Xcode で初めてのアプリ

－アプリが作りたい学生のために その 2－

First App by Xcode - For the Students to make iOS Applications, Part. 2

奥 山 徹　　曽我部 雄樹
Tohru Okuyama　　Yuki Sogabe

概要

前回の報告[1]では、iPhone 等のアプリを開発するための環境である Xcode[2]の簡単な説明と、インストールの方法及び簡単な使い方を紹介した。今回は、簡単な iPhone アプリを例として、アプリ開発の基礎事項を報告する。

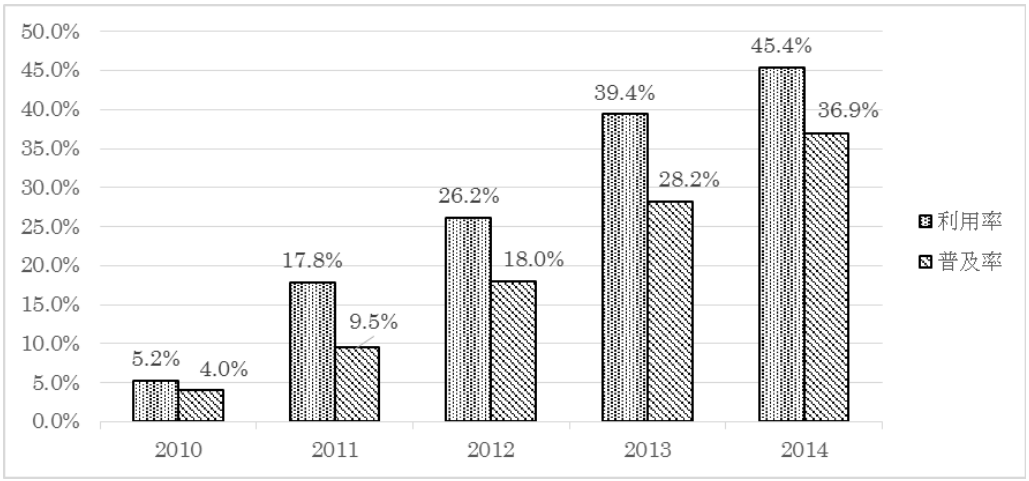
1. はじめに

スマートフォンやタブレット等の携帯情報端末は、現代人の必須の情報アイテムとして普及している。日本におけるスマートフォンの人口普及率は日経 BP コンサルティングの調査[3]によれば 36.9%（図1参照）であり、前年比 8.7%増である。図1に示す通り、人口普及率は年々増加している。なお、このデータは、人口動態による補正を加えてある。そのため、数値が他の調査より幾分、小さくなっている。例えば、ビデオリサーチインタラクティブの 2014 年2月の調査[4]では、54.0%（図 2 参照）となっている。調査方法や人口動態による補正のあるなしに関する詳細がわからないので、どちらの数値が正しいかは判断しかねる。しかしながら、現在のスマートフォンの普及率が 40～50%の間であることは間違いないと思われる。このように、既に国民の半数近いユーザを確保していることを考えると、インターネットのユーザサービスは PC を中心としたものから、PC＋スマートフォンへと移り変わっていくのは必然と考えられる。そのため、今後インターネット上のユーザサービス構築を目指す場合、通常の PC 向けホームページの他に、スマートフォン向けのホームページを作成することが必要となる。さらに、よりよいサービスを提供するためにはスマートフォンの専用アプリをリリースすることも視野に入れておかなければならない。そのため、スマートフォンのアプリ開発の技術（スマートフォンからのネットワークサービスへのアクセス方法等の開発を含む。）はインターネットサービスを提供するプログラマーにとって、必須のものとなっている。

本稿は、スマートフォンアプリ開発の初心者向けに行った講座の内容をまとめたものである。昨年の第一稿[1]では、具体的なアプリ開発の前準備の記述だけに終わってしまったが、今回以降、実際に講座で使った例を活用し、具体的なアプリ開発の方法について紹介する。なお、画面遷移やネットワーク接続等のより具体的なテクニックについては、次回以降に報告する予定である。

なお、対象とするのは iOS であるが、Android でも同様の講座を行っており、その内容についても報告

する予定である。



＊利用率は、性年代別に均等サンプリングした調査結果。
普及率は、日本の人口構成比を考慮して推定した数値。
2010～2012 年は 6 月調査で、2013～2014 年は 7 月調査

