

第一回 InterfaceBuilderと手書きと

Toru.Inoue@KISSAKI.tv

Agenda 1/9

- 今日やる事
 - 0.今後やる事の共有
 - 1.Xcodeで開発できる環境を整える
 - 2.InterfaceBuilder(以下IB)を使って簡単アプリ作り
 - 3.IBを使わず地獄のアプリ作り
 - 4.上記2と3の比較、ちょっと考え方
 - 5.プログラムの保存
 - 6.LookBack、次回予告と質問受付先について

Agenda2/9

- 0.今後やる事
 - 講義は隔週で、土曜に行います。
 - 詳しい事は運営から都度アナウンスがあると思うぜ！
 - 講義の形式について
 - 毎回、Agenda(目次と予想),Drive(実行),LookBack(振り返り)の3フェーズくらいで構成します。2h。長いな。トイレ行っていい？
 - 宿題とか欲しい？ 正直課すのが~~メンディ~~大変疲れる難しいのですが、質問はいつでも受け付けます。

techhubteach@googlegroups.com



Agenda3/9

- 0.今後やる事

- 講義の対象

- ど素人向けです。

- プログラムの「プ」の字を知らない人にも

- それ以外の人にも、**詰め込み型+a**の伝え方をします。

- たぶん、ココでしか聞けない内容です。xxxが〇〇なのはTechHubだけ！

- 講義の指針、目標

- iPhoneアプリのプログラムを通じて、

- スゲー簡単なiPhoneアプリのつくりかた、

- プログラマに**なりたくない**ても知っててほしい”仕事”の事、

- 等が学習できるようにするつもりです。

Agenda4/9

- 0.今後やる事
 - 講義Before&After
 - Before
 - なにもわからない
 - 他の言語ならわかる
 - 我流で頑張ってみた
 - After
 - 簡単なアプリが作れる
 - とりあえず他のプログラマと組んで仕事できる
 - プログラムについての勉強のしかたが何となく判るようになる
 - C言語などの基礎的な言語の勉強が必要になる
 - HTTPなどの基礎的Webテクノロジーについて勉強が必要になる
 - テストについての勉強が必要になる
 - 他のプログラマともっと効率よく組むための勉強が必要になる
 - etc

Agenda5/9

- 1.Xcodeでの開発
 - Xcodeてなに?
→開発環境アプリケーションで、
iPhoneのアプリが簡単に作れます。
 - 今回は、先ず、
今後の開発ができるように準備、調整します。

Agenda6/9

- 1.Xcodeでの開発
 - インストールしていない人！
 - ディスク貸すからDoしてください。
 - インストールしてるあいだ、隣のひとに見せてもらいましょう！ ククク教科書忘れた状態。。
 - あまつさえ、MacBookAir(新型)の人
 - →没収。

Agenda 7/9

- 1.Xcodeでの開発
 - インストールが終わったら
 - 起動して
 - DockにXcodeのアイコンが出ると思うので
 - よく使うから消えないようにしましょう

Agenda8/9

- 1.Xcodeでの開発
 - Objective-Cについて参考になる書籍

マジ俺無関係なので、紹介しても俺には一銭も入らない事に今気づいたが別にいいや、、

Objective-C逆引きハンドブック／林 晃さん

<http://www.amazon.co.jp/Objective-C逆引きハンドブック-林-晃/dp/4863540515>

詳説Objective-C 2.0／萩原 剛志さん

<http://www.amazon.co.jp/詳解-Objective-C-2-0-萩原-剛志/dp/4797346809>

→手前に用意しました。私物なのでカバー無いですけど。

Agenda9/9

→こっから先は、実技です。

Let's drive!

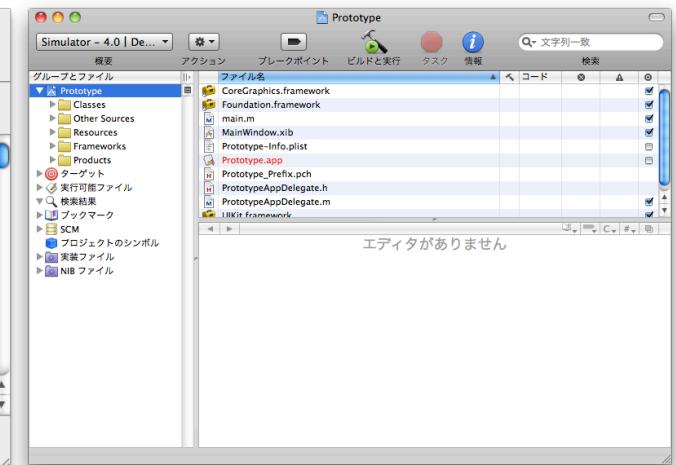
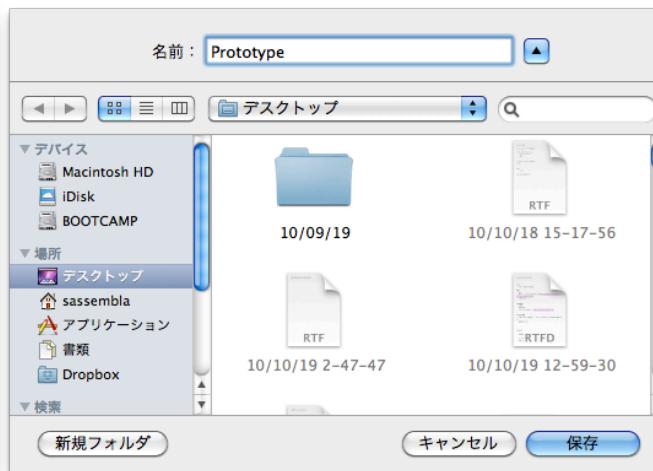
2.InterfaceBuilderを使って簡単アプリ作り

Drive1/34

2-0-0:新規プロジェクトを作つてみる。
プロジェクト名は、「Prototype」にしましよう。
大文字、小文字間違えないように。



ファイル>新規プロジェクト
一番上→右下の項目を選択!



