iPhone & iPad プログラミング Bible［上］

河西 朝雄著

Xcodeという統合開発環境を使用し、
Objective-Cというプログラミング言語を使って
iPhone/iPad向けアプリケーションを開発します。

KASAI.SOFTWARELAB

定価1,620円（税込）

iPhone&iPadプログラミングBible[上]

河西　朝雄著

iPhone/iPad向けアプリケーションを開発するにはAppleが提供するXcodeという統合開発環境を使用し、Objective-Cというプログラミング言語を使ってプログラムコードを記述します。

本シリーズは、iPhone/iPadアプリを開発するためのテクニックをすべて網羅するように22の章（カテゴリ）に分類して「上」、「下」の2分冊で構成しています。本書はその中の「上」です。

KASAI.SOFTWARELAB

定価 1,620 円（税込）

はじめに

iPhoneなどのスマートフォンやiPadなどのタブレット端末のユーザーインターフェースは指のタッチを基本とし、カメラやセンサーを内蔵し、音声認識・音声合成などが簡単に利用できる画期的なコンピュータです。マウス、キーボード、デイスプレイを主なユーザーインタフェースとするパソコンとは大きく異なります。「コンピュータ＝パソコン」の時代から「コンピュータ＝スマートフォン、タブレット端末」の時代に急速にパラダイムシフトしようとしています。スマートフォンは子供から女性、シニアまでの広い層に渡って、今までのパソコンユーザとは比べ物にならない数のユーザが見込まれます。

Appleが運営するアプリケーションマーケットとしてApp Store(アップストア）があります。2013年6月時点で有料、無料含め90万を超えるアプリケーションが提供されています、App Storeを通して企業だけでなく、一般ユーザが自作のアプリケーションを販売することができる点も今までにない利点です。つまり、ソフト会社の技術者以外にも、学生を中心に一般の人でもiPhone/iPadアプリで商売ができるようになる可能性があり、iPhone/iPadアプリ市場は今後急速に普及すると思います。

iPhone/iPad向けアプリケーションを開発するにはAppleが提供するXcodeという統合開発環境を使用し、Objective-Cというプログラミング言語を使ってプログラムコードを記述します。

本シリーズは、iPhone/iPadアプリを開発するためのテクニックをすべて網羅するように22の章（カテゴリ）に分類して「上」、「下」の2分冊で構成しています。本書はその中の「上」です。22の章というのはかなり多い章分けですが、細かく章分けをすることでカテゴリが分かり易く、各章のサイズは小さくなり初心者には、ひとつのまとまった単位がボリュームが少ないので、取りかかり易くなります。また、章の順序ではなく、知りたい章を先に学習することもできます。

既存の書籍やネット上の情報は重要な内容とそうでない情報がまぜこぜになっていたり、このプログラムをどこに書けばいいのかが曖昧だったり、サンプルが長すぎたりなど、初心者には理解しにくい内容が多いです。本シリーズではiPhone/iPadアプリを作る上で必要な技術的要素やテクニックを切り出し短いサンプルを付けて簡潔に提示します。

「上」は何らかの言語でプログラム経験はあるが、Objective-CやiPhone/iPadアプリを初めて勉強する人を主な対象とします。iPhone/iPadアプリを作るためには、Objective-Cの知識が必要になります。Objective-Cを本格的に学ぶには別な入門書が必要になりますが、本書ではiPhone/iPadアプリを作りながらObjective-Cも手っ取り早く学べるように工夫してあります。そこでまず、グラフィックスを利用して画面に四角や円、イメージなどを描画するプログラムを例にObjective-Cの基本的な言語仕様について2章で学び、さらに3章で詳しく説明します。iPone/iPadを動かしているOSをiOSと呼びます。iPhone/iPadアプリを作るためには、Objective-Cが持つデータ型だけでなく、iOSが提供する基本データクラスを

使う必要があります。これを4章で説明します。iPhone/iPadではボタンやラベルなどのGUI部品をユーザインターフェース要素（UI要素）と呼んでいます。iPhone/iPadで利用できる全てのUI要素の使い方を5章で説明します。画面を指でタッチしたり、デバイスをシェイクしたりするイベントが発生します。このイベントを処理する方法を6章で説明します。XcodeではiPhone/iPadアプリを作るための基本となるアプリケーションを7種類用意しています。各アプリケーションの作り方を7章で説明します。画面（ビュー）遷移はStoryboardのセグエ（segue）という機能を使うと簡単に２つの画面の接続関係を作ることができます。いろいろな画面（ビュー）遷移の方法を8章で説明します。通知センターと通知、タイマー、遅延実行、マルチタスクとバックグランド処理などのアプリケーションの制御に関する内容を9章で説明します。

本書は次のような章の構成となります。

１章　XcodeでのiPhone/iPadアプリの開発法
２章　グラフィックスを用いたObjective-C入門
３章　Objective-Cの文法
４章　iOSが提供する基本データクラス
５章　ユーザインターフェース要素（UI要素）
６章　イベント処理
７章　アプリケーションの種類
８章　画面（ビュー）遷移
９章　アプリケーションの制御

本書のプログラムはiPhoneを想定したものですが、基本的にはiPadでも使用できます。iPhone/iPad用のプログラムは基本的に同じ方法で記述できますが、両者は画面サイズが異なるため、画面サイズに依存するプログラムは変更が必要になります。また用意するアイコンのサイズも異なります。

これからiPhone/iPadアプリの開発を志す方々にとって、本書が少しでもお役に立てば幸いです。

2014年5月　河西朝雄

別冊のiPhone/iPadプログラミングBible[下]の内容は以下です。

目次

1章　XcodeでのiPhone/iPadアプリの開発法

XcodeはMacOS XおよびiOSにおける統合開発環境(IDE:Integrated Development Environment)です。C、C++、Objective C++、Java、AppleScriptなどでのプログラム開発をサポートします。iPhone/iPad用アプリケーションはXcodeのObjective-Cを使って開発します。

XcodeはMac OS X 10.4 Tiger と共に、2005年にリリースされたVer.2.0が最初です。2008年にリリースされたXcode 3.1からiOSにも対応するようになりました。2011年にXcode 4.0/4.1/4.2が矢継ぎ早にリリースされ、2012年に4.3/4.4がリリースされました。2012年9月にはiOS 6に対応した4.5がリリースされました。

この章ではXcode 4.2/4.3を使ってiPhone/iPad用アプリケーションを開発するための手順について解説します。

1-1　Xcode 4.2/4.3/4.4の特徴

Xcode 4.xの中でXcode 4.1から4.2において大きな改訂がありました。Xcode 4.2/4.3/4.4はXcode 4.1以前に対し、以下のような点が主に変更されています。

1.　Xcodeテンプレートの自動生成ファイル群の変更

Xcode 4.2/4.3で開発できるアプリケーションの種類は以下です。本書では当面はSingle View Application（旧バージョンのView-based Application）で作成します。

①Master-Detail Application

ナビゲーションコントローラを使用したマスターとデイテールの分割ビューからなるアプリケーションです。旧バージョンのNavigation-based ApplicationとSplit View-based Applicationを一緒にしたものに相当します。

②OpenGL Game

OpenGL ESベースのゲーム用アプリケーションです。旧バージョンのOpenGL ES Applicationに相当します。

③Page-Based Application

ページビューコントローラを使用したページベースのアプリケーションです。新たに追加されたアプリケーションテンプレートです。

④Single View Application

単一のビューを使用するアプリケーションです。旧バージョンのView-based Applicationに相当します。

⑤Tabbed Application

タブバーコントローラを使用したタブベースのアプリケーションです。旧バージョンのTab Bar Applicationに相当します。

⑥Utility Application

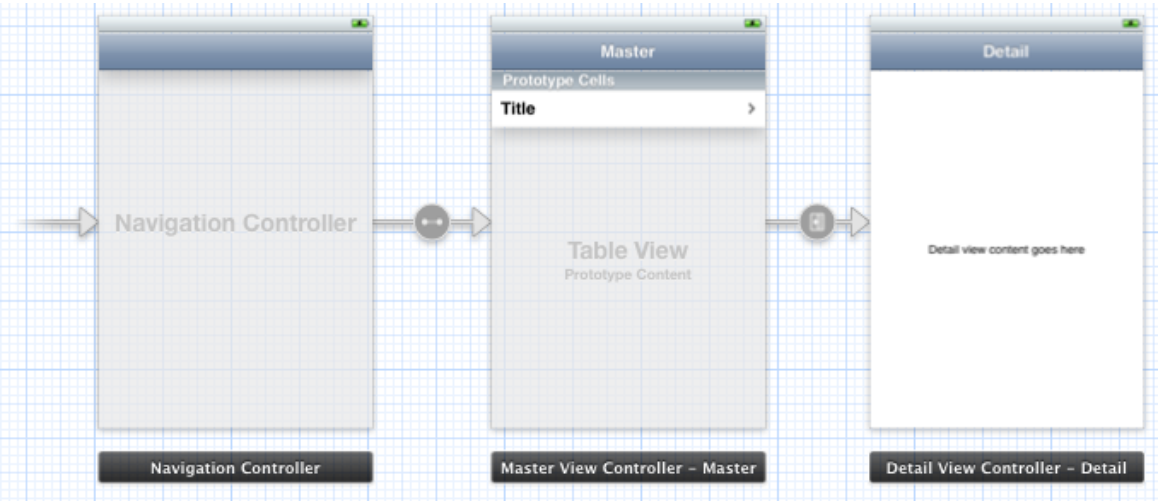
メインビューと代替ビューを持つユーティリティアプリケーションです。旧バージョンと同じ名前です。