図 1. スマートフォンの普及率と利用率（人口動態による補正済）

（日経 BP コンサルティング調べ[3]）

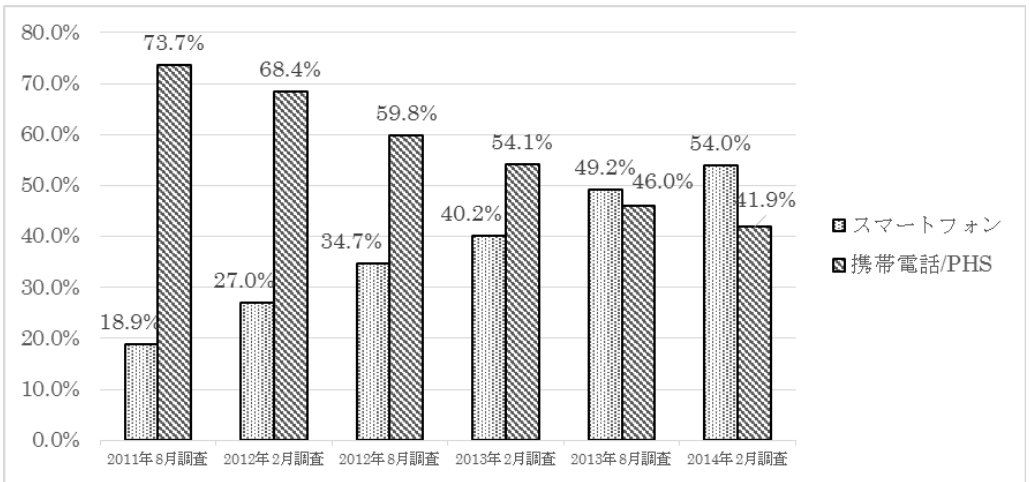


図 2. スマートフォンとフィーチャフォンの利用比率

（ビデオリサーチインタラクティブ調べ[4]）

余談ではあるが、調査結果について、特に人口動態調査による補正について、言及しておく。日本の人口構成は少子高齢化により高齢者側に大きく偏りがあることは周知の事実である。厚生労働省の人口動態調査による年齢別人口分布[5]によれば、60～64 歳の年齢層が人口のピークであり、40 歳から 70 歳が日本の人口構成の多くの部分を占めていることがわかる。そのため、統計調査の対象の母集団

を日本人の 10 歳から 80 歳とした場合、サンプリングする年齢構成を人口構成の比と整合するように補正してやる必要がある。ビデオリサーチインタラクティブの年齢別及び男女別の調査結果を図 3 に示す。この図から明らかなように、ビデオリサーチインタラクティブのデータは人口動態による補正をしているのは明らかであり、スマートフォンの普及率は日経 BP コンサルティングによる 36.9%のほうが妥当な数字と見なすべきである。何れにしても、まだ普及率が低い高齢者向けのアプリ開発は、今後の一つのマーケットプレイスとして考えておくべきである。

			スマートフォン	2013年8月調査との差（Points）	携帯電話/PHS	2013年8月調査との差（Points）
インターネットユーザー全体		21776	54.0%	4.8%	41.9%	−4.1%
男性計		11280	53.7%	4.1%	42.2%	−3.3%
	15～19才	871	77.8%	6.7%	21.0%	−6.1%
	20代	1873	72.9%	6.7%	27.4%	−6.6%
	30代	2352	62.9%	4.4%	37.0%	−3.7%
	40代	2417	53.7%	5.4%	44.9%	−2.5%
	50代	1960	41.4%	−0.6%	50.1%	−0.6%
	60代	1807	23.3%	3.1%	62.1%	−1.9%
女性計		10496	54.4%	5.5%	41.6%	−4.9%
	15～19才	827	84.8%	2.0%	19.2%	−6.5%
	20代	1829	83.1%	5.7%	21.5%	−6.9%
	30代	2330	64.4%	6.5%	36.0%	−4.1%
	40代	2352	45.8%	4.5%	48.6%	−4.1%
	50代	1764	37.9%	8.3%	52.3%	−5.9%
	60代	1394	17.6%	3.6%	65.2%	−2.8%

図 3. ビデオリサーチインタラクティブによる性・年齢別のスマートフォンの普及率

2. 入出力テストアプリ

最初に入力フィールドを定義して、そこで入力された文字列を出力する簡単なアプリの作成を行う。プログラミングの基本は、【入力】→【処理】→【出力】であり、もちろん【入力】が存在しないプログラムも存在する。しかし、【出力】の存在しないプログラムは、そもそも何の処理も実行しないプログラムと同じであり、実行させる必要すらない。したがって、【入力】と【出力】は、プログラムの基本中の基本と位置づけることができる。そこで、最初に【入力】と【出力】の基本を学ぶ次の課題について考える。

【課題1】 入力フィールドから入力されたテキストを適当に処理して出力するアプリを作成せよ。

プログラムの基本設計は、まさに【入力】→【処理】→【出力】の簡単な逐次処理である。まずは、Xcode の storyboard を使って、テキストの入力フィールドと出力フィールドを作成することから考える。

① Xcode を起動し、新しいプロジェクト（ここでは、図4に示したように「InputTest」という名前を付けている。）を作成する。Product Name のフィールドに名前を入力（図 4 の1の部分）し、必要事項を入力したら Next を押す（図 4 の 2 の部分）。