この画面を、特に良く使います！
プロジェクトウインドウ(PW)と
呼ぶことにします。

Drive2/34

2-0-1:起動してみよう

⌘+Enter か、



真っ白なものが映るiPhoneが出てくる筈。



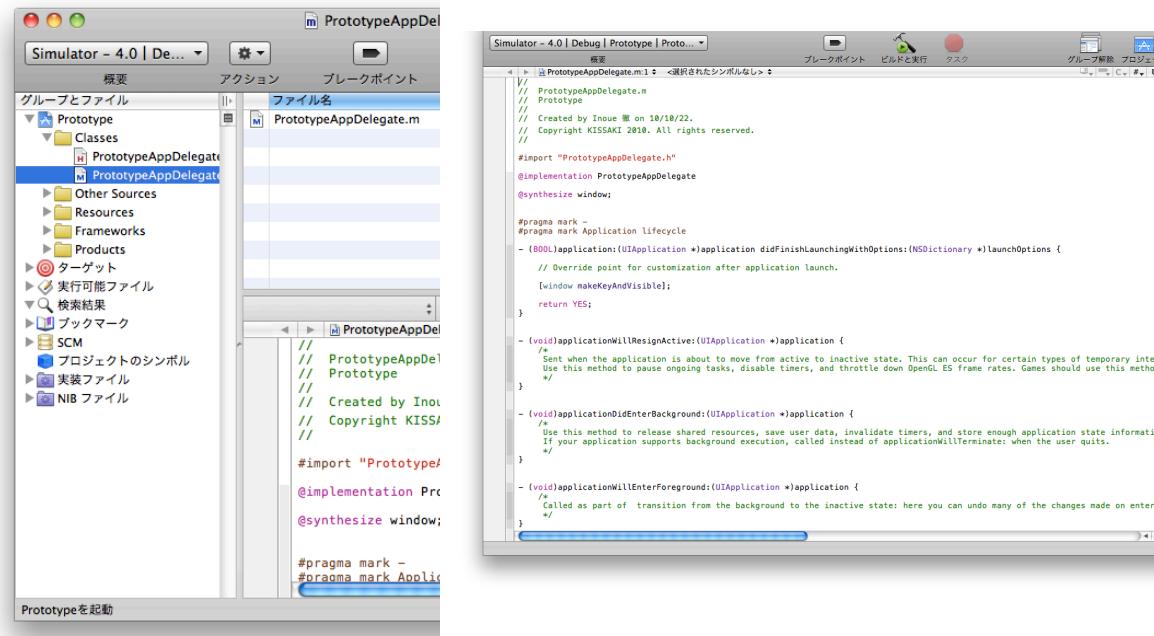
ちなみにこういう、Macの中で動く↑みたいな偽iPhoneを、Emulator(偽物、模倣するヤツ)といいます。

分からぬ英単語は出てくる度に辞書牽くとか書留めする事をお薦めします。

Drive3/34

2-0-2:ログを出そう(0)

プロジェクトウインドウ(以下PW)>Classes>PrototypeAppDelegate.mを開く、
そしてちょっとプログラムを書き込んでみよう。



```
// PrototypeAppDelegate.m
// Prototype
//
// Created by Inoue 徹 on 10/10/10.
// Copyright KISSAKI 2010. All rights reserved.

#import "PrototypeAppDelegate.h"

@implementation PrototypeAppDelegate

@synthesize window;

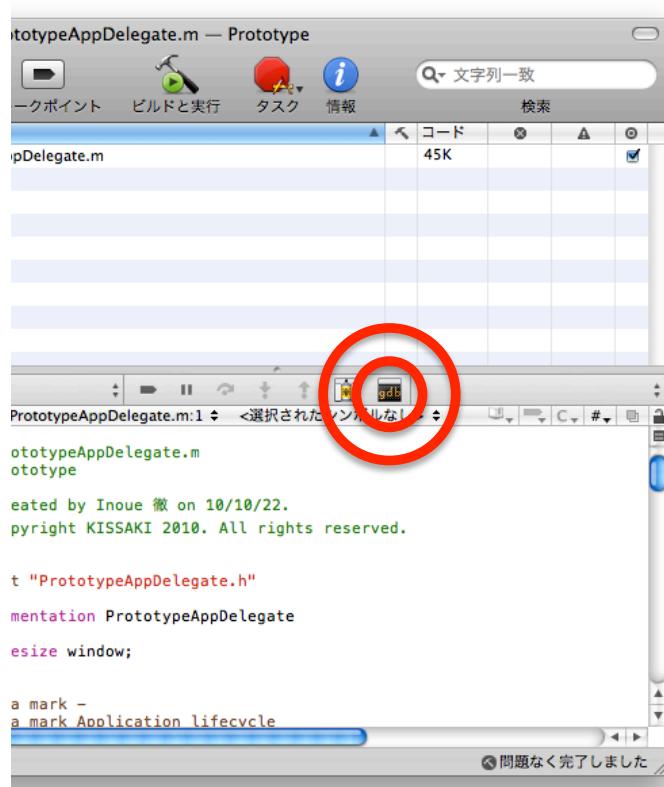
#pragma mark -
#pragma mark Application lifecycle

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    NSLog(@"%@", @"むにやむにや");
}
```

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
 NSLog(@"%@", @"むにやむにや"); // (0)

Drive4/34

2-0-3:書き加えたら、⌘+Sで保存、その後起動！

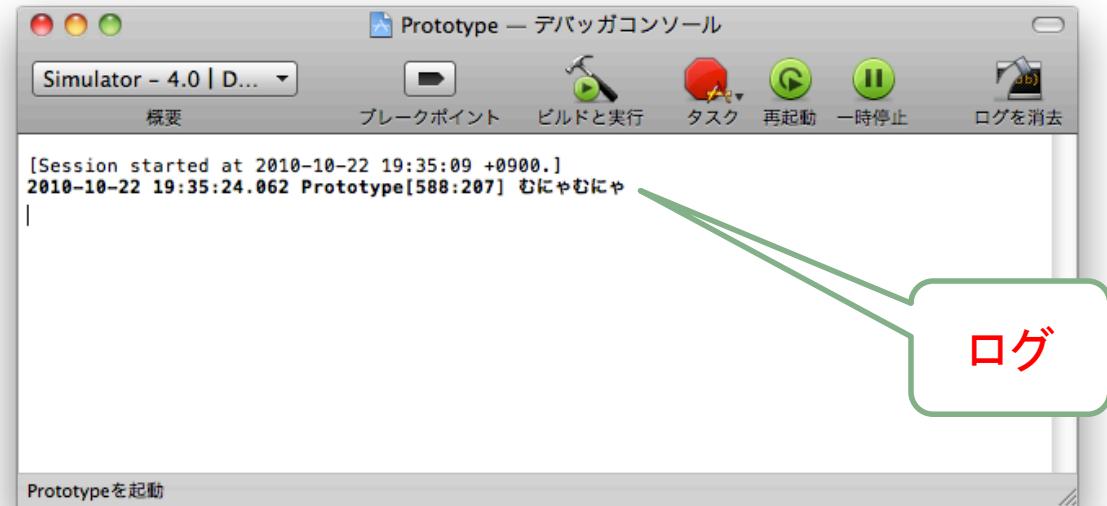


```
PrototypeAppDelegate.m — Prototype
[...] AppDelegate.m
[...]
PrototypeAppDelegate.m:1 ⇩ <選択されたソースが見つかりません! ⇩
PrototypeAppDelegate.m
Prototype
[...]
Copyright KISSAKI 2010. All rights reserved.

#import "PrototypeAppDelegate.h"
@implementation PrototypeAppDelegate
- (void)resizeWindow;
[...]
```