⑦Empty Application

空(Windowのみ)のテンプレートからなる任意のアプリケーションです。旧バージョンのWindow-based Applicationに相当します。

2. Storyboard導入

Xcode 4.1以前ではInterface Builderを使用していましたがより直感的にUI要素を作成できるStoryboardを導入しました。以下はMaster-Detail Applicationテンプレートでの Storyboadです。



3. Automatic Reference Counting（ARC）導入

iOS5からARC（Automatic Reference Counting：自動参照カウント）が使えるようになりました。これによりretainまたはreleaseなどのメモリ操作関連のキーワードを省略可能になり、クラッシュやメモリリークなどを防ぐことができます。ただし、それらキーワードがソース中で使用されているとコンパイル時にエラーが出るようになります。ARCを使わない場合は明示的にオブジェクトを獲得、解放し独自にメモリ管理しなければなりません。たとえば古いバージョンでは

```
NSArray *fruits=[NSArray
arrayWithObjects:@"apple",@"orange",@"strawberry",nil];
```

のように宣言されたオブジェクトは使用を終了するには

```
[fruits release];
```

のようにしてメモリ解放しなければいけませんでしたが、ARCを使えば[fruits release];は不要で逆にあればエラーとなります。同様に

```
NSMutableArray *arr = [[[NSMutableArray alloc] init] retain];
```

のようにretainを指定するとエラーとなります。これは

```
NSMutableArray *arr = [[NSMutableArray alloc] init];
```

とします。

1-2　Xcode 4.5/iOS 6の特徴

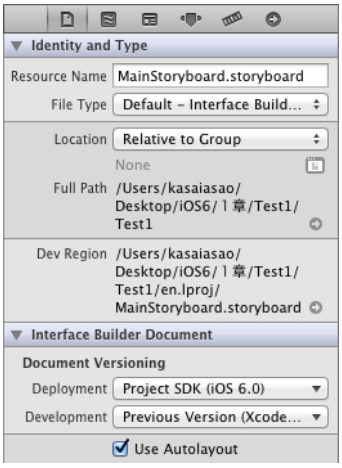
Xcode 4.5はiOS 6に対応し、以下のような特徴があります。

1.　iPhone5向け解像度(Retina 4-inch:640x1136)対応

Simulatorの画面サイズは従来のiPad,iPad(Retina),iPhone,iPhone(Retina 3.5-inch)の他に、iPhone(Retina 4-inch)が追加されました。

2.　Autolayout機能のサポート

Autolayout のサポートを追加しました。Autolayout は iOS 6 から追加された、複数のスクリーンサイズをサポートするための機能です。Storyboardの「File inspector」を選択すると、「Use Autolayout」にチェックが入っています。



Autolayout機能はiOS 5ではサポートしないので、iOS 5で動作させるとエラーとなります。Storyboardで「Use Autolayout」のチェックを外すとiOS 5用のアプリは作れますが、Autolayout機能を外したことによる弊害が生じます。「1-12　Xcode 4.5でiOS 5のアプリを開発する場合」参照。

3.　画面の回転処理メソッドの変更

shouldAutorotateToInterfaceOrientationメソッドが廃止され、代わりにsupportedInterfaceOrientationsメソッドとshouldAutorotateメソッドを使うようになっています。

4. UICollectionViewControlleの追加

UICollectionViewControllerは縦横の要素（セル）にサムネイルなどをスクロール表示し、セルを選択できるようにしたものです。「5-30　UICollectionViewController」参照。

5. Objective-Cの言語仕様の変更

この変更はXcode 4.4から変更されています。

①@shynsesizeコンパイラディレクティブの省略

「1-8　OutletとAction」参照。「3-6　Objective-Cで拡張された機能」－「11.　@propertyと@synthesize」参照。

②NSArray/NSMutableArrayやNSDictionary/NSMutableDictionaryに対し[ ]を使って要素を参照する機能

「4-7　NSArrayクラス」－「6.　[ ]による要素の参照」参照。「4-9　NSDictionaryクラス（連想配列）」－「4.　[ ]による要素の参照」参照。

6. Mapの変更

GoogleMapからApple独自のApple Mapsに変わります。使用するMapKit.frameworkは同じですが、表示される地図情報が異なります。

1-3　プロジェクトの作り方

「Single View Application」テンプレートのプロジェクトを作成する手順を以下に示します。

①Xcodeの起動

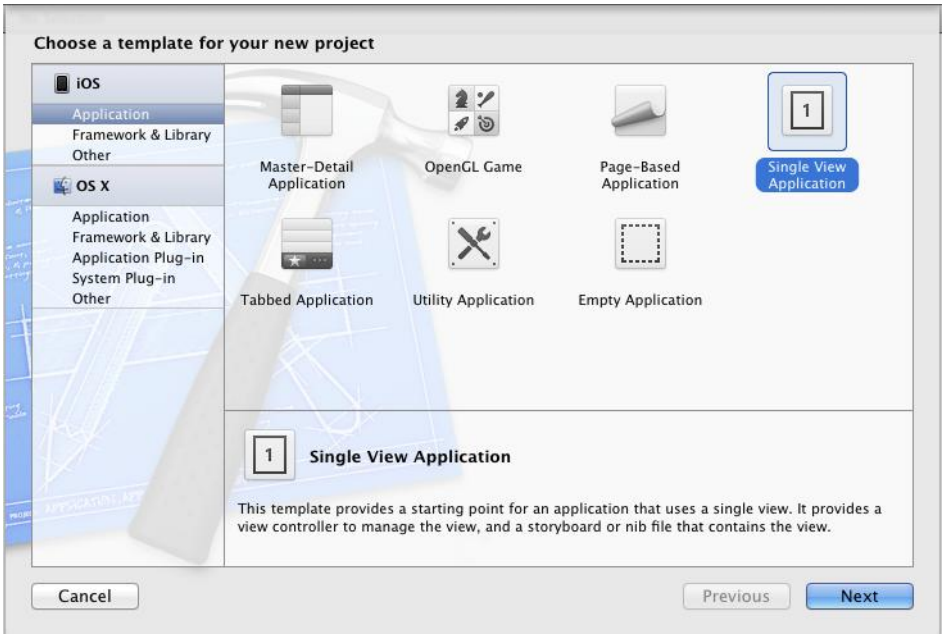
タスクバーからXcodeアイコンをクリックします。

②新規プロジェクトの生成

「Create a new Xcode project」を選択します。

③テンプレートの選択

「Single View Application」を選択します。



④オプションの選択

プロジェクトに対する各種オプションを以下のように設定します。

各項目の意味は以下の通りです。

・Product Name:製品名（プロジェクト名）です。この名前のフォルダが生成され、その中にプロジェクトファイルが格納されます。

・Organization Name:組織名（会社名、個人名）です。

・Company Identifier:会社の識別子です。

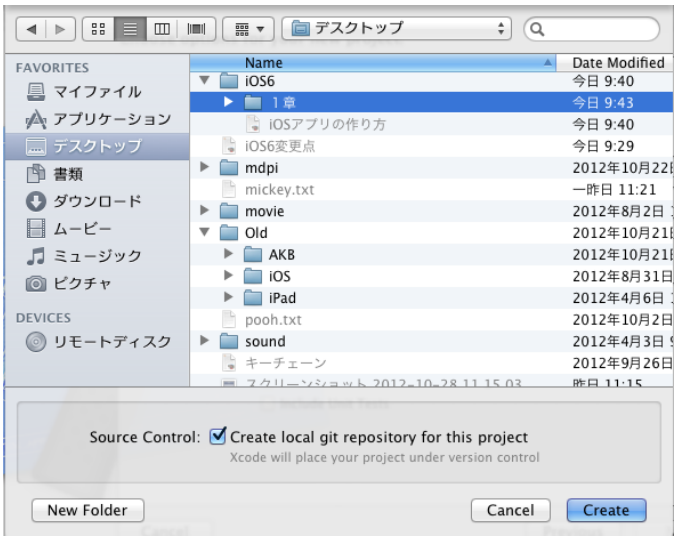
・Bundle Identifier：入力した「Company Identfier」と「Product Name」を合成した名前が自動的につきます。