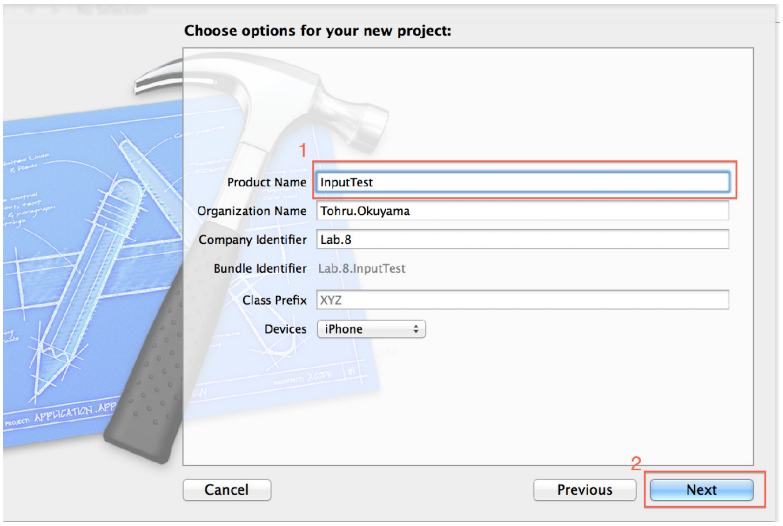


図 4.　新しいプロジェクト「InputTest」の作成。（テンプレートは Single View Application）

② テキストを入力するための入力フィールドを作成する。ここでは、オブジェクトライブラリから「Text Field」を使う。図 5 に示したように、1の部分（この部分は「ライブラリペイン」と呼ばれているが、本稿では単に「ライブラリ」あるいは「オブジェクトライブラリ」と呼ぶ。）で「Text Field」オブジェクトを選択し、2の部分にドラッグ＆ドロップする。すると、図に示すように入力のためのボックスを作成することができる。

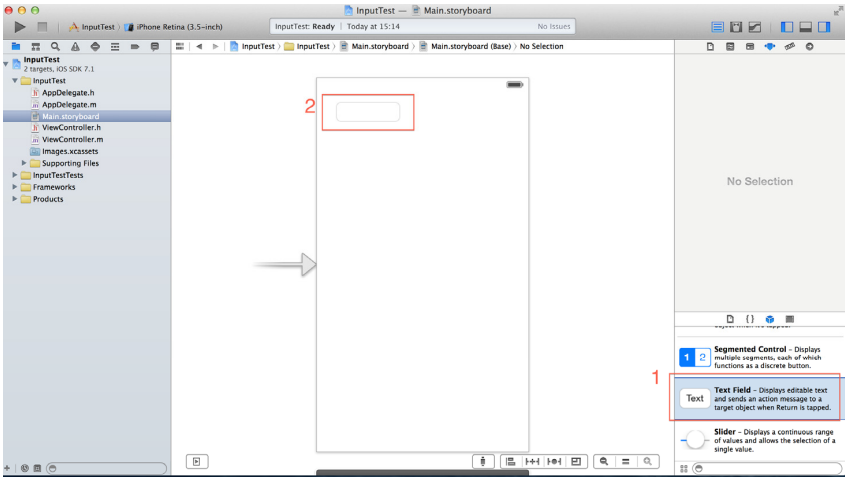


図 5. 入力のためのフィールドの生成

③ 処理したテキストを出力するための「Label」フィールドを作成する。「Label」は最も基本的な出力ボックスを生成するオブジェクトである。「Text Field」の場合と同様に、1の部分で「Label」のオブジェクト

を選択し、2の部分にドラッグ＆ドロップする。これで、テキストの出力フィールドを生成することができる。

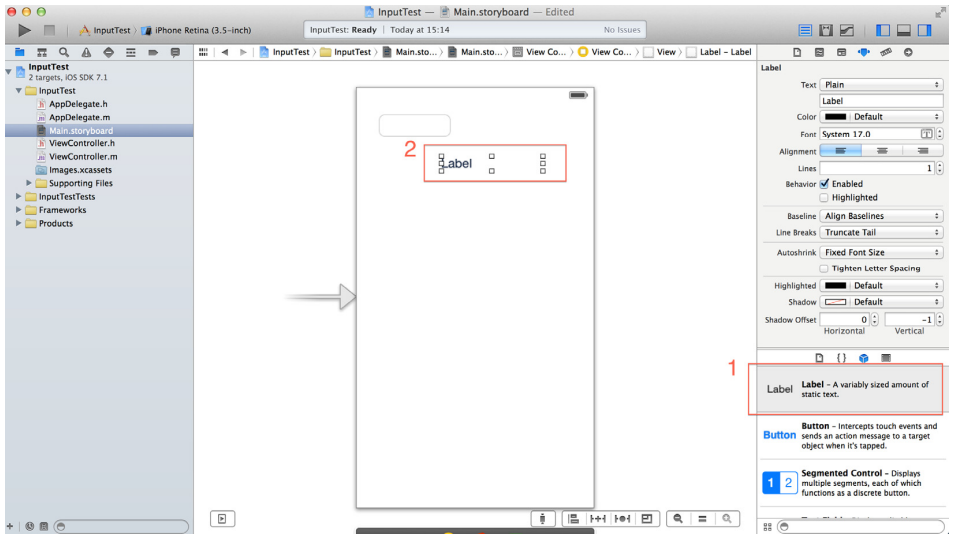


図 6.「Label」オブジェクトの生成

④　生成したボックスの属性を種々変更することができる。変更する場合は「インスペクタペイン」と呼ばれる領域（ライブラリの上にある部分、以下、単に「インスペクタ」と呼ぶ。）の各種パラメータを変更すればよい。図 7 の例では、Background パラメータを変更して、ラベルボックスの背景色を変更している。こうすれば、出力エリアの位置を明確にすることができる。なお、ボックス内の初期表示テキストが「空白」となっているのは、愛嬌と思ってもらいたい。

