び出す場合、viewWillAppear:メソッドの{}内に下記のコードを記述する。

```
[self add:100 to:10];
```

このように、インスタンスメソッドを呼び出す場合は、[]の左側にインスタンスの名前を記述する。上記の例ではselfを指定しているが、これは呼び出されたメソッド(上記の例ではviewWillAppear:メソッド)のインスタンス自身のことを示している。

(3) コンソールの表示の確認

(1)、(2)でメソッドが実行されたかを確認するために

```
2011-12-26 21:43:53.617 HelloWorld[1441:f803] Hello World!!
2011-12-26 21:43:53.619 HelloWorld[1441:f803] Answer = 110
```

図4 コンソールに表示される実行結果

は、XcodeツールバーのRunボタンを押して、デバッグエリアのコンソールに結果が表示されているかを確認すればよい。プログラムを実行するとコンソールには図4のように表示される。

4. 画面に文字を表示するプログラム

最後に、UILabelというクラスを使って画面に文字を表示する簡単なプログラムについて説明する。

4.1 プログラムの概要

3章で使用したプロジェクトをそのまま使用し、ViewController.mファイルのviewDidLoadメソッド内にUILabelクラスを使用して画面に文字列を表示するプログラミングの手順について説明する。

viewDidLoadメソッドとは、Viewがインスタンス化されてロードされたときに自動で呼び出されるメソッドである。

4.2 プログラミングの手順

画面に文字を表示するプログラミングの手順を下記に示す。

(1) UILabelクラスのインスタンスを作成する

ViewController.mファイルのviewDidLoadメソッドの{}内に、下記のコードを記述することで、UILabelクラスのインスタンスを作成することができる。

```
CGRect rect = CGRectMake(110, 200, 100, 30);
UILabel *label = [UILabel alloc];
label = [label initWithFrame:rect];
```

最初のコードは、CGRectMakeというC言語の関数を使って、Labelを画面に表示する座標(最初2つの引数[x, y])とLabelの大きさ(残りの2つの引数[width, height])の情報を持つCGRect型の構造体を作成している。Viewの座標系は左上が原点となっている。

次のコードでは、クラスメソッドallocを使用してUILabelクラスのインスタンスを作成しており、最後に、initWithFrame:メソッドにおいて、UILabelの座標と大きさを指定して初期化を行っている。

(2) Labelに表示する文字をセットする

次に、下記のコードを記述することで、UILabelのインスタンスに表示する文字をセットすることができる。

```
[label setText:@"Hello World!!"];
```

Labelに文字を表示するためには、setText:メソッドを使用することで、メソッドの引数に指定した文字を表示することができる。

(3) ViewにLabelを追加する

最後に、下記のコードを記述することで、作成したLabelをViewに追加することができる。

図5 実行結果

```
UIView *view = [self view];
[view addSubview:view];
```

最初のコードにおいて、UIViewControllerクラスからUIViewクラスのインスタンス変数を取得している。そして、最後のコードで取得したUIViewクラスのインスタンスのaddSubview:メソッドを使用して、UILabelをViewに追加している。

(4) プログラムを実行する

XcodeのRunボタンを押すとプログラムがビルドされ、エラーがなければ実行結果は図5のようになる。

5. む す び

本稿では、Xcodeの操作方法について説明し、iOSアプリケーションを作成するために必要なObjective-Cの概要と画面に文字を表示する簡単なプログラミングの手順について説明した。今回は、Interface Builderの使い方について解説していく予定である。

(2011年12月29日受付)

[文 献]

- 1) Objective-Cプログラミング言語, Apple Inc., <http://developer.apple.com/jp/d-evcenter/ios/library/documentation/ObjC.pdf>
- 2) 柴田文彦: "Xcode4入門for iOS/Mac OS X", 秀和システム

松村 将太郎 2011年、東京工業大学大学院総合理工学研究科博士後期課程修了。同年、同大学像情報工学研究所研究員。ユーザインタフェースや画像処理などの研究に従事。博士(工学)。

熊澤 逸夫 1981年、東京工業大学工学部電気電子学科卒業。1986年、同大学院博士課程修了。同年、同大学工学部助手。現在、同大学像情報工学研究施設教授。パターン認識・画像処理、人工ニューラルネットワーク、認知知覚モデルの研究に従事。博士(工学)、正会員。