起動したら、PWのここをクリック。
起動してないと表示されない。

デバッガコンソール



```
Simulator - 4.0 | D...
概要 ブレークポイント ビルドと実行 タスク 再起動 一時停止 ログを消去
[Session started at 2010-10-22 19:35:09 +0900.]
2010-10-22 19:35:24.062 Prototype[588:207] むにゅむにゅ
|
Prototypeを起動
```

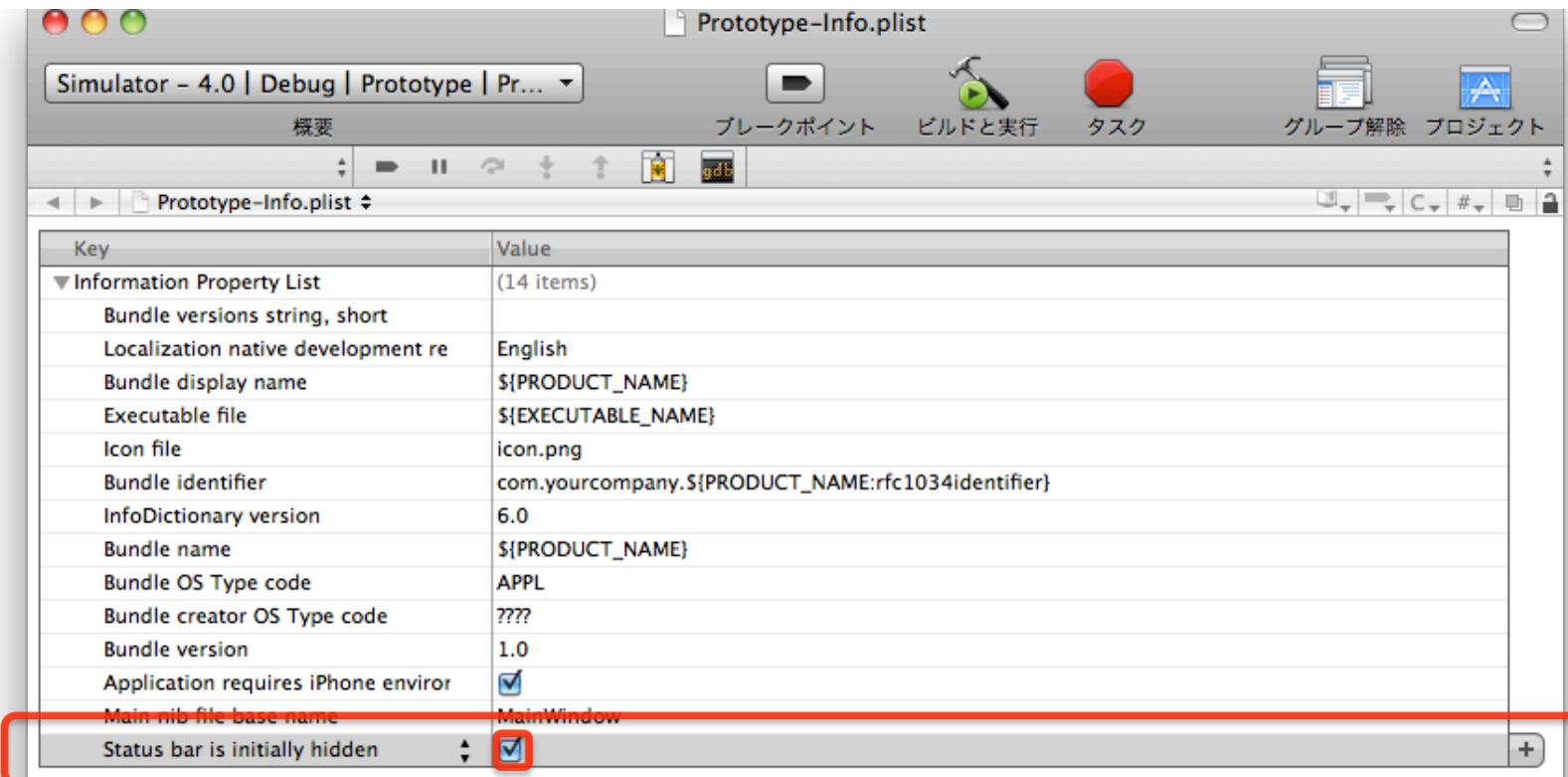
さっき書き込んだ文章が出たと思われ。
この、**デバッガコンソール**と**ログ**は、
アプリケーションがどんな感じに動いてるのか、
知るためにメチャメチャ使えます。

名前と見方と出し方は覚えてしまいましょう。
→暇だったら文章の内容を変えてみよう！

Drive5/34

2-1-0: アプリ上部のバーを消そう (1)

このバーは、実は凄く特殊な代物で、アプリケーションの設定から消す。



プロジェクトウィンドウの中のPrototype-Info.plistを開き、欄の右端にカーソルを持っていくと、+アイコンが現れる。候補から、Status bar is initially hidden を選択。
チェックボタンにチェックを入れて、保存して起動しよう。バー消えた？

