コデアルiOSアプリ勉強会
- Programming 24 -

nakasen_20th

目指すところ

- プログラマになる
- TwitterクライアントiOSアプリを作る

iOSアプリを作れるようになる

1. iOSアプリの構成を知る
2. 開発環境Xcodeを知る
3. Objective-Cの基礎を知る
4. UIKitを知る

Twitterアプリを作る

5. Twitter APIを知る

6. ググり方を知る（超重要）

7. 並列処理の方法を知る

プログラマになる

8. 作品を作りこむ

9. GitHubに公開する

10.ライトニングトークで自分を売り込む（発表）

1.　iOSアプリの構成を知る

iOS機器

- iPhone
- iPad　iPad mini
- iPod Touch

iOSのバージョン

- iOS7とそれ以前
  http://www.idownloadblog.com/2013/06/14/ios-6-vs-ios-7-the-different-ui-views/
  （見た目の大きな変化）

- 今回はiOS7に限定

  - iOS7動作機種

    - iPhone4以降

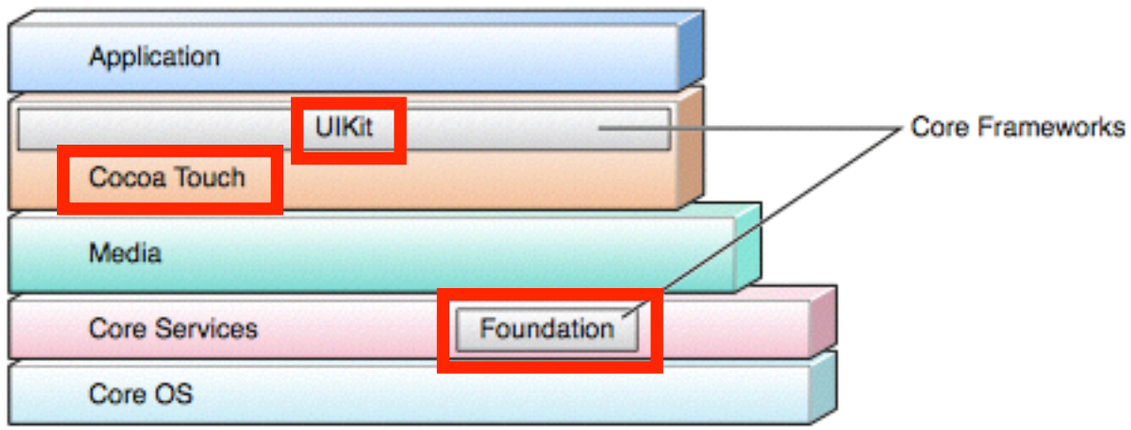
    - iPad 2以降、iPad mini全て

    - iPod Touch 第５世代以降

フレームワーク

- https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaFundamentals.pdf

Figure 1-2 Cocoa in the architecture of iOS



2.　開発環境Xcodeを知る

- Apple社純正統合開発環境
- バージョンは最新（5.1）に（20140401現在）
- 新規作成時は「Single View Application」で
- 画面構成に早く慣れる

Xcode画面構成

http://www.atmarkit.co.jp/ait/articles/1212/05/news022.html

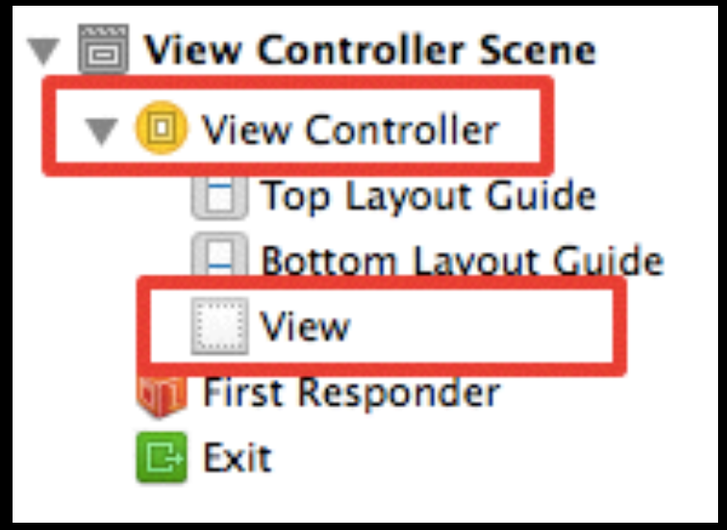
- ツールバー
- ナビゲーターエリア
- エディタエリア
- ユーティリティエリア
- デバッグエリア

ナビゲーターエリア

- 重要なのは３つ
  - Main.storyboard
  - ViewController.h
  - ViewController.m

Storyboard

- エディタエリアの初期状態を確認
- Document Outlineを表示
  - ★ View Controllerがひとつ
  - ★ その上にViewがひとつ