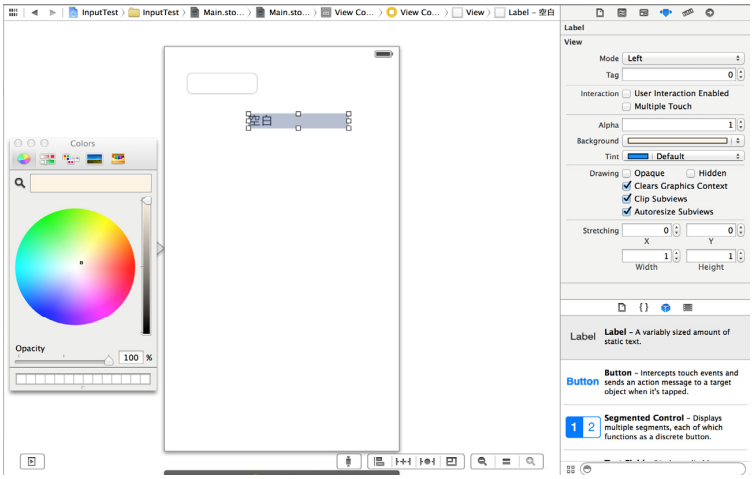


図 7. インスペクタによるオブジェクトの属性定義の例

以上で、必要とするオブジェクトの配置を終えたので、Xcode によりソースコードとこれらのオブジェクトを結合する。次節で結合の方法を示す。

3. オブジェクトとソースコードの結合（InputTest アプリの場合）

storyboard により追加されたオブジェクトはそのままで機能的に利用可能な状態となっているが、実際に各種のアクションを起こし（例えば、「Text Field」に値を入力する。）、入力された値をプログラム中で利用可能とするためには、プログラムのソースコードと結合してやる必要がある。図 8 は、「Text Field」オブジェクトを接続するソースコードの位置を示す。ここでは、ViewController.h というヘッダファイル内に、オブジェクトをプログラム内で利用するエントリを作成する。作成場所や作成方法は第一稿[1]に詳しく記述している。

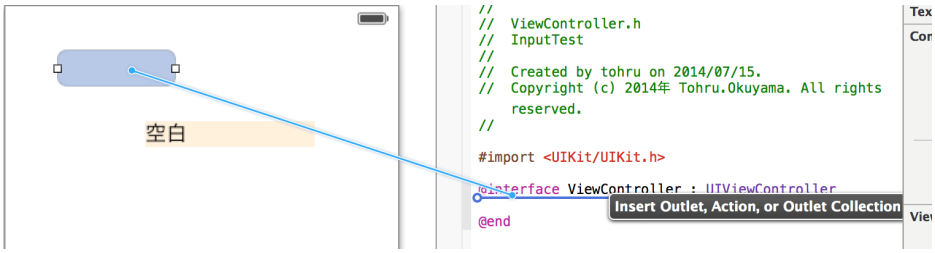


図 8.「Text Field」とソースコードの結合

図 9 に示す通り、「Text Field」のプログラム内でのオブジェクトのエントリ名（変数名）は「inputData」とした。また、「Label」のエントリ名は「labelField」とした。なお、いずれも Outlet として接続した。

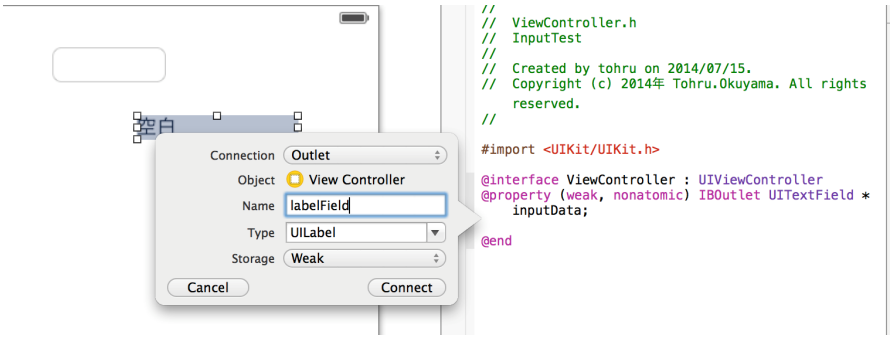


図 9.「Label」のソースコードとの結合

最後に、実際に処理するためのメソッドを作成する。前回「HelloWorld」プロジェクトでは処理のためのソースコードを最初にロードされ、実行される「ViewDidLoad」の中に記述したが、今回は、インターフェイスビルダー（IB）と連動するアクション（Action）としてメソッドを定義する。図 10 に示すように、