Drive6/34

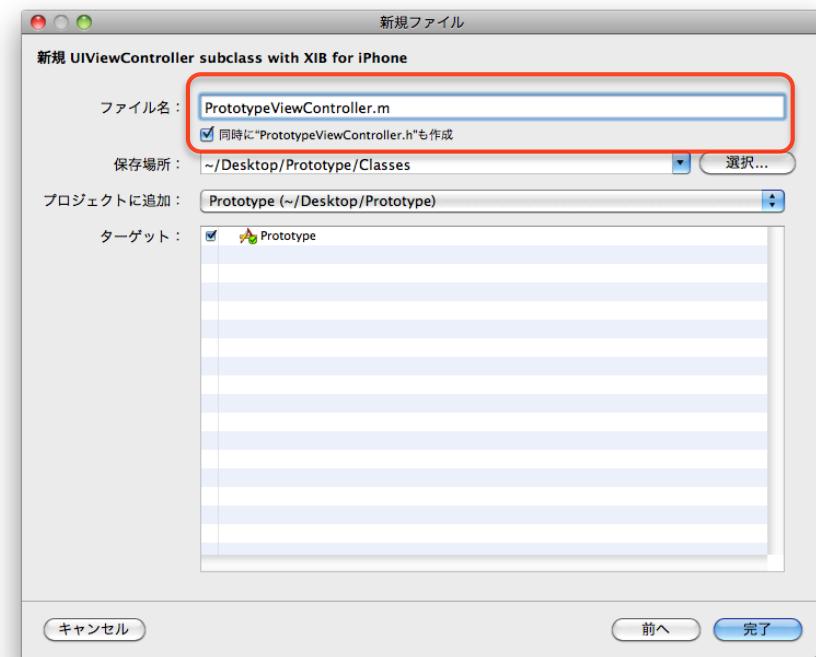
2-1-1:インターフェースビルダー(IB)で画面作り

→ViewControllerを作つてみよう。

PW>Classesを右クリック>追加>新規ファイル>



xibを追加、にチェックを入れる。
(英語かも。)