・Class Prefix：クラス名の先頭につける名前です。デフォルトで灰色の「XYZ」になっています。Prefixを指定しない場合はこのままにします。

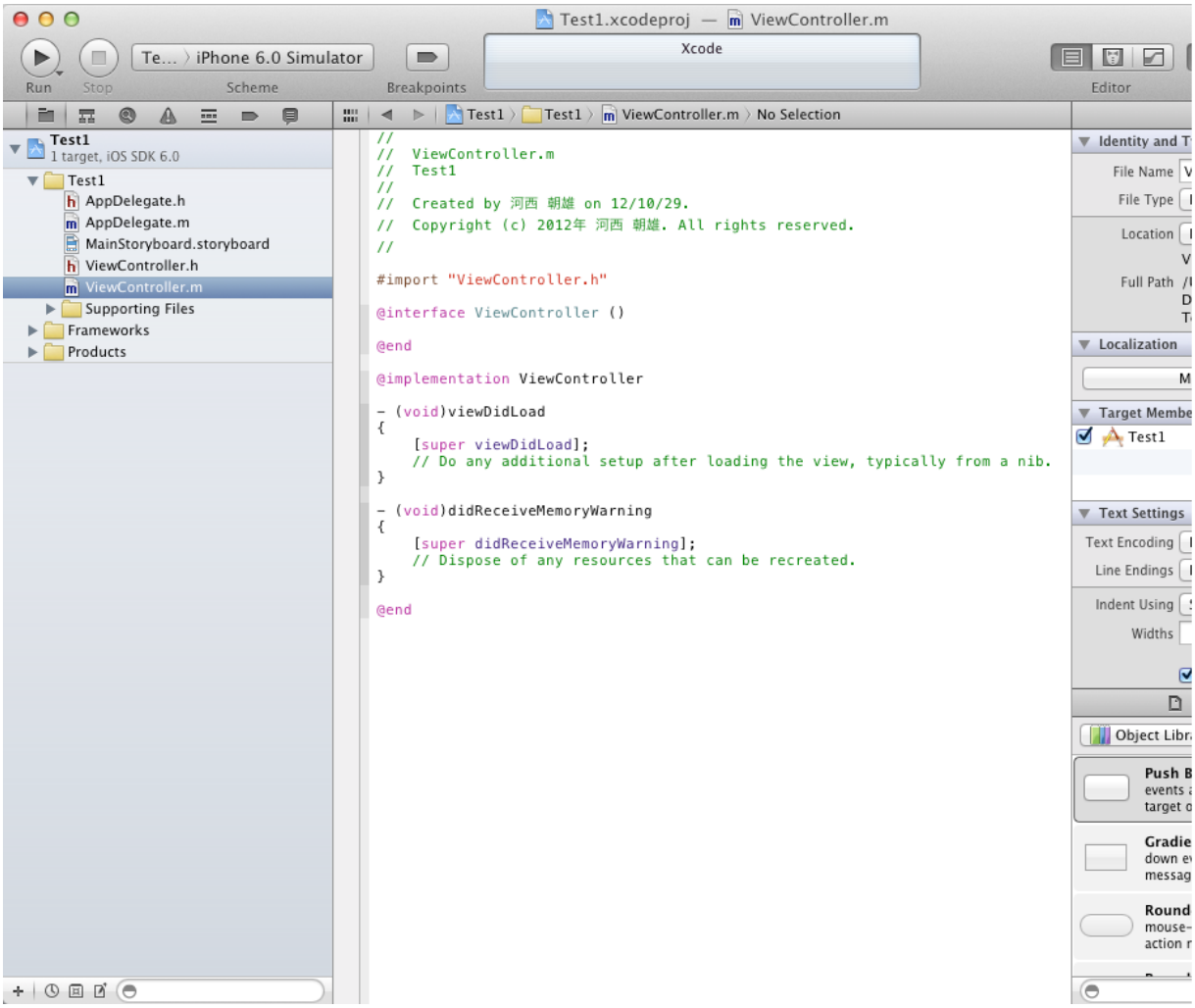
・Device Family:対象となるデバイスです。iPone,iPad,Universal(iPhone/iPad両用)の3種類が指定できます。

・Storyboards：Storyboardsを使用する場合にチェックを入れます。通常チェックを入れます。
・ARC：ARC（Automatic Reference Counting：自動参照カウント）機能を使用する場合にチェックを入れます。通常チェックを入れます。
・Include Unit Tests：Unit Testsをは使用する場合にチェックを入れます。通常チェックを入れません。

⑤保存先の選択
　保存先のフォルダを選択（この例ではデスクトップのiOS6/1章フォルダ）し「Create」で作成します。



⑥作成されたプロジェクト



プロジェクト内に作成されたファイルは以下です。

①AppDelegate.h/AppDelegate.m

アプリ全体の動作（起動と終了、バックグランドとフォアグランドなど）を規定するのに使います。デリゲートについては後で説明します。

②MainStoryboard.storyboard

オブジェクト（UI要素）、遷移、接続などの情報が記述されています。UI要素を配置したり、アウトレット接続を行うときに使います。

③ViewController.h/ViewController.m

Viewの動作を規定します。当面このファイルにコードを記述していきます。デフォルトで生成されているコードの意味を以下に説明します。

@interface～@endでクラスの宣言、@implementation～@endでクラスの定義（実装）を行います。デフォルトで作成されるメソッドはviewDidLoad、viewDidUnload、shouldAutorotateToInterfaceOrientationです。

・ViewController.h

```
#import <UIKit/UIKit.h> ←指定したファイルをインポートします

@interface ViewController : UIViewController ←クラスの宣言。クラス名はViewController。UIViewControllerは基底クラス

@end
```

・ViewController.m

```
#import "ViewController.h"

@interface ViewController () ←クラスの宣言。クラス名はViewController。基底クラスは省略できる

@end

@implementation ViewController ←クラスの定義（実装）。クラス名はViewController

- (void)viewDidLoad ←Viewがロード（生成）された時に呼び出されます
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning ←メモリ不足時に呼び出されます
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

「注」Xcode 4.3でのデフォルトコード

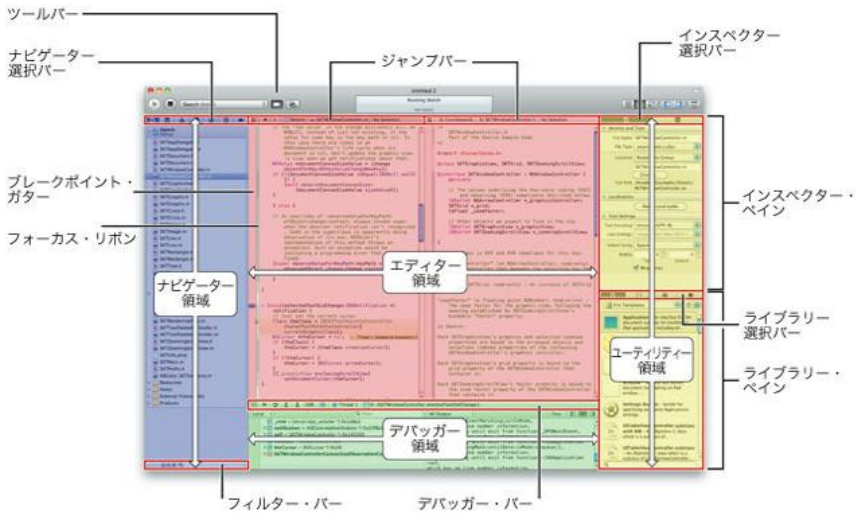
ViewController.mのデフォルトコードが異なります。didReceiveMemoryWarningメソッドの代わりに、viewDidUnloadメソッドとなります。
shouldAutorotateToInterfaceOrientationメソッドが余分にあります。