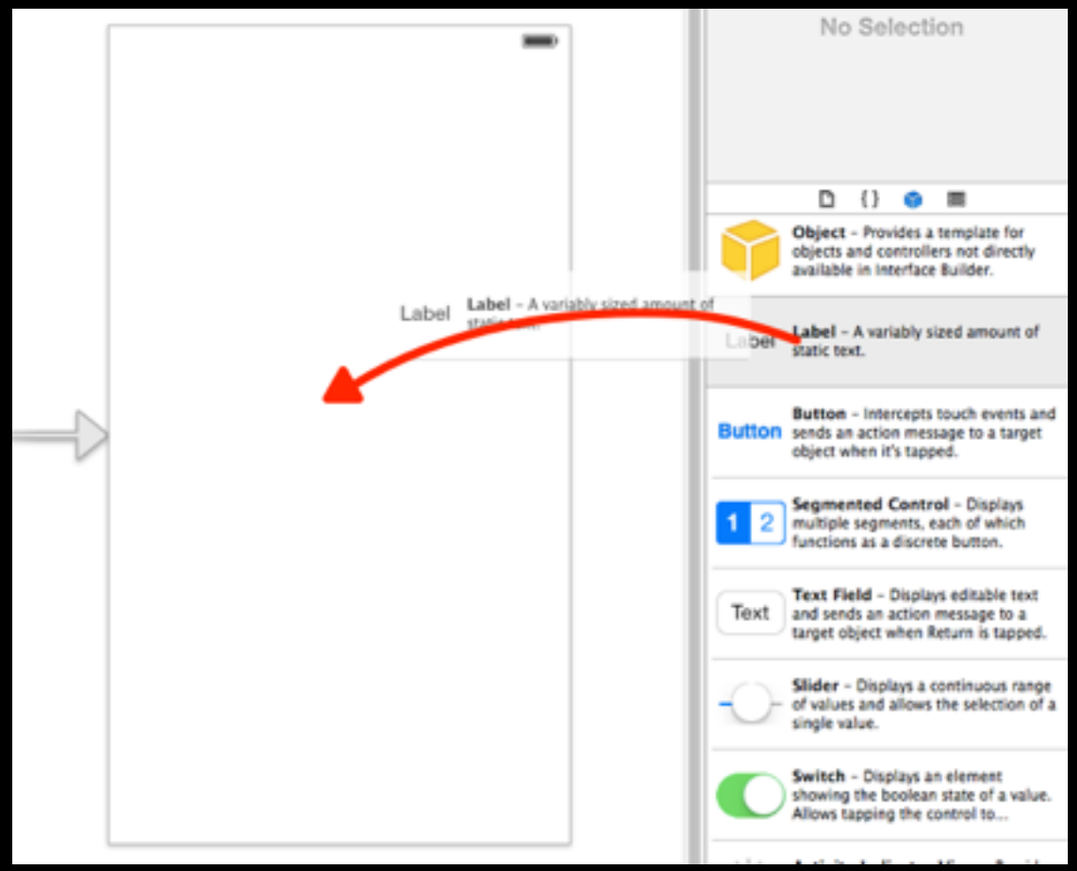
Object library

- UIパーツを確認
- ドラッグでView上に配置

★ Label

★ Button

★ View Controller



セグエで連結

- Buttononから次のView Controllerへ
  Control + ドラッグ

- セグエの種類は「Push」

- 最初のView ControllerにNavigation
  ControllerをEmbed In

- もう一度Document Outlineを確認

Navigation Controller

- View Controllerの遷移を記録して管理
- 画面上部のNavigation Barに戻るボタンを
  自動生成
- Embed InしたView Controllerから記録開始

Attributes inspector

- Viewの属性を変えてみる
- Labelのテキストを変えてみる
- Buttonの種類を変えてみる
- Segueの種類を変えてみる

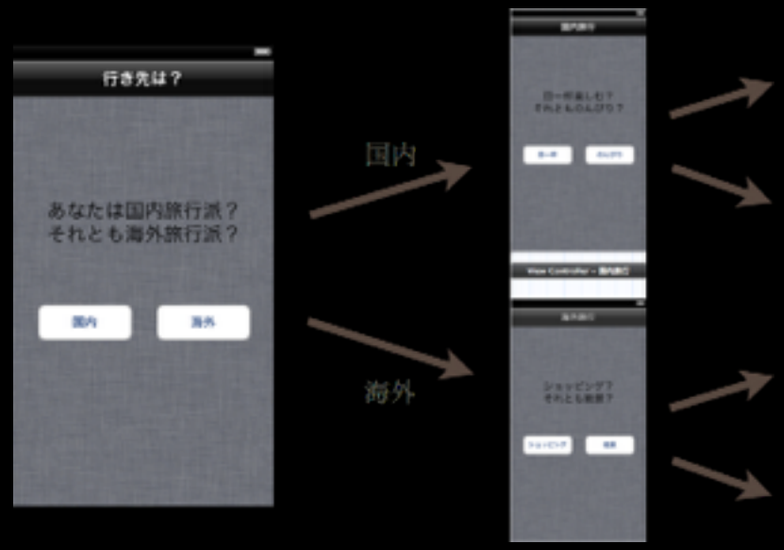
ビルド、実行

- ここで実行してみる（iOSシミュレータの起動）
- ハードウェアを選択
- 画面の向きを変えてみる

演習課題（Ex01）

- アンケートアプリ作成（コードは書かない！）
- 二者択一の画面遷移をセグエで実現

- 最終結果は４画面
- 題材は自由



3.　Objective-Cの基礎を知る

- C言語とオブジェクト指向要素のハイブリッド
- オブジェクトとそうでないもの（プリミティブ）が混在
- プリミティブと構文と関数でとりあえず動く

変数（プリミティブ）

- 変数は値の入れ物
- 型で分類
  - int型（整数）
  - float型（実数）
  - BOOL型（1か0、YESかNO）
    他の言語と混乱した人向け　→
    http://d.hatena.ne.jp/thata/20091123/1258950667
  - その他（文字型はオブジェクトの時に）