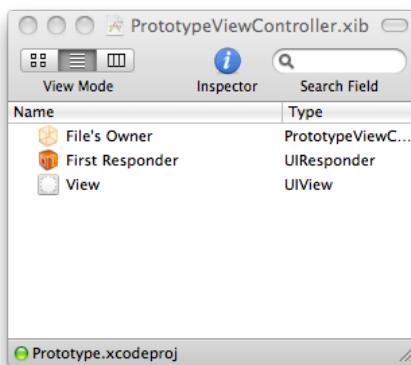


名前は「PrototypeViewController」で。

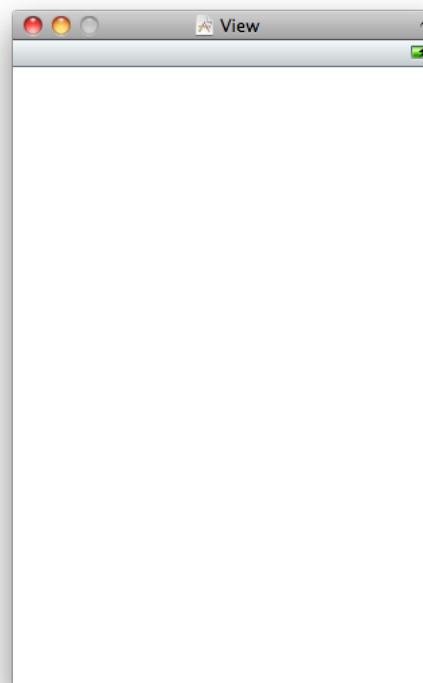
Drive7/34

2-1-2:PrototypeViewController.Xibをダブルクリック、IBが起動
こんな感じのものどもが開くと思う。

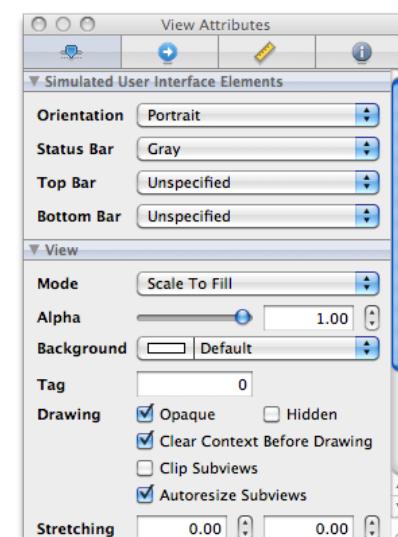
PrototypeViewController.Xib



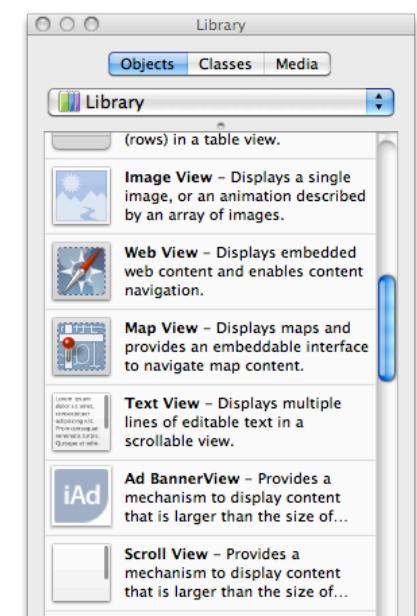
View



View Attributes



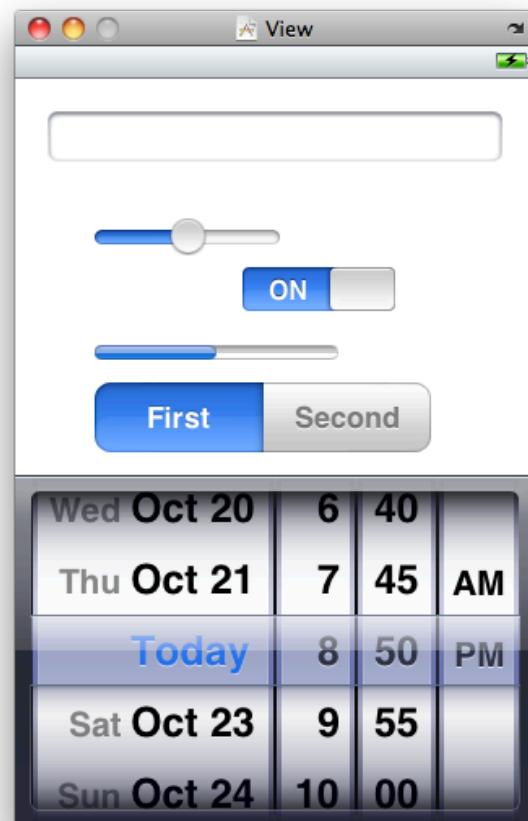
Library



で、文字を入力するTextFieldをViewに入れてみよう。ここは実演。
Library→Viewへと、ドラッグアンドドロップ。

Drive8/34

2-1-3:こんな感じにしてみた 後悔はしていない



なにこれ、、、

あ、TextFieldだけは最低でも一個付けてね！
で、TextFieldの恥じっこ端っこを、両端とも
画面の端に近づけておいてね。

こいつを、今のアプリケーションに
表示してみよう。

Drive9/34

2-1-4:プログラム書きます

PW>Classes>PrototypeAppDelegate.mを開いて、下記を追加。保存しよう。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions: ~~ {  
    NSLog(@"むにやむにや");  
    prototypeViewController = [[PrototypeViewController alloc] init];//(2)  
    [window addSubview:prototypeViewController.view];//(3)
```

Drive10/34

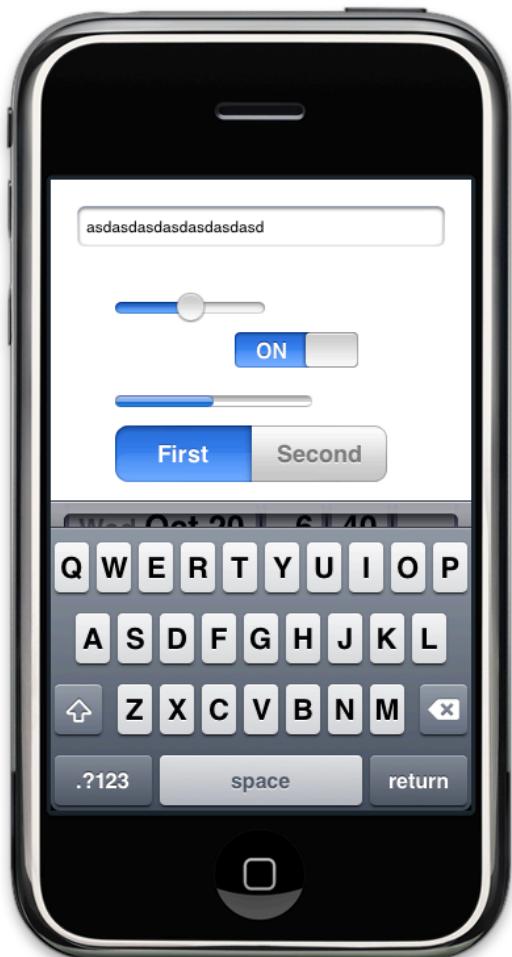
2-1-5:もうちょいプログラム書きます

PW>Classes>PrototypeAppDelegate.hを開いて、下記を追加。保存しよう。

```
#import <UIKit/UIKit.h>
#import "PrototypeViewController.h"//(4)
@interface PrototypeAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow *window;
    PrototypeViewController * prototypeViewController;//(5)
}
```

Drive11/34

2-1-6:起動してみよう。



、、、これはひどい。
うまく出ましたね。
作った通り。

TextFieldをクリックしてみよう。
キーボードが出てきて、文字を打ち込める筈。
他の部品も動く筈。

→なんかダメボな人。ウインドウの右下が
こんな感じの筈。

```
prototypeViewController = [[PrototypeViewController alloc] init];//(2)
[window addSubview:prototypeViewController.view];//(3) ⓘ 'prototypeViewContr...
[window makeKeyAndVisible];
return YES;
}
```

Build failed (1 個のエラー)

失敗 ① 1

Drive12/34

2-1-7:凄く端折ったエラーの見方

The screenshot shows the Xcode interface with the file `PrototypeAppDelegate.m` open. The code is as follows:

```
// Copyright KISSAKI 2010. All rights reserved.  
//  
#import "PrototypeAppDelegate.h"  
  
@implementation PrototypeAppDelegate  
  
@synthesize window;  
  
#pragma mark -  
#pragma mark Application lifecycle  
  
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)options  
{  
    NSLog(@"むにやむにや");//(0)  
    prototypeViewController = [[PrototypeViewController alloc] init];//(2)  
    [window addSubview:prototypeViewController.view];//(3) ①'prototypeViewContr...  
  
    [window makeKeyAndVisible];  
  
    return YES;  
}
```

A red exclamation mark icon in the gutter indicates a warning or error at line 3, column 1. The status bar at the bottom left says "Build failed (1 個のエラー)".

人生初エラーおめでとう。

プログラム書くウインドウの中に
どこがエラーか、

下のバーにエラーの個数が何個、
とか書かれてる筈。

Drive9,10を見直してみよう。
だいたいスペルミスとか、
スペースが入ってないとか、
そんなだ！

一寸自分にガッカリする位の理由だ！

Drive13/34

2-2-0:エミュレータを回転させてみよう
⌘(かcontrolか何か)+上下左右キーで端末が回転するぞ。



なんということでしょう。

ドラゴンボールが全部そろったかのような気分。
ただ、中身が、iPhoneが回転してもそのままな筈。これは切ない。

Drive14/34

2-2-1:中身も回転させてみよう(6)

PW>Classes>PrototypeViewController.mを開いて、下記を追加。保存しよう。