```
- (void)viewDidUnload ←Viewがアンロード（解放）された時に呼び出されます
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrie
ntation ←デバイスが回転した時に呼び出されます
{
    return (interfaceOrientation != UIInterfaceOrientationPortraitUpsideDown);
}
```

1-4　Xcodeのプロジェクトウインドウの構成

Xcodeのプロジェクトウインドウの構成は以下の通りです。



1.　ツールバー

ツールバーは以下のような各ブロックで構成されています。

①Run/Stop

Simulatorの実行と停止を行います。

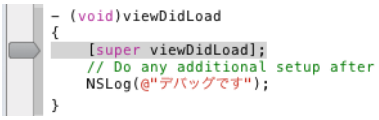
②Scheme

ターゲットとなるデバイス（Simulatorまたは実機）を指定します。

③Breakpoints

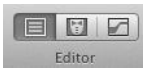
設定してあるブレークポイントを有効/向こうで切り替えます。

ブレークポイントはエディタ領域の各行の左端をクリックすることで設定/解除ができます。



④Editor

エディタ領域に表示する画面構成を選択します。左から「Standard editor：標準のエディタです」、「Assistant editor : Outlet、Action接続する際に使います」、「Version editor」です。

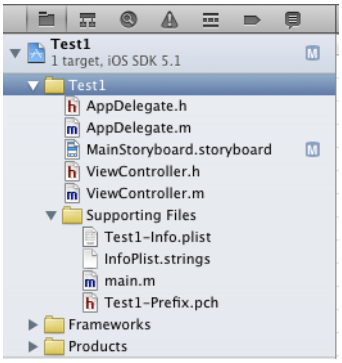


⑤View

各領域の表示/非表示を選択します。左から「ナビゲータ領域」、「デバッグ領域」、「ユーティリティ領域」の表示/非表示を選択します。

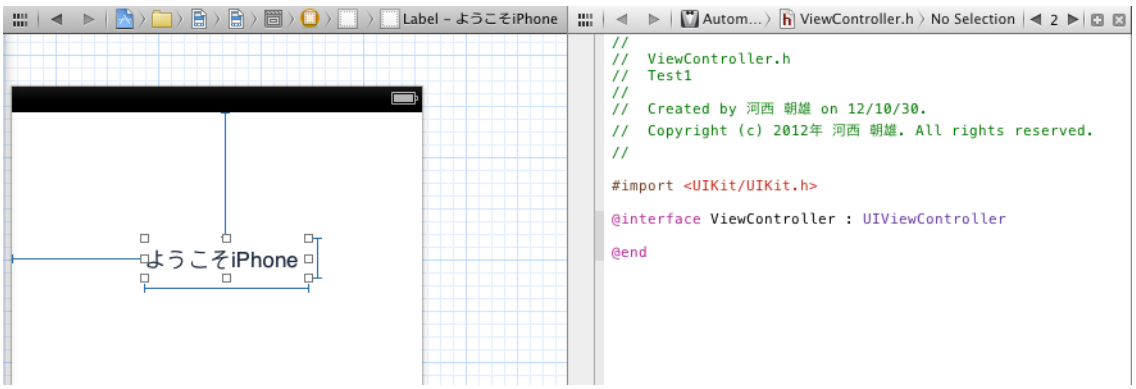
2. ナビゲータ領域

上部のタブを切り替えることで、Project navigator,Symbol navigator,Search navigator,Issue navigator,Debug navigator,Breakpoint navigator,Log navigatorが表示されます。以下はProject navigator でプロジェクト内の各種ファイル一覧が表示されます。



3.　エディタ領域

コード、storyboard,plistなどのすべてのファイルの編集領域です。

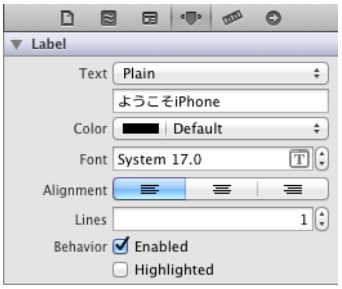


4.　ユーティリティ領域

ユーティリティ領域は上部のインスペクタペインと下部のライブラリペインに分かれます。

①インスペクターペイン

アイコンで示されるタグで各インスペクターを選択します。左から「File inspector」、「Quick Help inspector」、「Identity inspector」、「Attributes inspector：UI要素の属性を設定します」、「Size inspector」、「Connections inspector」です。

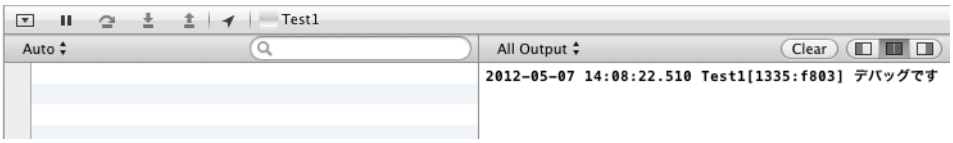


②ライブラリペイン

アイコンで示されるタグで各ライブラリを選択します。左から「File Template library」、「Code Snippet library」、「Object library：UI要素の一覧が表示されます」、「Media library」です。

5. デバッグ領域

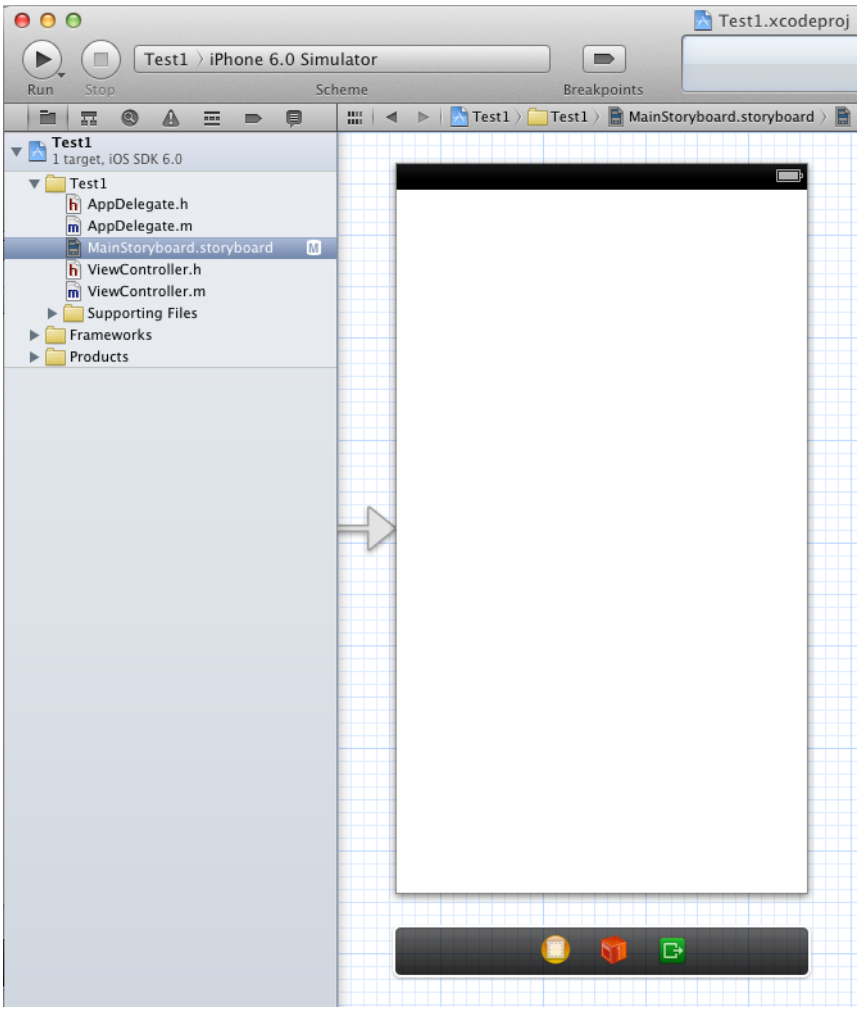
デバッグ情報が出力される領域です。「NSLog(@"デバッグです");」のようなLog出力の結果が表示されます。



1-5　UI要素の配置

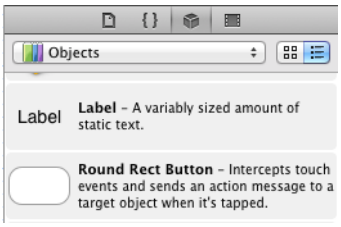
　ラベルやボタンなどのUI（User Interface）要素をStoryboardを使って配置する方法を説明します。

①MainStoryboard.storyboardを選択



②オブジェクトライブラリの選択

 ボタンをクリックしてオブジェクトライブラリ画面を開きます。

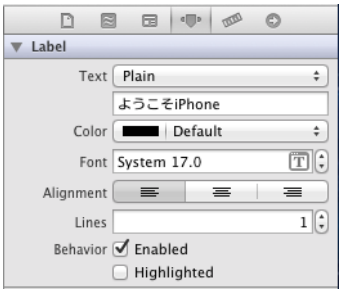


③LabelをViewへ配置

オブジェクトライブラリからLabelをViewにドラッグドロップします。

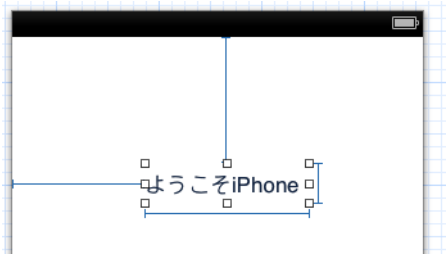
④Labelのタイトルを設定

Attribute inspectorタブ を選択し,Text属性に「ようこそiPhone」を設定します。



⑤ラベルのサイズ、位置の調整

ラベルを囲む□にマウスを合わせラベルのサイズを調整します。ドラッグドロップで位置を移動することもできます。



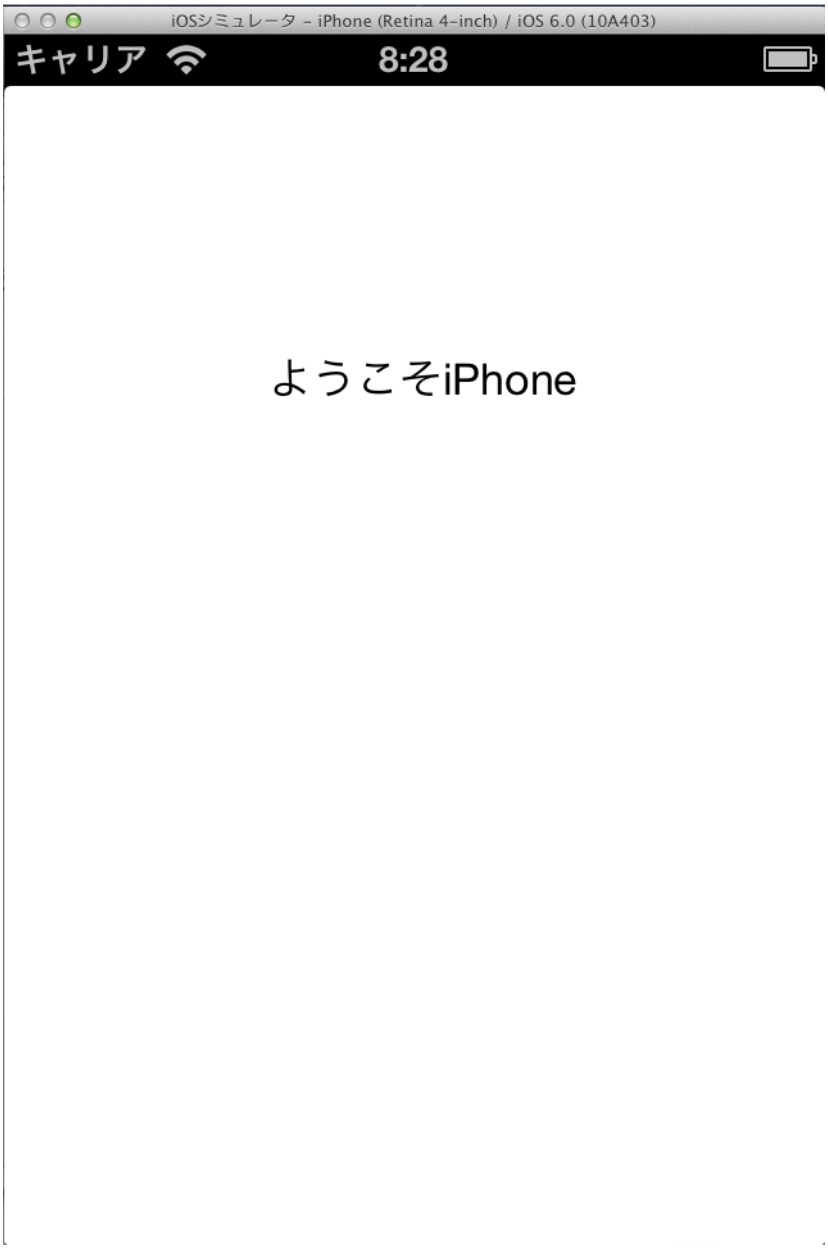
「注」iPhone(Retina 4-inch)を選択したときのSimulator画面は縦方向は画面一杯のサイズになり、縦方向の内容は全部入りきれずスクロールして表示します。このため画面上部に近いUI要素はSimulatorでの実行時にスクロールされている場合があり、一見すると表示されていない状況になる可能性があります。スクロールすれば見えるわけですが、最初の実行結果が確実に見えるようにするにはUI要素の位置は画面上部近辺でなく少し下げた位置に配置しておいた方が良いでしょう。

　または「1-7　Simulatorの解像度と表示サイズ」－「2.　Simulator画面の表示パーセント」で示す方法で画面サイズを変更します。

1-6 プロジェクトの実行

実行ボタンで実行します。