構文（if文）

- if文は条件分岐

```
int age = 20;  //  この数値を変えてみる
if (age >= 20) {
    NSLog(@“成人"); // NSLog関数は後述
} else {
    NSLog(@"未成年");
}
```

構文（for文）

- for文は繰り返し

```
for (int i = 1; i <= 10; i++) { // 繰返し条件
    NSLog(@"%d", i); // %dは整数を十進表示する書式
}
```

- if、forともに条件式に注意

  条件式で「等しい」は「==」（比較演算子）
  http://www.objectivec-iphone.com/introduction/operator/logical-operator.html

- 構文は入れ子が可能（ifの中にif、など）

構文（switch文）

- switch文は条件分岐（3分岐以上可）

```
int era = 1; // この数値を変えてみる
switch (era) {
    case 0:
        NSLog(@"昭和");
        break;
    case 1:
        NSLog(@"平成");
        break;
    default:
        NSLog(@"年号 error!”); // 大正生まれはどうする？
        break;
}
```

演習問題

- 掛け算九九
- NSLog関数で"1 × 1 = 1"、"1 × 2 = 2"・・・
- for文の入れ子で短く書く

関数

- 引数を与えて戻り値を得る
- プログラミング言語では引数や戻り値がない場合がある
  →単なる処理の固まり
- f(x)=2x は

```
int twice(int value) {
    return 2 * value;
}
```

- この関数を呼び出す時は

```
int x = 5;
int answer = twice(x);
```

関数いろいろ

- 戻り値がない場合
  - ★ 型はvoid
  - ★ returnがない
- 引数がない場合
- 引数が２つ以上の場合

```
void display2()
{
    NSLog(@"2 times.");
}

int always2()
{
    return 2;
}

int twice(int value) {
    return 2 * value;
}

int multiple(int value1, int value2)
{
    return value1 * value2;
}
```

トピックス：名前

- 名前決めるの大変！
  - 変数名
  - 関数名
  - 仮引数名
- これだけ書籍の何十ページ分
- 困ったら一緒に悩みましょう

関数の呼び出し

- それぞれの場合

```
display2(); // "2 times."

int answer = always2(); // 2 -> answer

int answerTwice = twice(100); // 200 -> answerTwice

int answerMulti = multiple(5, 3); // 15 -> answerMulti
```

演習問題

- 消費税込計算関数　iTax()
- 引数100を与えれば戻り値108が得られる

- 税率も引数に含めたい場合は、
  引数２つの関数を作成

オブジェクト指向

- クラス
  変数定義とメソッド定義の設計書

- メソッド
  関数定義のオブジェクト指向版

- インスタンスの生成
  クラス（設計書）を元に実体を生み出す
  実体は変数フィールドとメソッドを持つ

クラスの書き方

- @interface
  - ★ 変数フィールド名、メソッド名の宣言部
- @implementation
  - ★ メソッド実装部

- @interfaceは「.h」に書かれることが多い
  @implementationは「.m」に書かれることが多い

- ViewController.hとViewController.mを見てみよう

メソッドの書き方

- 戻り値がない場合とある場合
- 引数がない場合と１つの場合と２つ以上の場合
- ラベル付き引数はやっかい

```
// メソッド定義
- (float)average:(int)score1 eScore:(int)score2 jScore:(int)score3
{
    return (float)(score1 + score2 + score3) / 3.0f;
}
```

- メソッド呼び出しの書式が不思議

```
Score *score = [[Score alloc] init]; // インスタンス生成
[score setIdNumber:107]; // セッタ呼び出し
float average3 = [score average:70 eScore:50 jScore:80]; // メソッド実行
NSLog(@"ID番号%dの学生の３科目の平均点は%.2f点です。", [score getIdNumber], average3); // ゲッタ呼び出し
```

インスタンスの生成方法

- allocしてinitする
- newする（上に同じ）
- allocしてからinitの代わりにinitWith〜を使う
  （指定イニシャライザ）
- allocとinitWith〜をまとめて行うメソッド
  （コンビニエンスコンストラクタ）を使う
- その他（この後出てくるクラスはほとんどこれ）

クラス宣言、実装、実行

- Scoreクラス（クラス名は英大文字から）

```
@interface Score : NSObject
{
    int idNumber;    // 今回は未使用
    int math;        // 今回は未使用
    int english;     // 今回は未使用
    int japanese;    // 今回は未使用
}

- (float)average:(int)score1 eScore:(int)score2 jScore:(int)score3;

@end

@implementation Score

- (float)average:(int)score1 eScore:(int)score2 jScore:(int)score3
{
    return (float)(score1 + score2 + score3) / 3;
}

@end
```