ViewController.h 内に新しい IBAction を定義する。図 10 に示す、「selector」はメソッドの名前、「(id)sender」は引数の受け渡しのための記述と思ってよい。(詳しい解説は、ここでは省略する。)

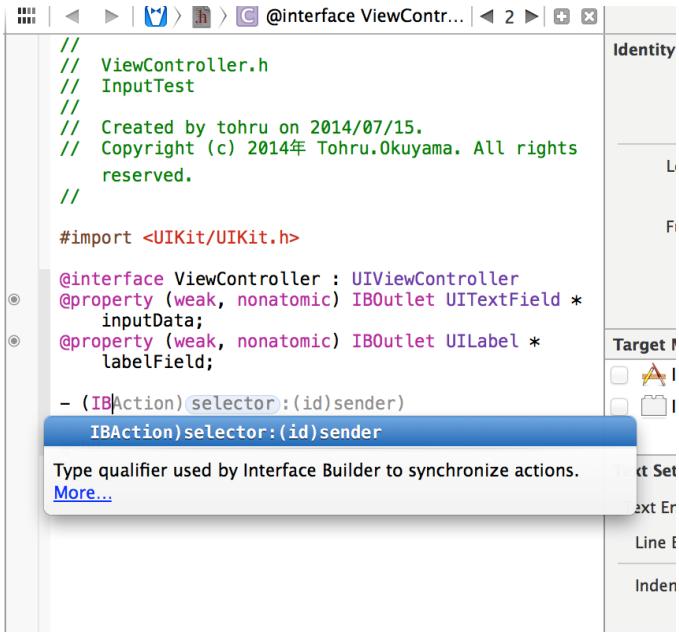


図 10. VeiwContoller.h における IBAction の定義

今回は、図 11 に示す通り、メソッド名を「inputFieldTest」とし、その他の部分はそのままとしておいた。

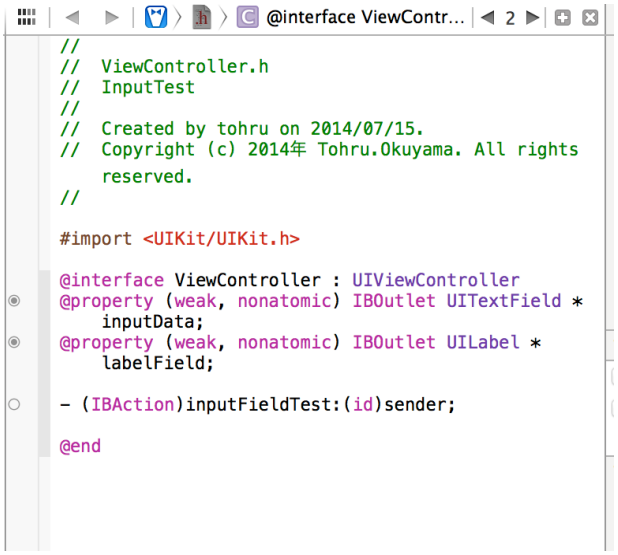


図 11. 新しいメソッドの定義

ボタンのように、それが押されるActionによって動作する場合は、ソースコードとボタンのActionとの接続は、前回示したように難しくない。しかし、入力フィールドのように、入力時のActionが一通りに限定できない場合は、作成したメソッドがどのようなActionにより起動するかを、プログラマーが定義してやる必要がある。