以下のようなSimulator画面が表示されます。デバイスが「iPhone Retina 4-inch」の場合。



「注」実行結果は選択されているデバイス（iPhone,iPhone Retina 3.5inch,iPhone Retina 4-inch）により異なります。デバイスがiPoneの場合以下のようになります。

1-7 Simulatorの解像度と表示サイズ

Simulatorの解像度と表示サイズはデフォルトで「iPhone(Retina 4-inch)」と「100%」ですが、以下のように設定できます。

1.　Retinaディスプレイ

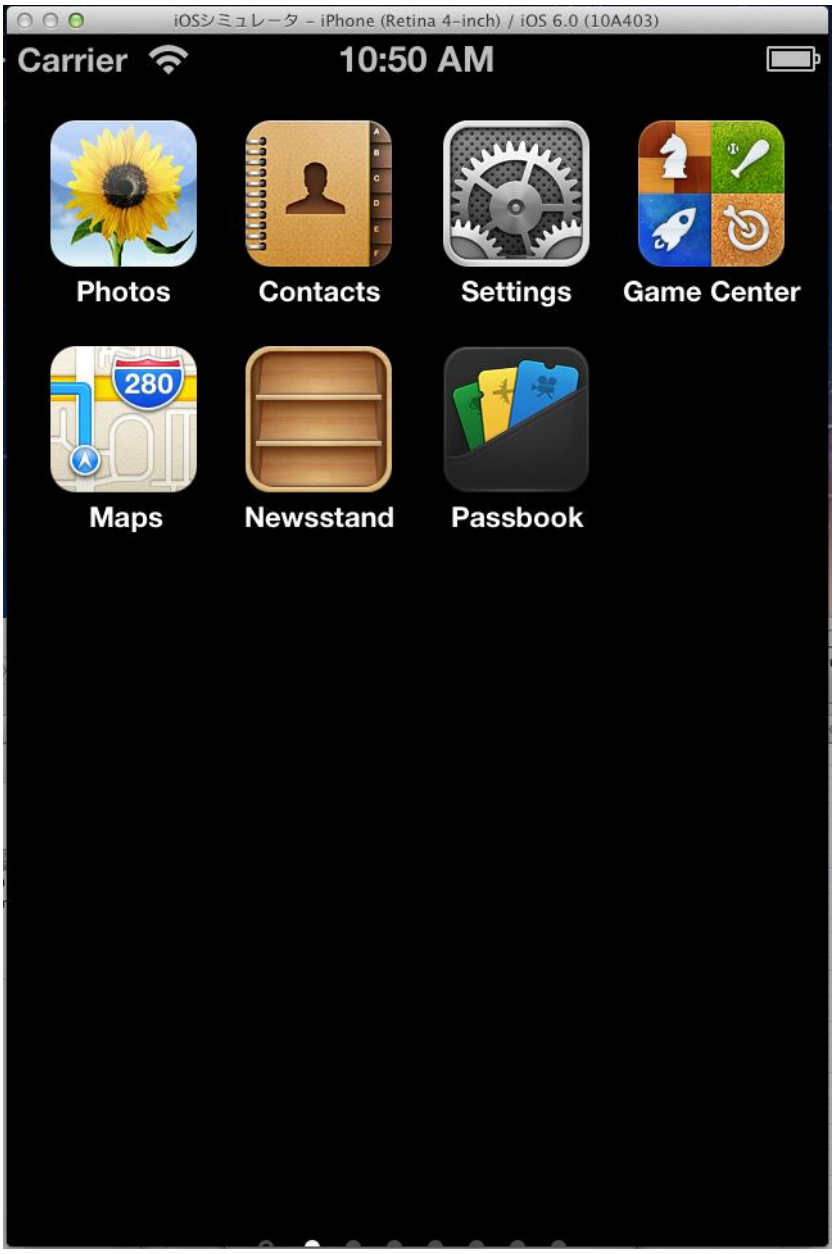
iPhone 4/4Sに搭載された従来のものより4倍のピクセル密度（326ppi）を持つディスプレイをRetinaディスプレイと呼びます。iPhone 5もRetinaディスプレイです。ディスプレイの解像度については「4-27 UIScreenクラス」参照。

Simulatorのデバイスを選択するにはiOSシミュレータ  を選択し「ハードウエア」ー「デバイス」メニューからデバイスを選択します。

・iPhoneを選択したときのSimulator画面

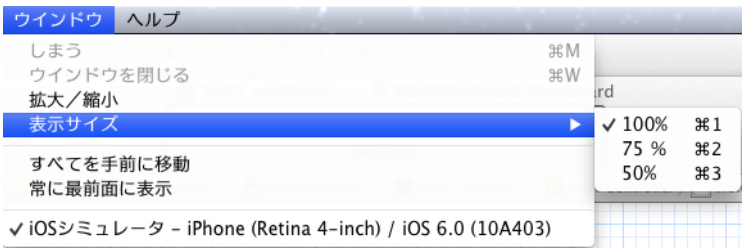
・iPhone(Retina 4-inch)を選択したときのSimulator画面

Simulatorの縦方向は画面一杯のサイズになり、縦方向の内容は全部入りきれずスクロールして表示します。



2. Simulator画面の表示パーセント

iPhone(Retina 4-inch)を選択したときのSimulatorの縦方向は画面一杯のサイズになり、縦方向の内容は全部入りきれずスクロールして表示します。「ウインドウ」－「表示サイズ」を「75%」に設定すると見やすい画面サイズとなります。


ウインドウ
ヘルプ
しまう ⌘M
ウインドウを閉じる ⌘W
拡大／縮小
表示サイズ
すべてを手前に移動
常に最前面に表示
iOSシミュレータ – iPhone (Retina 4-inch) / iOS 6.0 (10A403)
100% ⌘1
75 % ⌘2
50% ⌘3

1-8　OutletとAction

先のTest1プロジェクトは単にラベルのテキストをView上に表示するだけのもので、Objective-Cのコードをユーザが記述する必要はありませんでした。

ここではラベルの内容を変更したり、ボタンをクリックしたときのアクションをコードで記述する方法を説明します。UI要素とコードとの接続(Connection)をOutletとActionを用いて行います。

1.　ラベルのOutlet作成とボタンのAction作成手順

①ラベルとボタンを配置

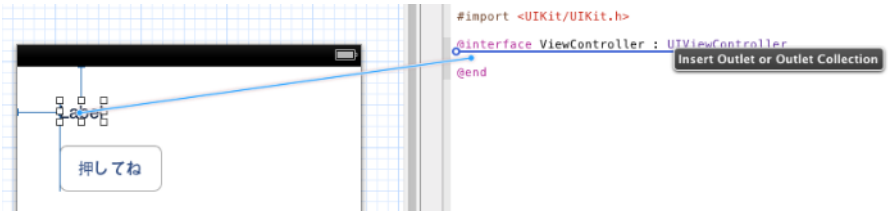
Object LibraryからLabelとRound Rect ButtonをStoryboardにドラッグドロップします。

②AssistantEditorの選択

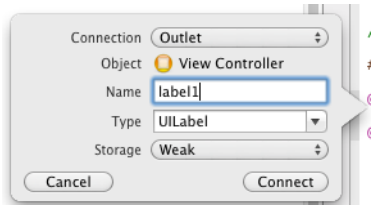
Assistant Editor（以下の中央）を選択し、左にStoryboard、右にViewController.hを表示します。

③ラベルのOutletを作成

「control」キーを押しながらラベル上でマウスダウンしてViewController.h上にドラッグドロップします。

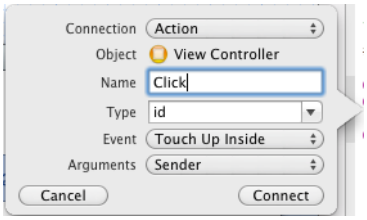


Conenectionはデフォルトの「Outlet」、Nameを「label1」とします。Type、Storageはデフォルトの「UILabel」、「Weak」とします。これでコード中でこのラベルを「label1」というプロパティ名で使用することができるようになります。



④ボタンのActionを作成

「control」キーを押しながらボタン上でマウスダウンしてViewController.h上にドラッグドロップします。Conectionに「Action」を選択します。Nameを「Click」とします。Type、Event、Argumentsはそれぞれデフォルトで設定されている「id」、「Touch Up Inside」、「Sender」とします。これでボタンがクリックされたときに呼ばれるメソッドを定義できます。Eventの種類の「Touch Up Inside」はタッチアップ動作を意味します。



⑤自動生成されたコード

以上の操作によりViewController.hとViewController.mに自動生成されたコードは以下の下線部です。

ラベルに関しては「.h」で@propertyを使ってプロパティ名の宣言文が自動生成されます。このときXcode 4.5では「_label1」というインスタンス変数がシステム側で自動で用意されます。ラベルを参照する際、プロパティを使う場合は「self.label1」とし、インスタンス変数を使う場合は「_label1」とします。詳細は「3-6 Objective-Cで拡張された機能」－「11. @propertyと@synthesize」参照。Objective-Cでは@で始まる文は通常の命令文ではなく、コンパイラへの指示を与えるものでコンパイラデイレクイブと呼びます。ボタンアクションに関しては「.h」でアクションメソッドのプロトタイプ宣言をし、「.m」でアクションメソッド本体を定義します。

Objective-Cではメソッドを以下のように「-」から書き始めます。メソッド名、引数名は「:」で区切り、それぞれ先頭に()で括って型を指定します。メソッドについては「3-6 Objective-Cで拡張された機能」－「10.　クラスメソッドとインスタンスメソッド」参照。