- 実行（インスタンスを生成してメソッド実行）

```
Score *score = [[Score alloc] init];
float average3 = [score average:80 eScore:90 jScore:70];
NSLog(@"３科目の平均点は%.2f点です。", average3);
```

変数へのアクセス

- クラスで宣言した変数はクラス外から見えない
- 値のセット、値の読み出しができるように

それぞれメソッドを用意する

★ セッタ

★ ゲッタ

```
- (void)setIdNumber:(int)number
{
    idNumber = number;
}

- (int)getIdNumber
{
    return idNumber;
}
```

変数へのアクセス（2）

- 宣言部で@property指定をすると
  セッタ、ゲッタが自動生成される

```
@interface Score : NSObject
{
    // 以下の４項目は今回は未使用。セッタ、ゲッタを使ってアクセスできる。
    // int idNumber;
    int math;
    int english;
    int japanese;
}

@property (nonatomic) int idNumber;
```

- ドットシンタックスでアクセスできるので便利

```
Score *score = [[Score alloc] init]; // インスタンス生成

// [score setIdNumber:107]; // セッタ呼び出し
score.idNumber = 107; // ドットシンタックスでセット

float average3 = [score average:70 eScore:50 jScore:80]; // メソッド実行

// NSLog(@"ID番号%dの学生の３科目の平均点は%.2f点です。", [score getIdNumber], average3); // ゲッタ呼び出し
NSLog(@"ID番号%dの学生の３科目の平均点は%.2f点です。", score.idNumber, average3); // ドットシンタックスでゲット
```

演習問題（Ex02）

- 電卓の消費税ボタンCalciTaxクラスを作成
  - クラス税率taxフィールドを持つ
  - taxに値をセットするsetTaxメソッド持つ
  - taxの値を読み出すgetTaxメソッドを持つ
  - 税込み計算を行うiTaxメソッドを持つ
- インスタンスを生成し、計算を実行する

Foundationフレームワーク

- 基礎的なクラスをたくさん定義してある
  - NSNumberクラス
  - NSStringクラス
  - NSArrayとNSMutableArrayクラス
  - NSDictionaryとNSMutableDictionaryクラス
  - その他たくさん

NSNumberクラス

- 数値オブジェクト（インスタンス化して用いる）
- オブジェクトだがalloc initしなくても
  初期化できる

- alloc initしても良い

```
NSNumber *myNumber1 = @1;

NSNumber *myNumber2 = [[NSNumber alloc] initWithInt:1]; // 指定イニシャライザ
NSNumber *myNumber3 = [NSNumber numberWithInt:1]; // コンビニエンスコンストラクタ

NSLog(@"%@, %@, %@", myNumber1, myNumber2, myNumber3);
```

NSStringクラス

- 文字列オブジェクト
- オブジェクトだがalloc initしなくても初期化できる

- alloc initすると怒られる

```
NSString *myName1 = @"Funassy";

NSString *myName2 = [[NSString alloc] initWithString:@"Funassy"];   ⚠ Using 'initWithString:' with a literal is redundant
NSString *myName3 = [NSString stringWithString:@"Funassy"];         ⚠ Using 'stringWithString:' with a literal is redundant

NSLog(@"%@, %@, %@", myName1, myName2, myName3);
```

NSArrayクラス（NSMutableArrayクラス）

- 配列オブジェクト（複数の要素を持つ）
- 要素はオブジェクトならなんでも良い
- オブジェクトだがalloc initしなくても初期化できる
- alloc initしても良い

```
NSArray *rgbArray1 = @[@"Red", @"Green", @"Blue"];

NSArray *rgbArray2 = [[NSArray alloc] initWithObjects:@"Red", @"Green", @"Blue", nil];
NSArray *rgbArray3 = [NSArray arrayWithObjects:@"Red", @"Green", @"Blue", nil];

NSLog(@"%@, %@, %@", rgbArray1[0], rgbArray2[1], rgbArray3[2]);
```

- NSMutableArrayオブジェクトは要素の変更、追加、削除が可（NSArrayはそれらが不可、ということ）

NSDictionaryクラス（NSMutableDictionaryクラス）

- 辞書オブジェクト（キーと値のセット）
- 要素はオブジェクトならなんでも良い
- オブジェクトだがalloc initしなくても初期化できる
- alloc initしても良い

```
NSDictionary *personDict1 = @{@"name":@"Funassy", @"age":@39};

NSDictionary *personDict2 = [[NSDictionary alloc] initWithObjectsAndKeys:
                             @"Funassy", @"name", [NSNumber numberWithInt:39], @"age", nil];
NSDictionary *personDict3 = [NSDictionary dictionaryWithObjectsAndKeys:
                             @"Funassy", @"name", [NSNumber numberWithInt:39], @"age", nil];

NSLog(@"%@, %@, %@", personDict1[@"name"], personDict2[@"age"], personDict3[@"age"]);
```