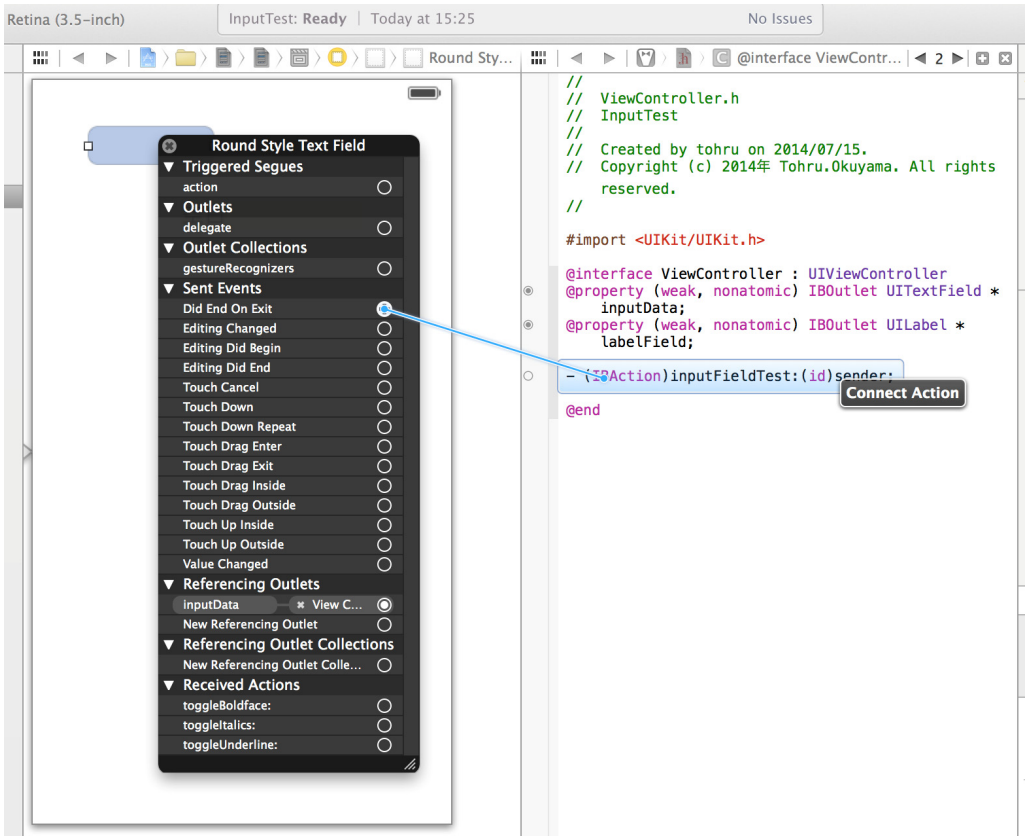


図 12. 入力フィールドとActionを記述したメソッドの接続

図12に示すように、storyboard上に配置した「Text Field」を右クリックすると、右側に示したように発生するイベント(=定義済 Action)の一覧が示される。今回は入力が終了したときに動作するようにしたい。そのため、「Did End on Exit」(Exitが発生したら終了する。)を選択し、そのイベントが発生した場合に動作するメソッドと結合する。実際の接続には「control」キーを押しながら接続するメソッド名の部分にドラッグしてやる。すると、図12に示すように「Connect Action」と表示されて、動作とメソッドが接続される。

接続がうまくいくと、図13のように、「Did End On Exit」イベントと関連づけられたメソッドが表示される。後は、このメソッドを実際にプログラムすればよい。メソッドの外形は、ViewContoroller.m 内に既にでき上がっているので、その中に、次の1行を追加する。

[_labelField setText:_inputData.text];