```
- (型)メソッド名:(型)引数名
{
}
```

・ViewController.h

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

@interface ViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *label1;　←プロパティ名の宣言
- (IBAction)Click:(id)sender;　←アクションメソッドのプロトタイプ宣言

@end
```

・ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
```

```
}

- (IBAction)Click:(id)sender {←アクションメソッド本体の定義
}

@end
```

「注」Xcode 4.3/4.4では以下のように@synthesizeを使ってプロパティ名とインスタンス変数名を関連づけます。この場合、インスタンス変数はプロパティと同じ名前が割り当てられます。従ってラベルを参照する際、プロパティを使う場合は「self.label1」とし、インスタンス変数を使う場合は「label1」とします。

```
@implementation ViewController
@synthesize label1; ←プロパティ名と同じ名前のインスタンス変数を関連づける
```

「.m」のviewDidUnLoadメソッドにViewの終了でラベルの参照を解放する処理の[self setLabel1:nil];が自動生成されます。プロパティ名label1の先頭を大文字にしたメソッド名が生成され（システムが管理する別な場所）、そのメソッドにnilを渡すことでlabel1の解放処理を行っています。

```
- (void)viewDidUnload {
    [self setLabel1:nil];  ←Viewの終了でラベルの参照を解放する処理
    [super viewDidUnload];
}
```

⑥ボタンアクションの記述

自動生成されているClickメソッド内に、ボタンをクリックしたときの処理を記述します。ここではラベル（label1）にテキストを表示します。下線部が追加したコードです。
ラベルのtextプロパティに表示する文字列を設定します。

```
- (IBAction)Click:(id)sender {
    self.label1.text=@"押したよ"; // インスタンス変数を用いるなら_label1.text
}
```

「注」Objective-CではNSString型の文字列オブジェクトを@””で表します。詳細は「4-5 NSStringクラス」を参照してください。

「注」Xcode 4.3/4.4においてはインスタンス変数を用いて「label1.text」のように記述している例がよくあります。これを、Xcode 4.5でそのまま使うとエラーとなります。何故ならXcode 4.5では_label1というインスタンス変数名が使用されているからです。従ってプロパティを使って参照するならselfを付けて「self.label1.text」とし、インスタンス変数を使って参照するなら「_label1.text」とします。

2. Outlet（アウトレット）

アウトレットとはStoryboadが管理するオブジェクト（主にUI要素）に付けたプロパティです。Xcode上でアウトレット接続を作成すると、その接続はStoryboadに保存されアプリケーション実行時にプロパティとインスタンス変数の間で相互通信ができるようになります。

アウトレットは@propertyコンパイラデイレクイブを使って宣言します。以下はlabel1という名前のプロパティを宣言しています。Storyboad上のラベルはプロパティ名のlabel1で管理することになります。

```
@property (weak, nonatomic) IBOutlet UILabel *label1;
```

プロパティ名と同じ名前のインスタンス変数を宣言するには@synthesizeコンパイラデイレクイブを使います。以下はlabel1という名前のインスタンス変数を宣言しています。

```
@synthesize label1;
```

以上の２つのコンパイラデイレクイブによりプロパティのlabel1とインスタンス変数のlabel1がアウトレット接続されることになります。これによりコード中からインスタンス変数のlabel1を使って「label1.text」とすることでStoryboadが管理するUI要素のラベル(プロパティ名label1)を参照することができます。

なお、@propertyコンパイラデイレクイブのパラメータの意味は以下の通りです。

・weak

storageの指定で、weakまたはstrongを指定します。weakは弱い参照を示し、strongは強い参照を示します。Single View Applicationテンプレートの場合のデフォルトはweakです。Master-Detail Applicationテンプレートの場合のデフォルトはstrongです。weakの場合は不要な循環参照(Strong refersnce cycle)が避けられ、Viewの終了で参照は自動的に解除されることになります。strongの場合はユーザが明示的に参照を解除する必要があります。ただしStoryboadでアウトレット接続した場合はweakでもstrongでもviewDidUnloadメソッド内に「 [self setLabel1:nil];」のようなコードが自動で生成されます。

・nonatomic

マルチスレッド環境で各スレッドからスレッドセーフで参照できるようにするのがatomicで、そこまで要求しないのがnonatomicです。

・IBoutlet

アウトレットであることを示す型修飾子です。IBはInterface Builder（現在のStoryboad）の意味です。

・UILabel

ラベルオブジェクトの型です。「UILabel *」はUILabelへの参照型を意味します。

3. Action（アクション）

StoryboadでAction接続すると「.h」ファイルと「.m」ファイルでコードが生成され、UI要素のアクションとアクションメソッドが接続されます。

「.h」ファイルで以下のようなプロトタイプ宣言を行います。