- NSMutableDictionaryオブジェクトは要素の変更、追加、削除可（NSDictionaryはそれらが不可、ということ）

コレクションクラス

- 複数の要素を持てるオブジェクトをコレクションと言う
  - NSArray
  - NSMutableArray
  - NSDictionary
  - NSMutableDictionary
  - その他
- for文との相性が良い　@ITの記事がわかりやすい
  http://www.atmarkit.co.jp/ait/articles/0901/21/news126_2.html
- 特定要素の取り出し、要素の操作、追加削除変更など、メソッドを把握しておく

演習問題（Ex03）

- 成績表を一覧表示するReportクラスを作成
  - 学籍番号、氏名、成績フィールド
  - 平均点を求めるメソッド
  - セッタメソッド、ゲッタメソッド
- 成績フィールドはNSDictionaryで持つ
  - 数学、英語、国語キーの値を持つ

4. UIKitを知る

- UI（ユーザインターフェース）の役割
  - 情報の伝達
  - ユーザの誘導
  - 総合的なユーザ体験

- UIで迷ったらAppleのドキュメントに立ち返る
  「iOSヒューマンインターフェースガイドライン」
  https://developer.apple.com/jp/devcenter/ios/library/documentation/userexperience/conceptual/mobilehig/BasicsPart/BasicsPart.html#//apple_ref/doc/uid/TP40006556-CH2-SW1

StoryboardでUI構築

- UI構築作業をStoryboardに任せる
- もう一度Object libraryでUIパーツを確認
- もう一度Attributes inspectorの項目を確認

ViewControllerグループ

- UIViewControllerクラス
- UINavigationControllerクラス
- UITableViewControllerクラス
- その他

Viewグループ

- UILabelクラス
- UIViewクラス
- UIImageViewクラス
- UITextViewクラス
- その他

Controlグループ

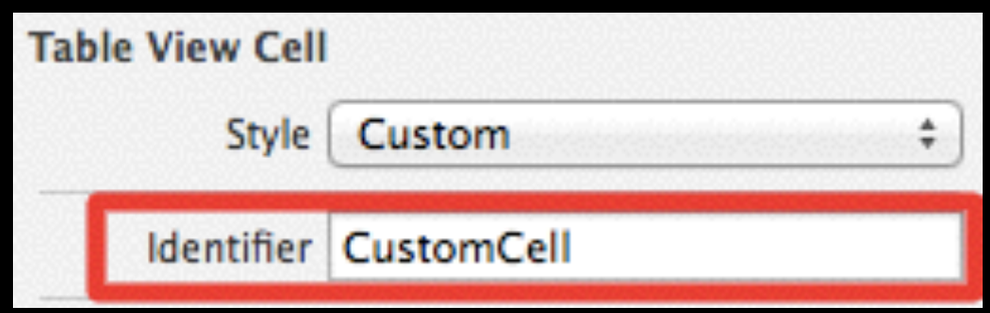
- UIButtonクラス
- UISegmentedControlクラス
- UITextViewクラス
- その他

その他グループ

- UITableViewCellクラス
- UIGestureRecognizerクラス
- その他

UIパーツをStoryboardで生成

- xibファイルを作成してセルをカスタマイズ
  - ★ UITableViewCellクラスのサブクラスを作成
  - ★ xibファイルを同時に作成
  - ★ xibファイルをGUIでカスタマイズ
  - ★ Identifierを合わせる



UIパーツをコードのみで作成

- ラベルやボタンをStoryboardを使わず作成

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    self.view.backgroundColor = [UIColor lightGrayColor]; // Viewの背景色を変更

    UILabel *myLabel1 = [[UILabel alloc] initWithFrame:CGRectMake(60, 50, 200, 100)];
    // 指定イニシャライザで初期化
    myLabel1.backgroundColor = [UIColor blueColor]; // Labelの背景色を変更
    myLabel1.textColor = [UIColor yellowColor]; // Labelの文字色を変更
    myLabel1.textAlignment = NSTextAlignmentCenter; // 中央揃え
    myLabel1.text = @"ラベル1"; // Labelタイトル
    [self.view addSubview:myLabel1]; // LabelをViewに貼り付け

    UIButton *myButton1 = [UIButton buttonWithType:UIButtonTypeCustom];
    // コンビニエンスコンストラクタで初期化
    myButton1.frame = CGRectMake(60, 200, 200, 100);
    myButton1.backgroundColor = [UIColor redColor]; // Buttonの背景色を変更
    [myButton1 setTitle:@"ボタン1" forState:UIControlStateNormal]; // 通常時のButtonタイトル
    [myButton1 addTarget:self
                  action:@selector(buttonAction:)
        forControlEvents:UIControlEventTouchUpInside]; // ボタンを押した時の処理（メソッドの実行）
    [self.view addSubview:myButton1]; // ButtonをViewに貼り付け
}
```

UITextFieldをコードのみで作成

- 文字入力を受け付けるサンプルソース

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

    self.view.backgroundColor = [UIColor lightGrayColor];
    UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(60, 50, 200, 30)];
    textField.borderStyle = UITextBorderStyleRoundedRect;
    textField.placeholder = @"年齢を入力して下さい。";
    textField.keyboardType = UIKeyboardTypeNumbersAndPunctuation;
    textField.returnKeyType = UIReturnKeyDone;
    [textField addTarget:self
                  action:@selector(displayMessage:)
        forControlEvents:UIControlEventEditingDidEndOnExit];
    [self.view addSubview:textField];

    _messageLabel = [[UILabel alloc] initWithFrame:CGRectMake(20, 100, 280, 50)];
    _messageLabel.textColor = [UIColor redColor];
    _messageLabel.textAlignment = NSTextAlignmentCenter;
    [self.view addSubview:_messageLabel];
}

- (void)displayMessage:(UITextField *)inputTextField {
    int inputNumber = [inputTextField.text intValue];
    int answerAge = inputNumber + 5;
    _messageLabel.text = [NSString stringWithFormat:@"あなたの本当の年齢は%d才ですね。", answerAge];
}
```