実際のプログラムの挿入箇所を図 14 に示す。

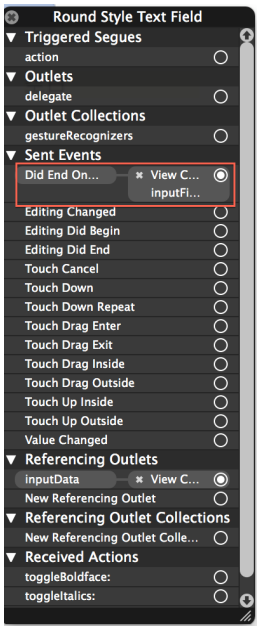


図 13.　メッソッドとの結合結果

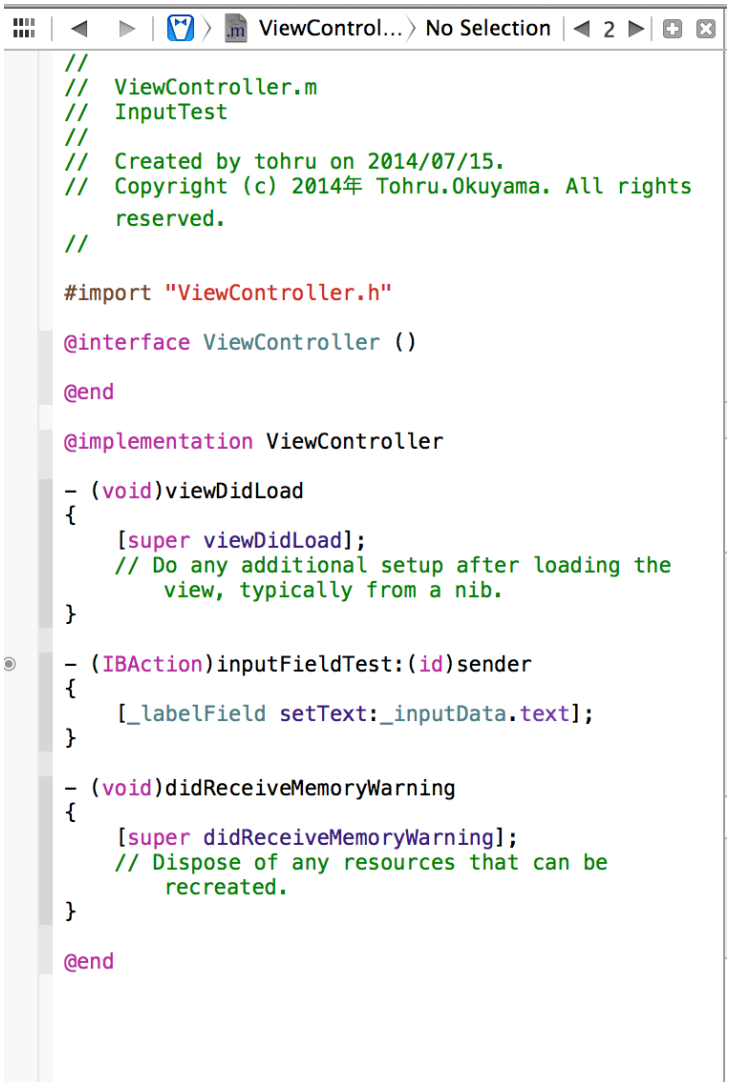


図 14. 新しいメソッドの挿入場所

プログラムの内容は簡単である。「_inputData」は入力フィールドに設定された名前であり、ここに入力されたデータが格納される。これは変数に見えるが、実際には、あるインスタンスの値を入れたオブジェクトである。そのため、実際の値を入れるためのサブフィールドが多数存在する。例えば、プログラム中の「_inputData.text」の「.text」というサブフィールドに入力されたテキストが保存される。「Label」オブジェクトに定義された「SetText」メソッドは「Label」のテキストサブフィールドに「:」の後ろのテキストを代入する

ことになる。「Label」の場合、テキストサブフィールドにテキストが代入されるとそれが表示される。

実際に、ビルドして実行した結果を図 15 に示す。①は、エミュレータが起動された直後の画面、②は、テキストフィールドにテキストを入力している画面、③は、結果がラベルフィールドに表示された例である。実際のデータ入力には、画面上にソフトキーボードが表示され、そのキーボードを使って入力することができる。

①　②　③

図 15. エミュレータによるアプリの実行例

実際の講座では、一つのテーマを終えると、それに関連する練習問題を解かせる。この例では、以下の 2 つの設問について、プログラムさせた。いずれも、簡単な例なのですぐにプログラムできると思っていたが、実際にはかなり時間が掛かった。コンピュータプログラムは小さなミスでも許してくれない。そのため、コーディング時のエラー、ビルド時のエラー、実行時のエラー（思い通りの実行結果が得られない。）等が多々発生し、それらのエラー（いわゆる、「バグ」と呼ばれるもの。）を除去する作業に時間を取られ、プログラムの実行まで進まないのが現状であった。つまり、単にプログラムの方法論を教えるだけではダメで、エラーメッセージの見方や具体的なデバッグ（「バグ」を取ること。）の方法を総合的に教えなければならない。

1. 入力文字列が複数になるような入力フィールドを作成し、テストしてみよ。
2. 最初に入力した文字列を覚えておいて、次に入力した文字列と適当に連結するプログラムを作成せよ。 例えば、「I」が最初の文字で、「you」が二番目とすると、「I love you.」と表示することを考える。「you」の代わりに、「AKB48」とすると、当然、「I love AKB48.」 とならなければな

らない。（「I」を「He」に代えても「loves」とはならない。これをプログラムするには別の要素を考える必要がある。）

4. 練習問題のプログラム例

実際のプログラムを以下に示す。これは、プログラムの一例であって、やり方は他にもあるので、検討してほしい。 なお、図 16 に示した練習問題 2 の実行例は、Xcode 6 でのプログラム例となっている。そのため、前半で紹介した Xcode 5 を使った場合（図 15 参照）と、使用しているエミュレータの種類やインターフェイスの作りが異なっているので注意してほしい。

【練習問題1】の回答例は省略。

【練習問題2】の回答例は以下の通り。

ソースファイル１　-ViewController.h

```
//
//  ViewController.h
//  Inp
//
//  Created by tohru on 2014/07/15.
//  Copyright (c) 2014 年  Tohru.Okuyama. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
@property (weak, nonatomic) IBOutlet UITextField *InpData;  // ←TextField の定義
@property (weak, nonatomic) IBOutlet UILabel *OutData;  // ←Label の定義
@property NSString *memory;
        // ←文字列を記憶する文字列オブジェクトへのポインタの定義
@property int count;  // ←入力回数をカウントするカウンタ変数の定義

- (IBAction)InputTest:(id)sender;  // ←入力処理のためのメソッドの定義

@end
```