```
/*
// Override to allow orientations other than the default portrait orientation.
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
// Return YES for supported orientations
    return YES;//(6)
    //return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/
```

Drive15/34

2-2-2:起動してポン

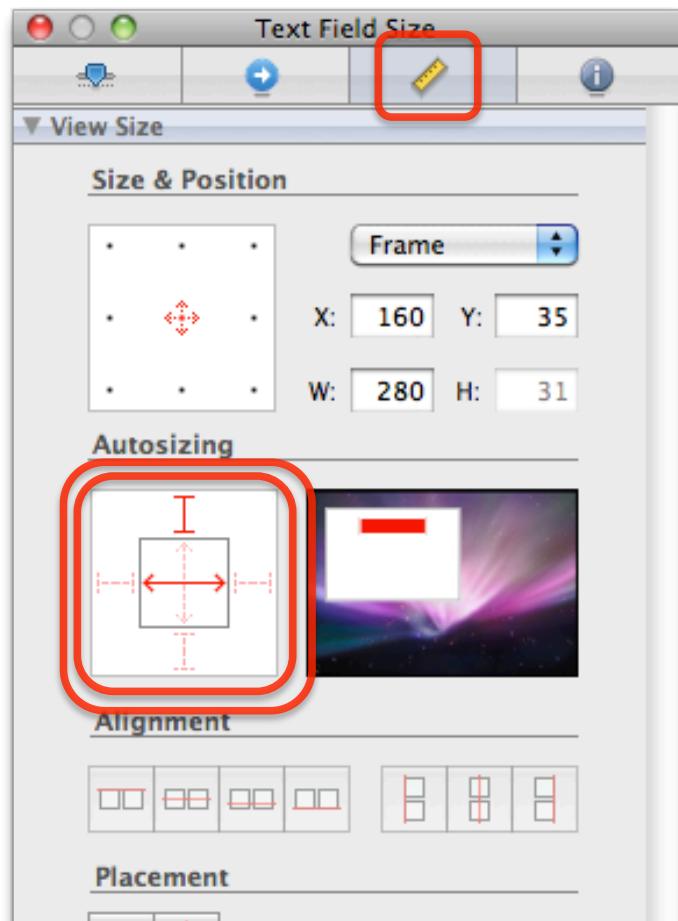
はい、感動しましたかね。

次行きます。横にしたんだけど、TextFieldの幅が、なんか端まで届いてなくて残念ですね。



Drive16/34

2-2-3:回転したら画面にフィットするように



TextFieldエ、..、もっとこう、、びよーんとさ、..
回転したら、iPhoneらしく奇麗に、画面に合わせて
ぴったり変形してほしいじゃん。

PW>PrototypeViewController.xibを開いて、
TextFieldを選択、Inspectorから、
定規マークを選択。

Autosizingの赤い→をクリックして、
四角の中、横矢印だけにしよう。(7)

はあ、、もうネタが尽きてきたよ...

君のTextFieldは きっと伸びる

Drive17/34

2-3-0:ところでTextFieldに文字入力してみつか

大分前から、TextFieldに文字を入力できる。ヒヤッハー!!

でも、何してもキーボードが引っ込まないっす。引っ込ませるためのプログラムが必要(8,9)。

PW>Classes>PrototypeViewController.mを開いて、下記を追加。保存しよう。

```
@implementation PrototypeViewController  
- (IBAction) textEnd {//(8)  
    NSLog(@"入力終了!");  
}
```

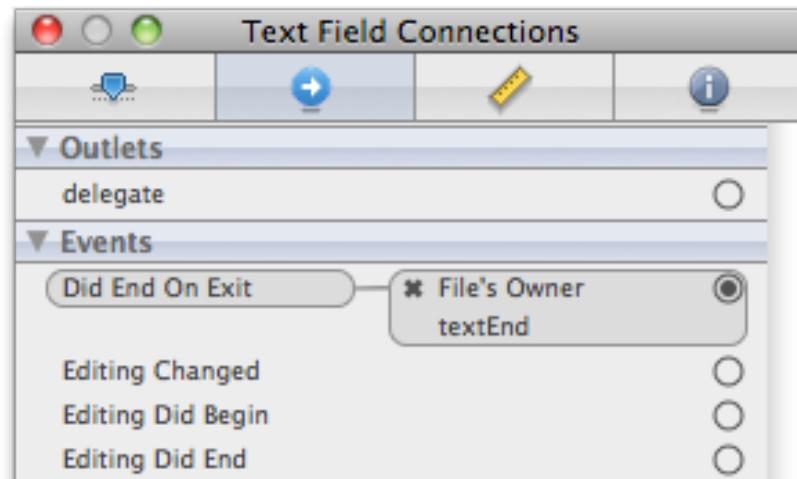
PW>Classes>PrototypeViewController.hを開いて、下記を追加。保存しよう。

```
@interface PrototypeViewController : UIViewController {  
}  
- (IBAction) textEnd;//(9)
```

Drive18/34

2-3-1:IBで先刻のtextEndメソッドを調整！(10)

ここは実演！



保存して起動してみよう。
TextFieldに文字を打って、Enterを押すとキーボードが消えます。

これで、ひとまずおしまい。

Drive19/34

→折り返し地点！

今までやった事を手動でガンバルノデス。

3. IBを使わず地獄のアプリ作り

Drive20/34

3-0-0: 人力で画面作り、の前に

→今までのプログラムを消す！ 、、、のはもったいないので、ただ見えなくします。

PW>Classes>PrototypeAppDelegate.mを開いて、下記を追加。保存しよう。