UIKit重点項目

- UIViewControllerクラス
  - ライフサイクル
  - Navigation Controllerによる管理
    遷移してきた画面（ViewController）を配列で記憶
    pushで追加、popで取り出し（戻る）
- UITableViewControllerクラス
  - 構成要素
  - 代表的なメソッド
  - セルの生成方法

UIViewController

- 各メソッドの呼び出されるタイミング
  http://d.hatena.ne.jp/glass-_-onion/20120405/1333611664
  など、良記事がたくさん

  - viewDidLoad
  - viewWillAppear:
  - viewWillDisappear:
  - prepareForSegue::
  - didReceiveMemoryWarning

UITableViewController

- 各メソッドの機能
  http://blogios.stack3.net/archives/604
  （UITableViewの解説）など、良記事あり

  - セクション数を返すメソッド
  - 各セクションのセル数を返すメソッド
  - 各セル表示の際に呼び出されるメソッド
  - 各セルの高さを決定するメソッド
  - 各セルのクリック時に呼び出されるメソッド
    など

UITableViewController

・各メソッドのソースコード例

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1; // セクション数
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return 10; // セクション内の行数
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath // セルの定義
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"Cell" forIndexPath:indexPath]; // 再利用

    // Configure the cell...
    cell.textLabel.text = [NSString stringWithFormat:@"%d", indexPath.row]; // 行番号を表示
    return cell;
}

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath // セルの高さ
{
    return indexPath.row % 2 * 30 + 30; // 1行おきにセルの高さを変更
}

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath // セルクリック時
{
    NSLog(@"row = %d", indexPath.row); // 行番号をログ出力
}
```

5. Twitter APIを知る

- Twitterでできること
  - タイムラインを表示する
  - つぶやく
  - リツイート、リプライ、お気に入り、その他

今回の作成目標アプリ

- アカウントの選択
- ホームタイムラインの表示
- ツイート画面からツイート
- リツイートボタン、お気に入りボタン
- その他（オリジナルツイート画面等）

APIの利用

- APIの役割
  - 外部にプログラムの機能を開放し、アプリケーション開発の負担を緩和する
- Twitter APIの役割
  - サードパーティのTwitterアプリ作成の手間を減らす

Twitter APIの利用方法

- Twitter Developers “REST API v1.1 Resources”
  https://dev.twitter.com/docs/api/1.1

- リソースURLを指定してHTTP(S)でアクセス

- リソースURLとレスポンス例が掲載

- 用語に慣れが必要
  （「つぶやく」= statuses update など）

Twitter API

- 情報の要求はGETメソッド
- 情報の送信はPOSTメソッド
- レスポンスはJSONデータ
  - ★ JSONデータをよく見て慣れておく
  - ★ JSONデータはiOS側でNSDictionaryに変換
    「キー：値」のペア、入れ子構造など共通

JSONデータの扱い

- iOSアプリ側でNSDictionaryに変換
- NSJSONSerializationクラスのメソッドを利用

```
NSArray *timeLineData =
[NSJSONSerialization JSONObjectWithData:responseData // JSONデータ(NSData)
                                options:NSJSONReadingAllowFragments
                                  error:&jsonError];
```

- 並列処理で（後述）
- エラーを考慮

アプリからHTTPアクセス

- SLRequestクラスのメソッドを利用
- リソースURLを指定してアクセス

```
NSURL *url = [NSURL URLWithString:@"https://api.twitter.com"
              @"/1.1/statuses/home_timeline.json"];
NSDictionary *params = @{@"count" : @"100",
                         @"trim_user" : @"0",
                         @"include_entities" : @"0"};
SLRequest *request = [SLRequest requestForServiceType:SLServiceTypeTwitter
                                        requestMethod:SLRequestMethodGET
                                                  URL:url
                                           parameters:params];
[request performRequestWithHandler:^(NSData *responseData,
                                     NSHTTPURLResponse *urlResponse,
                                     NSError *error) {
    if (responseData) {
        self.httpErrorMessage = nil;
        if (urlResponse.statusCode >= 200 && urlResponse.statusCode < 300) {
```

- 並列処理で
- エラーを考慮

並列処理

- Webへのアクセスは重い処理
  ユーザの操作を妨げてはならない
- 重い処理は裏で並列処理
- Objective-CのGCDを利用（非同期処理）

GCD

- 処理単位をキューで管理
- メインキューとグローバルキュー
- GUI処理は必ずメインキューで
- 重い処理をグローバルキューへ

dispatch_async()関数

- 一般的な書式