```
- (IBAction)Click:(id)sender;
```

「.m」ファイルで以下のような本体定義を行います。

```
- (IBAction)Click:(id)sender {
}
```

上の文の意味は以下の通りです。

・IBAction

Interface Builder(現在のStoryboad)側のUI要素からのメッセージに対応するためのメソッドであることを示す特別な型です。通常のメソッドの型のように戻り値の型ではありません。

・Click

アクションメソッドの名前です。

・id
　idはObjective-Cで扱う全ての総称型です。
・sender
　引数の名前です。アクションを起こしたオブジェクトの情報がsenderに渡されています。「((UIButton *)sender).tag」のようにしてオブジェクトの情報を取得することができます。引数の型がidという総称型なので、ボタンの場合は(UIButton *)でキャストする必要があります。

StoryBoadのAction接続で指定できるEventの代表的なものとして以下があります。
・Touch Up Inside
　オブジェクト内のタッチアップ動作で発生します。
・Touch Up Outside
　オブジェクト外のタッチアップ動作で発生します。
・Touch Down
　タッチダウン動作で発生します。
・Touch Down Repeat
　ダブルタッチのダウン動作で発生します。
・Did End On Exit
　キーボードのEnterキーが押されたときに発生します。
・Value Changed
　入力内容が変化したときに発生します。

1-9　オブジェクトとメソッド

２つのテキストフィールドに入力した値をボタンのクリックで加算し、結果をラベルに表示するプログラムを作ります。

テキストフィールド(TextField)を２つ配置しプロパティ名をtext1,text2とします。ラベルを２つ配置し、１つ目はテキストを「＋」とします。２つ目はプロパティ名をlabel1とします。ボタンを配置しテキストを「＝」とし、ボタンアクションをClickとします。

・ViewController.h

```
@property (weak, nonatomic) IBOutlet UITextField *text1;
@property (weak, nonatomic) IBOutlet UITextField *text2;
@property (weak, nonatomic) IBOutlet UILabel *label1;
- (IBAction)Click:(id)sender;
```

・ViewController.m

```
- (IBAction)Click:(id)sender {
    int a=[self.text1.text intValue];
    int b=[self.text2.text intValue];
    self.label1.text=[NSString stringWithFormat:@"%d",a+b];
}
```

Appleの「Objective-C 2.0プログラミング言語」マニュアルには以下のような記述がされています。
オブジェクトに何かを実行させるには、メソッドの適用を指示するメッセージをオブジェクトに送信します。Objective-Cではメッセージ式を[と]で囲みます。

[receiver message]

この書式は言い換えると、以下のようになります。

[オブジェクト メソッド:引数]

上のプログラムにおける[text1.text intValue]のtext1.textがオブジェクト、intValueがメソッドです。これはtext1.textで示すNSString型オブジェクトの値をint値に変換します。これは以下のようにしても同じです。