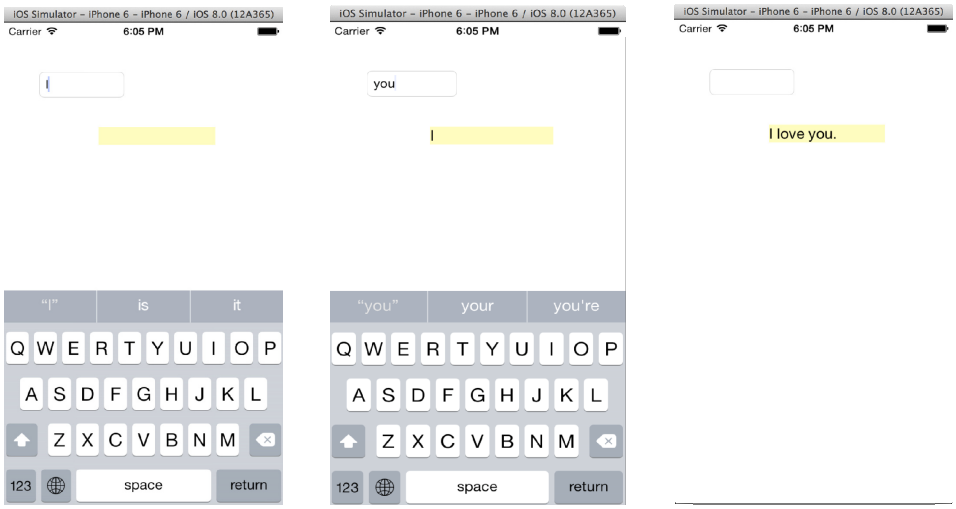
ソースファイル２　-ViewController.m

```
//
//  ViewController.m
//  Inp
//
//  Created by tohru on 2014/07/15.
//  Copyright (c) 2014 年 Tohru.Okuyama. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

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

    _count = 0;  // ←カウンタ変数の初期設定
}

- (IBAction)InputTest:(id)sender  // ←処理メソッド本体の宣言
{
    if (_count == 0)  // ←入力回数のチェック（0回は初期状態）
    {
        _memory = _InpData.text;  // ←入力文字列を記憶用の文字列オブジェクトに代入
        _count++;  // ←カウンタ変数のインクリメント（一つ増加させる）
        [_InpData setText:@""];  // ←入力データオブジェクトのテキスト部分の初期化
    } else {
        _memory = [_memory stringByAppendingString:@" love "];  // ←「love」を結合
        _memory = [_memory stringByAppendingString:_InpData.text];
                                // ←二番目の文字列を結合
        _memory = [_memory stringByAppendingString:@"."];  // ←  「.」を結合
        _count = 0;  // ←カウンタ変数の初期化
        [_InpData setText:@""]; // ←入力データオブジェクトのテキスト部分の初期化
    }
    [_OutData setText:_memory]; // ←ラベルへのデータの出力
}
```

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

@end
```

このプログラムの一つ目のポイントは、文字列を記録するための文字列オブジェクト（memory）と入力回数をカウントするカウンタ変数（count）を public 変数として「ViewController.h」の中で定義していることである。プログラムでは、カウンタ変数を使い1回目の入力が 2 回目の入力かを判断している。そして、1 回目の入力の場合、そのまま memory に代入し、2 回目の入力の場合、memory に ″ love ″ という文字列を加え、その後、2 回目の入力を加え、最後に ″.″ を加えている。そして、最後に memory の内容を「Label」にコピーして表示している。

図 16 は実行時の画面のスナップショットである。①は 1 番目の文字列「I」と入力したところであり、②は 2 番目の文字列である「you」を入力しているところである。2 番目の文字列を入力し、Enter キーを押すと、③のような結合した文字列が表示される。



①　②　③

図 16. 練習問題 2 の実行画面の例

5. 終わりに

以上、簡単なプログラム例により、Xcode による初級プログラミング、特に storyboard（Interface Builder：IB）への部品（オブジェクト、要素）の配置とプログラムのソースコードとの結合の方法について

記述した。これにより、IB を使ったプログラムの要点は理解できるはずである。後は、ひたすらプログラムを書いてみて習熟してほしい。

なお、前回も紹介したが、Apple では、処理系を Objective-C から Swift に変更しており[6]、新しく初める初学者は Swift を使うべきである。Swift については、稿を改めて紹介したいと考えている。

文献など

[1] 奥山徹、曽我部雄樹、「Xcode ことはじめーアプリが作りたい学生のために その1」、『情報学研究』、朝日大学情報教育研究センター、第 23 巻、1-16 頁、2014 年 3 月.

[2] Apple Developer, “Xcode - The complete toolset for building great apps.”.
URL: https://developer.apple.com/xcode/

[3] 日経BPコンサルティング、「スマートフォンの国内普及率は36.9% スマホ満足度はソフトバンクが4年連続 1 位、携帯電話を含めるとKDDI(au)が 8 年連続 1 位に！」、『日経 BP コンサルティング ニュースリリース』、2014 年 8 月.
URL: http://consult.nikkeibp.co.jp/news/2014/0829sp/

[4] ビデオリサーチインタラクティブ、「スマートフォン所有率は、2 年前の 2 倍、過半数の 54.0%に。僅かではあるが、はじめて女性のスマートフォン所有率が男性を上回る。」、『ビデオリサーチインタラクティブ プレスリリース』、2014 年 2 月.
URL: http://www.videoi.co.jp/release/20140225.html

[5] 厚生労働省、「平成25年(2013)人口動態統計(確定数)の概況」、2014 年 9 月.
URL: http://www.mhlw.go.jp/toukei/saikin/hw/jinkou/kakutei13/index.html

[6] S. Sekine、「アップル、新プログラミング言語 Swift を発表。レガシーを廃して高速化した iOS/OS X 開発用」、2014 年 6 月. URL: http://japanese.engadget.com/2014/06/02/swift-ios-os-x/

奥山 徹 （経営学部経営情報学科教授）
曽我部 雄樹 （経営学部ビジネス企画学科講師）