```
dispatch_async(グローバルキューの取得, ^{
    重い処理「A」
    dispatch_async(メインキューの取得, ^{
        重い処理完了後の処理「B」
    )};
});
重い処理の完了を待たずに実行する処理「C」
```

- 処理「A」と処理「C」は並行して実行される。
- 処理「A」が終了した時に処理「C」が重い処理をしている場合は、
  メインキューの取得待ちになる。

dispatch_async()サンプル

- サンプルソース

```
- (IBAction)timeConsumingTask:(id)sender {
    dispatch_queue_t mainqueue = dispatch_get_main_queue();
    dispatch_queue_t globalQueueDefault =
    dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

    dispatch_async(globalQueueDefault, ^{
        for (int i = 0; i < 10000; i++) {
            for (int j = 0; j < 100000; j++) {

            }
        }
        dispatch_async(mainqueue, ^{
            self.messageLabel.text = @"Finish!";
        });
    });
    self.messageLabel.text = @"Wait...";
}
```

Twitter APIにおける並列処理

- SLRequestやNSJSONSerializationクラスのメソッドではブロックを利用
  （ ^{ 処理 } ）
- ブロック内の処理は並列で実行
  終了後GUI処理をする場合は
  メインキューを取得する必要あり

Twitter APIのレスポンス

- HTTPのステータスコードを調べる
  - 200番台が成功
  - 400番台はクライアントエラー
  - 500番台はサーバエラー
- エラーの種類によってアプリのメッセージを変えるべき

ツイート画面

- SLComposeViewControllerを利用

```
if ([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter]) { //利用可能チェック
    NSString *serviceType = SLServiceTypeTwitter;
    SLComposeViewController *composeCtl = [SLComposeViewController
        composeViewControllerForServiceType:serviceType];
    [composeCtl setCompletionHandler:^(SLComposeViewControllerResult result) {
        if (result == SLComposeViewControllerResultDone) {
            //投稿成功時の処理

        }
    }];
    [self presentViewController:composeCtl animated:YES completion:NULL];
}
```

- オリジナルツイートシートにするなら
  UITextViewを含むViewControllerを用意する

認証処理

- Twitterアカウント処理が必要
- 複数のアカウントがある場合は
  アクションシートで選択
- 認証が成功したらidentifierを持ち回る

ACAccount

- 複数のアカウントから選択

```
self.accountStore = [[ACAccountStore alloc] init];
ACAccountType *twitterType =
[self.accountStore accountTypeWithAccountTypeIdentifier:ACAccountTypeIdentifierTwitter];
[self.accountStore requestAccessToAccountsWithType:twitterType
                                           options:NULL
                                        completion:^(BOOL granted, NSError *error) {
    if (granted) {
        self.twitterAccounts = [self.accountStore accountsWithAccountType:twitterType];
        if (self.twitterAccounts.count > 0) {
            ACAccount *account = self.twitterAccounts[0];
            self.identifier = account.identifier;
            dispatch_async(dispatch_get_main_queue(), ^{
                self.accountDisplayLabel.text = account.username;
            });
        } else {
            dispatch_async(dispatch_get_main_queue(), ^{
                self.accountDisplayLabel.text = @"アカウントなし";
            });
        }
    } else {
        NSLog(@"Account Error: %@", [error localizedDescription]);
        dispatch_async(dispatch_get_main_queue(), ^{
            self.accountDisplayLabel.text = @"アカウント認証エラー";
        });
    }
}];
```

アクションシートから選択

- UIActionSheetから一つを選択するコード

```
- (IBAction)setAccountAction:(id)sender
{
    UIActionSheet *sheet = [[UIActionSheet alloc] init];
    sheet.delegate = self;

    sheet.title = @"選択してください。";
    for (ACAccount *account in self.twitterAccounts) {
        [sheet addButtonWithTitle:account.username];
    }
    [sheet addButtonWithTitle:@"キャンセル"];
    sheet.cancelButtonIndex = self.twitterAccounts.count;
    sheet.actionSheetStyle = UIActionSheetStyleBlackTranslucent;
    [sheet showInView:self.view];
}

- (void)actionSheet:(UIActionSheet *)actionSheet clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (self.twitterAccounts.count > 0) {
        if (buttonIndex != self.twitterAccounts.count) {
            ACAccount *account = self.twitterAccounts[buttonIndex];
            self.identifier = account.identifier;
            self.accountDisplayLabel.text = account.username;
            NSLog(@"Account set! %@", account.username);
        }
        else {
            NSLog(@"cancel!");
        }
    }
}
```

タイムライン表示

- タイムライン用ViewControllerに
  アカウント情報を引き継ぐ

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue.identifier isEqualToString:@"TimeLineSegue"]) {
        TimeLineTableViewController *timeLineTableViewController = segue.destinationViewController;
        if ([timeLineTableViewController isKindOfClass:[TimeLineTableViewController class]]) {
            timeLineTableViewController.identifier = self.identifier;
        }
    }
}
```