```
NSString *val1=text1.text;
int a=[val1 intValue];
```

先のプログラムにおける

[NSString stringWithFormat:@"%d",a+b]

のNSStringがオブジェクト、stringWithFormatがメソッド、@"%d",a+bが引数です。これは書式制御文字に従って文字列に変換します。%dが書式制御文字で、「,」以後の引数（この例ではa+b）の内容を10進整数値の文字列に変換します。

text1.textやval1は生成された動的オブジェクトであるのに対し、NSStringは改めて生成されることなく使用できる静的オブジェクトです。この場合のオブジェクト名はクラス名となります。

1-10　Simulatorの言語環境

1.　言語の設定

Simulatorの言語はデフォルトでEnglishになっています。このため画面の文字は英語で表示されます。またソフトキーも日本語入力できません。

日本語にする方法は、実機と同じで「Settings」アイコンを押し、General > International > Languageの順で進み、「日本語」を選択し「Done」ボタンを押して完了です。

アイコンの「Settings」が「設定」に変わりました。

ソフトキーも以下のように通常の英数字・記号の他に日本語対応になります。

2.　書式の設定

日付やカレンダーなどの書式を日本表示にしたい場合は「設定」アイコンを押し、一般 > 言語環境 > 書式の順で進み、「日本」を選択します。

1-11　アイコンのサイズ

iPhone,iPod touch,iPadのアプリにおいて標準的な見栄えを提供するためにはアイコンのサイズを目的に合わせて標準化する必要があります。

「iOS Human Interface Guidelines」の「Custom Icon and Image Creation Guidelines」では以下のようなアイコンサイズを定めています。

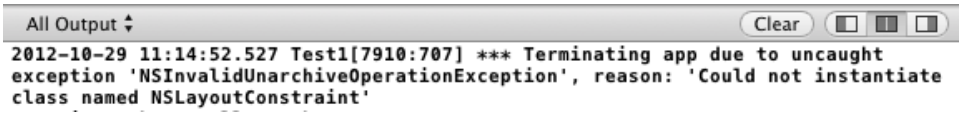
アイコンのサイズ（ピクセル）はiPhoneとiPod touch用とiPad用で別れます。さらにそれぞれが、Retinaディスプレイ対応用の倍のサイズのアイコンがあります。Retinaディスプレイ対応用のアイコンのファイル名はプライマリファイル名の末尾に「@2x」を付けます。

アイコンの種別	iPhone iPod touch	iPhone iPod touch のRetina対応	iPhone5の Retina対応	iPad	iPadの Retina対応	意味
Application icon	57×57	114×114	114×114	72×72	144×144	ホーム画面におくアプリケーションアイコン
Application icon	1024×1024	1024×1024	1024×1024	1024×1024	1024×1024	App Storeに載せるアプリケーションアイコン
Small icon (Spotlight)	29×29	58×58	58×58	50×50	100×100	Spotlight検索結果で表示するアイコン
Small icon (Settings)	29×29	58×58	58×58	29×29	58×58	設定画面で表示するアイコン、TableViewのセルのアイコン
Document icon	22×29	44×58	44×58	64×64	128×128	アプリで対応ファイルを開いた時にPOP内に表示するアイコン
Web clip icon	57×57	114×114	114×114	72×72	144×144	ホーム画面におくWebサイトへのリンクアイコン
Bar icon	20×20	40×40	40×40	20×20	40×40	Navigation Bars,Toolbarsに表示するアイコン
Bar icon	30×30	60×60	60×60	30×30	60×60	Tab Barsに表示するアイコン
Launch image	320×480	640×960	640×1136	768×1004	1536×2008	portrait表示での起動時画像（スプラッシュ画像）

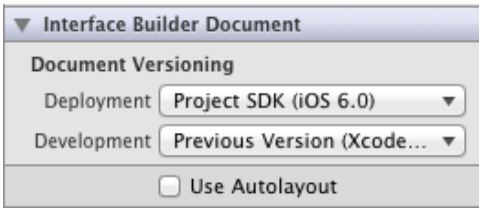
Launch image	-	-	-	1024×748	2048×1496	landscape表示での起動時画像（スプラッシュ画像）

1-12　Xcode 4.5でiOS 5のアプリを開発する場合

Xcode 4.5でStoryboardを使ってUI要素を配置した場合、これをiOS 5で動作させるためには「Use Autolayout」のチェックを外さないと、以下のような実行時エラーとなります。つまりiOS 5ではAutolayoutを行うためのNSLayoutConstraintクラスをサポートしていないとうエラーです。



Xcode 4.5を使ってiPhone 4/4S（iOS 5）用アプリを作る場合は「Use Autolayout」のチェックを外します。



「Use Autolayout」のチェックを外したことによる弊害として以下があります。

①Storyboardに最初に配置したUI要素がUIImageViewやUITextViewの場合、実行時のサイズがおかしくなります。場合によっては表示されないこともあります。

②最初のUIImageViewのサイズがおかしくなったことにより、このUIImageViewに対するタッチイベントを取得できなくなります。

以下のように２つのUIImageViewを配置したとします。

　これを実行すると以下のように最初のUIImageViewの高さは異様に小さくなってしまいます。

これを解決するためには以下の方法が考えられます。

・ImageViewのModeがデフォルトの「Scale To Fill」だと、イメージの縦サイズがおかしくなるので、「Top Left」などを指定します。

・一番最初にダミーのUI要素を配置し、非表示にしておきます。

・Storyboardで配置せずにコードで記述します。

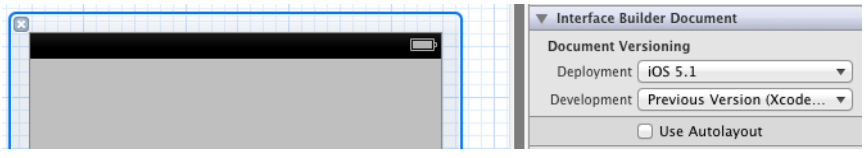
1-13　XIBファイル

XIBファイルはStoryboardが導入される前（Xcode 4.1以前）に使われていた古いインターフェースです。Xcode 4.2以後はStoryboardになり事実上使用されなくなりましたが、Xcode 4.5でAutolayout機能を導入したため、IOS 5用にStoryboardが使えない（Use Autolayoutのチェックを外すと各種不具合が出る）ので、iOS 5アプリ開発用に復帰しました。XIBファイルを使う場合は上のような弊害はありません。XIBファイルはiOS 6でも使用できます。

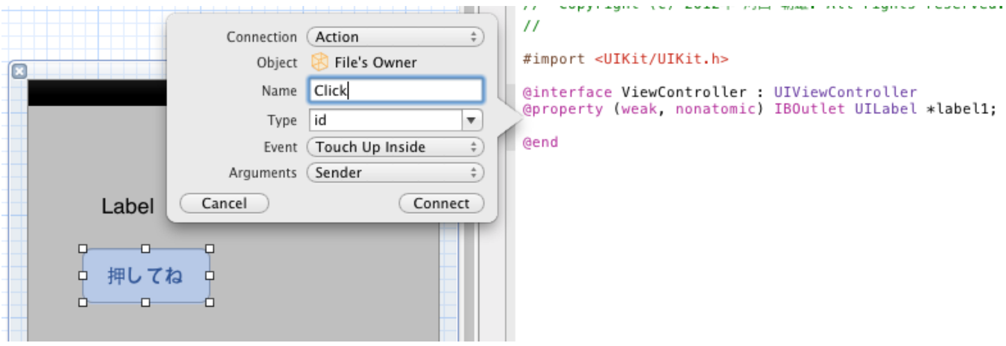
XIBのIBはInterface Builderの頭文字ですが、XはXcodeという説とXMLという説があります。プロジェクト生成時に「Use Storyboards」のチェックを外します。

ViewController.xibというXIBファイルが作成されます。

「Interface Builder Document」のDeploymentに「iOS 5.1」を選択し、Use Autolayoutのチェックを外します。



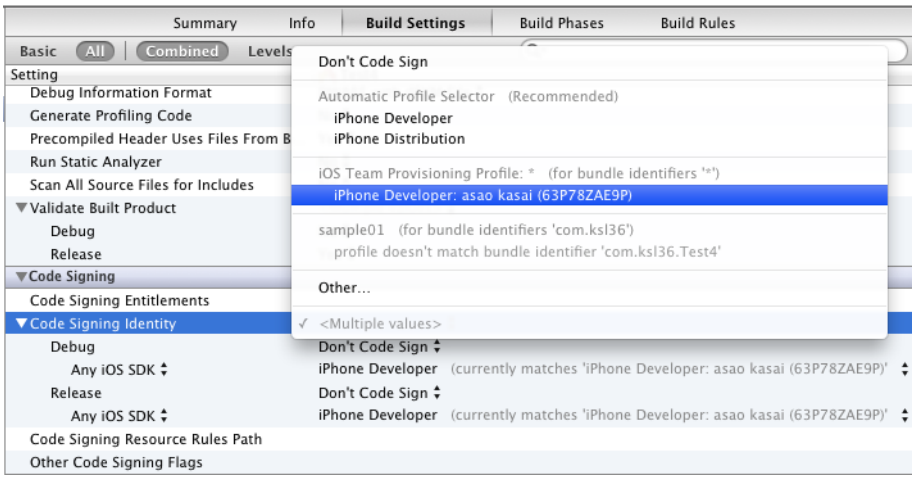
UI要素の配置方法やOutletとActionの接続方法はStoryboardと同じです。


Connection
Action
Object
File's Owner
Name
Click
Type
id
Event
Touch Up Inside
Arguments
Sender
Cancel
Connect
Label
押してね
//
#import <UIKit/UIKit.h>
@interface ViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *label1;
@end

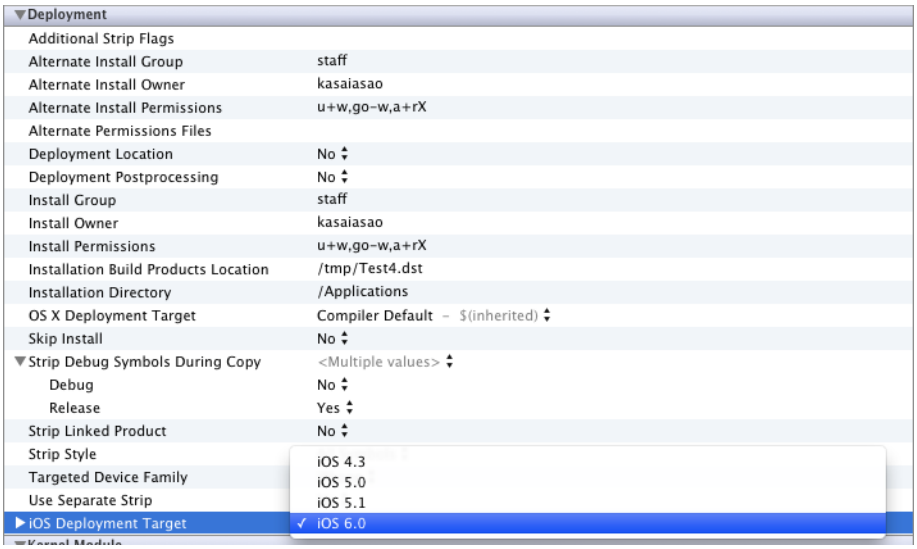
1-14　実機での実行

プロジェクトタグを選択します。

「Build Settings」タグを選択し、「Code Signing」の「Code Signing Identity」に登録されているiPhoneのIDを設定します。



「Deployment」の「iOS Deployment Target」にiPhoneのOSバージョンを設定します。



この設定で、Schemeの候補に実機の名前が登録されます。

著者略歴

河西 朝雄（かさいあさお）

山梨大学工学部電子工学科卒（1974年）。長野県岡谷工業高等学校情報技術科教諭、長野県松本工業高等学校電子工業科教諭を経て、現在は「カサイ．ソフトウエアラボ」代表。

「主な著書」

「入門ソフトウエアシリーズC言語」、「同シリーズJava言語」、「同シリーズC++」、「入門新世代言語シリーズVisualBasic4.0」、「同シリーズDelphi2.0」、「やさしいホームページの作り方シリーズHTML」、「同シリーズJavaScript」、「同シリーズHTML機能引きテクニック編」、「同シリーズホームページのすべてが分かる事典」、「同シリーズiモード対応HTMLとCGI」、「同シリーズiモード対応Javaで作るiアプリ」、「同シリーズVRML2.0」、「チュートリアル式言語入門VisualBasic.NET」、「はじめてのVisualC#.NET」、「C言語用語辞典」ほか（以上ナツメ社）

「構造化BASIC」、「Microsoft LanguageシリーズMicrosoft VISUAL C++初級プログラミング入門上、下」、「同シリーズVisualBasic初級プログラミング入門上、下」、「C言語によるはじめてのアルゴリズム入門」、「Javaによるはじめてのアルゴリズム入門」、「VisualBasicによるはじめてのアルゴリズム入門」、「VisualBasic6.0入門編、中級テクニック編、上級編」、「Internet Language改訂新版シリーズ ホームページの制作」、「同シリーズJavaScript入門」、「同シリーズJava入門」、「New Languageシリーズ標準VisualC++プログラミングブック」、「同シリーズ標準Javaプログラミングブック」、「VB.NET基礎学習Bible」、「原理がわかるプログラムの法則」、「プログラムの最初の壁」、「河西メソッド: C言語プログラム学習の方程式」、「基礎から学べるVisualBasic2005標準コースウエア」、「基礎から学べるJavaScript標準コースウエア」、「基礎から学べるC言語標準コースウエア」、「基礎から学べるPHP標準コースウエア」、「なぞりがきC言語学習ドリル」、「C言語　標準ライブラリ関数ポケットリファレンス[ANSI C,ISO C99対応]」、「 C言語　標準文法ポケットリファレンス[ANSI C,ISOC99対応]」ほか（以上技術評論社）

iPhone&iPadプログラミングBible[上]

2014年5月18日　初版　第1刷発行
著者：河西 朝雄
発行者：河西 朝雄
発行所：カサイ．ソフトウエアラボ
長野県茅野市ちの813　TEL.0266-72-4778

デザイン：河西 朝樹

定価＝1,620円（税込）