```
- (BOOL)application:(UIApplication *)application 中略ons {
    NSLog(@"むにやむにや");//(0)
    prototypeViewController = [[PrototypeViewController alloc] init];//(2)
    // [window addSubview:prototypeViewController.view];//(3)
```



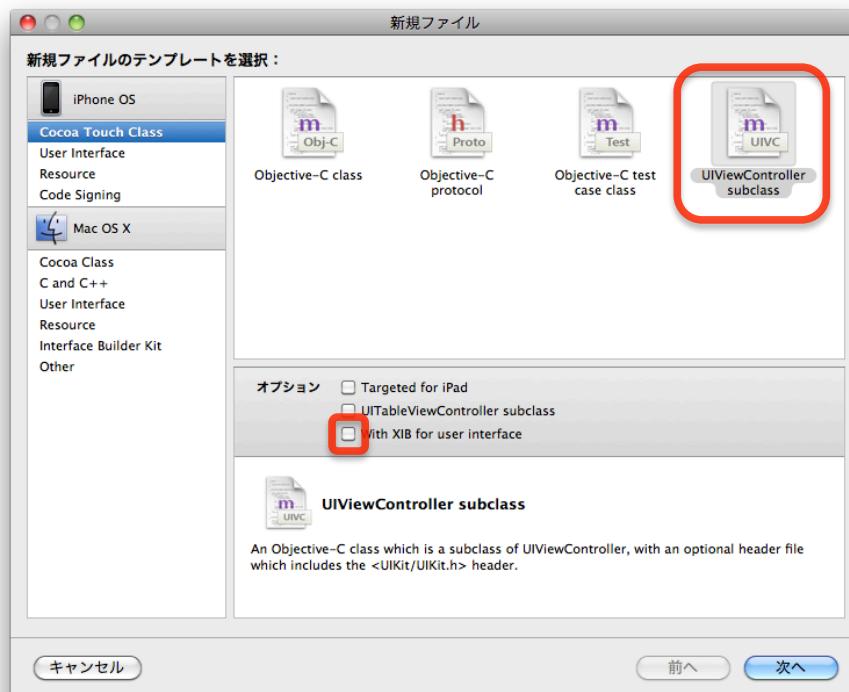
なんもかんも消えた筈だぜ。
真っ白にな、、、。

Drive21/34

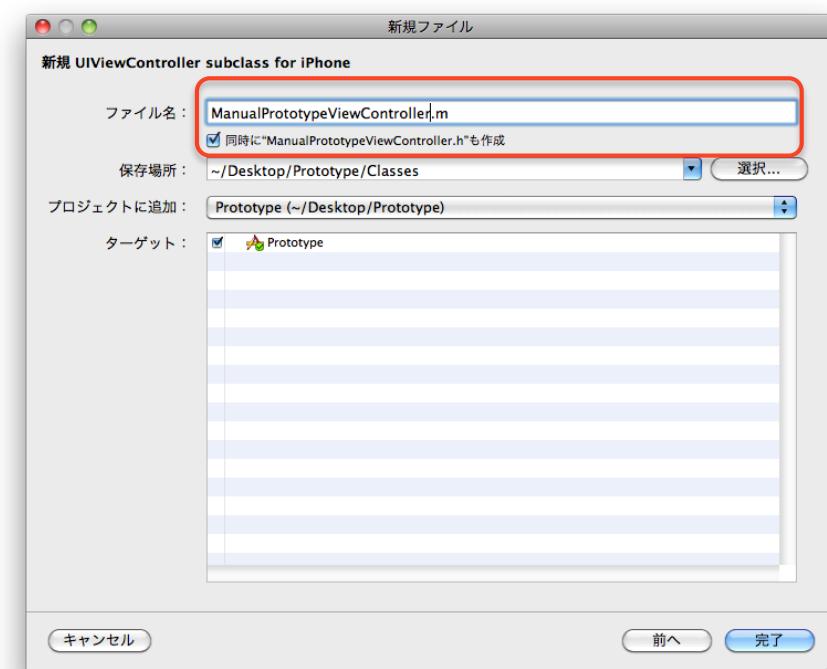
3-0-1: 人力で画面作り

→ ViewControllerをもう一つ作ってみよう。

PW>Classesを右クリック>追加>新規ファイル>



xibを追加、にチェックを入れない。



名前は
「**ManualPrototypeViewController**」で。

Drive22/34

3-0-2:記述！

→とりあえず書き込む。

PW>Classes>PrototypeAppDelegate.mを開いて、下記を追加。保存しよう。

```
- (BOOL)application:(UIApplication *)application 中略ons {
    NSLog(@"%@", @"むにゅむにゅ");
    prototypeViewController = [[PrototypeViewController alloc] init];
    // [window addSubview:prototypeViewController.view];

    manualPrototypeViewController = [[ManualPrototypeViewController alloc] init];
    UITextField * textField = [[UITextField alloc]
        initWithFrame:CGRectMake(10, 10, window.frame.size.width-20, 30)];
    [manualPrototypeViewController.view addSubview:textField];
    [textField setBorderStyle:UITextBorderStyleRoundedRect];
    [textField setAutoresizingMask:UIViewAutoresizingFlexibleWidth];
    [textField addTarget:manualPrototypeViewController action:@selector(textEnd)
        forControlEvents:UIControlEventEditingDidEndOnExit];
    [window addSubview:manualPrototypeViewController.view];
```

Drive23/34

3-0-3:記述2！

→とりあえず書き込む。

PW>Classes>PrototypeAppDelegate.hを開いて、下記を追加。保存しよう。

```
#import "PrototypeViewController.h//(4)
#import "ManualPrototypeViewController.h//(4m)

@interface PrototypeAppDelegate : NSObject <UIApplicationDelegate> {
    UIWindow * window;
    PrototypeViewController * prototypeViewController;//(5)
    ManualPrototypeViewController * manualPrototypeViewController;//(5m)
```

Drive24/34

3-0-4:記述3!

PW>Classes>ManualPrototypeViewController.mを開いて、下記を追加。保存しよう。

```
/*
// Override to allow orientations other than the default portrait orientation.
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)中略aceOrientation {
// Return YES for supported orientations
    return YES;//(6m)
    //return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
*/
```

Drive25/34

3-0-5:これでラスト 記述4! 5!

入力。

PW>Classes>ManualPrototypeViewController.mを開いて、下記を追加。保存しよう。

```
@implementation ManualPrototypeViewController  
- (void) textEnd {//(8m)  
    NSLog(@"マニュアル入力終了!");  
}
```

PW>Classes>ManualPrototypeViewController.hを開いて、下記を追加。保存しよう。

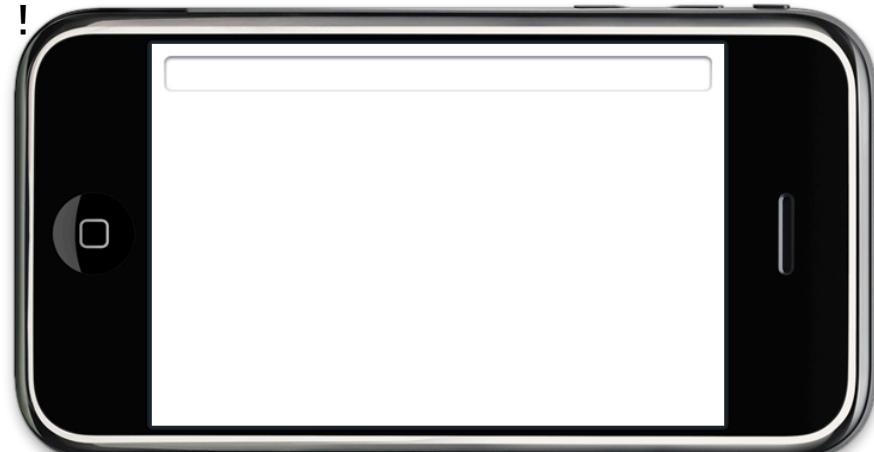
```
@interface ManualPrototypeViewController : UIViewController {  
}  
- (void) textEnd;//(9m)
```

Drive26/34

3-0-6:起動してみよう



回転したら合わせてまわる(6m)、
回転した時の画面に合わせて伸びる(7m)、
文字入力してEnter押したらキーボード引っ込む(8m,9m,10m)、
など！



ぶっちゃけエラー無く一発で通っても、何個エラーが出て
何回やり直しても、あんまり才能とか関係ないです。

だって写してるだけだもん。

Drive27/34

3-1-0:種明かし

“(2)”とか”(3)”とか、今までちょこちょこ書いてたと思います。

“(2m)”は、“(2)”と対応しています。それ以外にも、全て”数字m”と”数字”は、対応した処理です。

PW>Classes>PrototypeAppDelegate.mを例に見てみるとか。

```
manualPrototypeViewController = [[ManualPrototypeViewController alloc] init];//(2m)
UITextField * textField = [[UITextField alloc]
    initWithFrame:CGRectMake(10, 10, window.frame.size.width-20, 30)];
[manualPrototypeViewController .view addSubview:textField];
[textField setBorderStyle:UITextBorderStyleRoundedRect];
[textField setAutoresizingMask:UIViewAutoresizingFlexibleWidth];//(7m)
[textField addTarget:manualPrototypeViewController action:@selector(textEnd)
    forControlEvents:UIControlEventEditingDidEndOnExit];//(10m)
>window addSubview:manualPrototypeViewController.view]; // (3m)
```

Drive28/34

、、ってだけなんですが、大分感覚が違ってると思います。
これで、作業はあと一つ！ 時間はある？

4.上記2と3の比較、ちょっと考え方

Drive29/34

4-0: 考察してみよう

IBでやっている事が、アプリ作成をどう変えているか。

結論から言うと、いいところと、悪いところが有るんですわ。

いいところ

→ プログラム書かずにいろいろ置ける！調整できる！最高やんw

正直、あんなにいっぱい短時間で好きなどこに置くのはプログラムでは無理です。

悪いところ

→ 結局プログラムで書かなきゃいけないところが或る、しかも微妙に違う。

(8),(9)の行程はだいたい共通、、、(IBOutlet)と、(void)、、

→ IBで作っても作らなくても、プログラムで書けてしまう

IBでの設定とプログラムの設定、両方があったら、どうなると思う？

答えは、「作った人でも判らなくなる」(これは後日体験してもらいます)

Drive30/34

4-1:ということは、IBの使いどころって？

基本、諸刃の剣。

だいたいこっちの方向いてる刃の方が鋭い。

でも便利なので、限定して使います。

使うのには注意が必要だよ！ってこと。

指針は次回の授業で、割とあっさり出ます。

という訳で、

Drive31/34

つづく

Drive32/34

つづく

5. プログラムの保存

Drive33/34

5-0:以下が詳しい(ていうかウチ)

<http://kissaki-blog.blogspot.com/2010/05/git.html>

、、、なんだけど、今日はインターネットが無いので、ディスクからインストールしましょう。

Gitというプログラムを使います。

- ・Gitフォルダをコピーしてもらったと思います。
- ・フォルダ内の、git-1-1.7.0-intel-leopard.dmgをダブルクリック。
適当にYesとか入れるといいです。
- ・フォルダ内の、GitXStable-1.app.zipをダブルクリック。

なんか出てきた



アイコンを、

Macintosh HD>アプリケーション フォルダに入れときましょう。

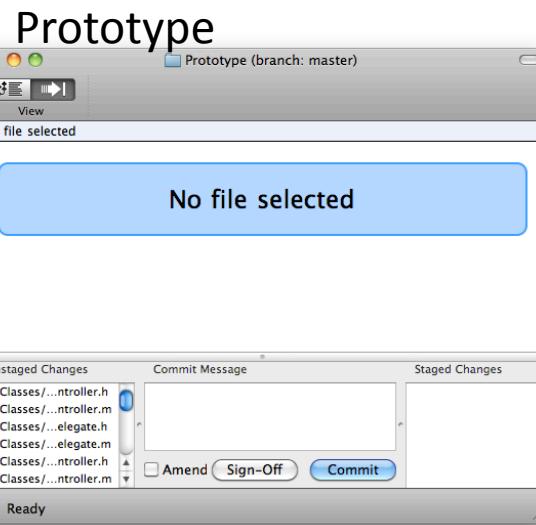
Drive34/34

5-1:GitXを起動、レツツセーブ

File>new



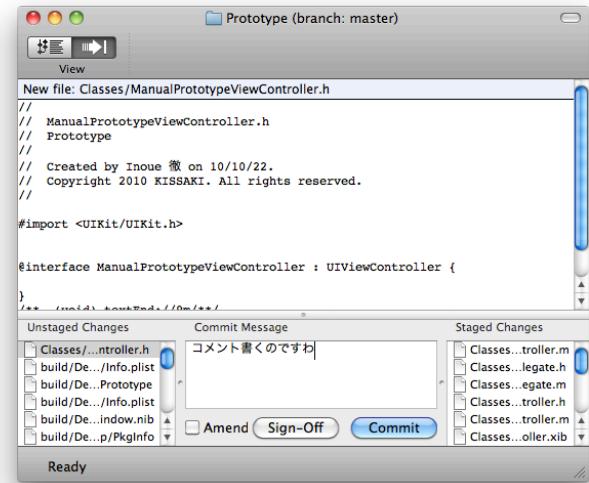
Prototypeという名前の
フォルダを選択、Open。



左下のウインドウのBuild/で始まるヤツ以外を選択、
ダブルクリック。内容が右下のウインドウに移ります。

Build/で始まるヤツは、選択して右クリック、ignore filesを選択。
.gitignoreが新しくできるので、これも選択してダブルクリック。

真ん中のウインドウにコメントを入力、Commitして完了！



6.LookBack

1. Agenda, Drive, Lookbackという授業の形を共有
2. Xcodeでプログラム書いた
3. インターフェースビルダー(IB)使った
4. 同じ内容を人力コードで書いた
5. 保存した
6. コレ。

ハイ、おつかれさまです。

プログラムについての質問、とかいろいろ受け付けます。

一つだけ前提があり、

- ・内容がプログラムに関係ある事
- ・内容がTechHubに関係ある事
- ・必ず「井上さんへ ~~より」などメール本文に宛先、出元を書く事。
あれ、三つある。

techhubteach@googlegroups.com



Continue?

次回予告：

次回は、

- ・プログラム、プログラマに共通する“あるルール”
→IBのつかいどころ

- ・サンプルプログラム改編
→お楽しみ。

この2点について、やります。

次回、

「手書きとルールと」