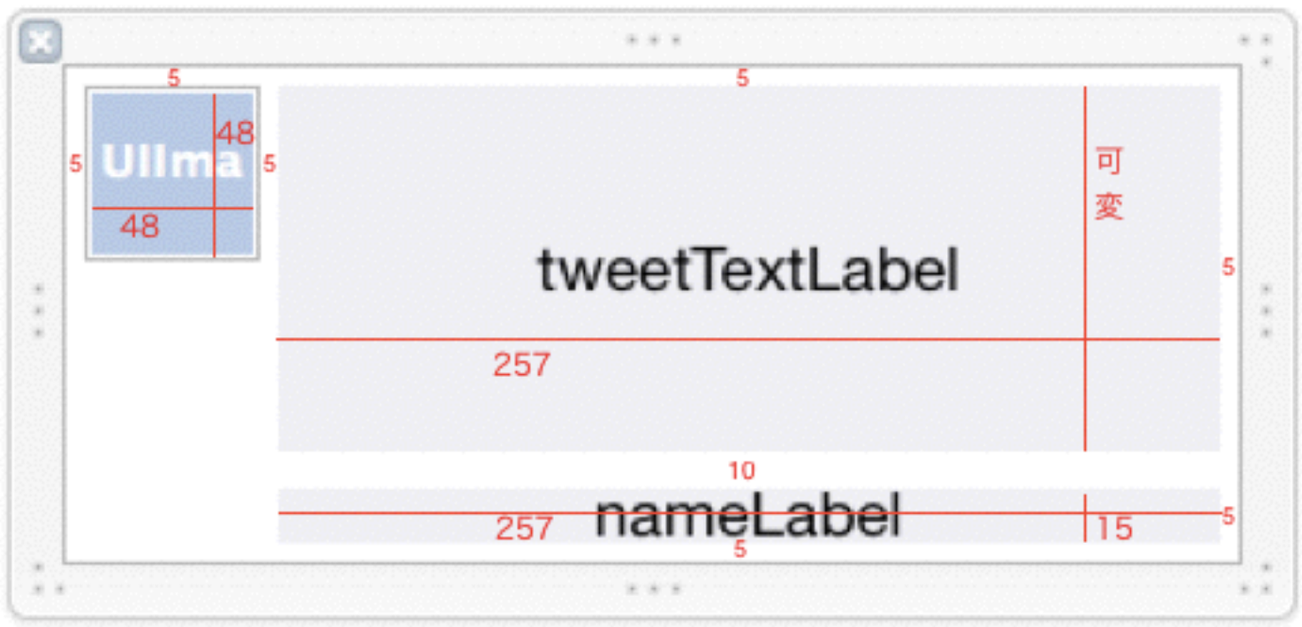
カスタムセル

- UITableViewCellのサブクラスを作成し、initWithStyle::でUIパーツを配置してカスタマイズ

```
- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)reuseIdentifier
{
    self = [super initWithStyle:style reuseIdentifier:reuseIdentifier];
    if (self) {
        // Initialization code
        _paddingTop = 5;
        _paddingBottom = 5;

        _tweetTextLabel = [[UILabel alloc] initWithFrame:CGRectZero];
        _tweetTextLabel.font = [UIFont systemFontOfSize:14.0f];
        _tweetTextLabel.textColor = [UIColor blackColor];
        _tweetTextLabel.numberOfLines = 0;
        [self.contentView addSubview:_tweetTextLabel];

        _nameLabel = [[UILabel alloc] initWithFrame:CGRectZero];
        _nameLabel.font = [UIFont systemFontOfSize:12.0f];
        _nameLabel.textColor = [UIColor blackColor];
        [self.contentView addSubview:_nameLabel];

        _profileImageView = [[UIImageView alloc] initWithFrame:CGRectZero];
        _profileImageView.image = _image;
        [self.contentView addSubview:_profileImageView];
    }
    return self;
}
```

カスタムセル続き

- カスタムセルの設計



- layoutSubViews

```
- (void)layoutSubviews
{
    [super layoutSubviews];

    self.profileImageView.frame = CGRectMake(5, self.paddingTop, 48, 48);
    self.tweetTextLabel.frame = CGRectMake(58, self.paddingTop, 257, self.tweetTextLabelHeight);
    self.nameLabel.frame = CGRectMake(58, self.paddingTop + self.tweetTextLabelHeight + 10, 257, 15);
}
```

TimeLineTableViewController

- viewDidLoadでアカウント引き継ぎ
  及び初期設定

```
[super viewDidLoad];

[self.tableView registerClass:[TimeLineCell class] forCellReuseIdentifier:@"TimeLineCell"];

self.mainQueue = dispatch_get_main_queue();
self.imageQueue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_BACKGROUND, 0);

ACAccountStore *accountStore = [[ACAccountStore alloc] init];
ACAccount *account = [accountStore accountWithIdentifier:self.identifier];

NSURL *url = [NSURL URLWithString:@"https://api.twitter.com"
              @"/1.1/statuses/home_timeline.json"];
NSDictionary *params = @{@"count" : @"100",
                         @"trim_user" : @"0",
                         @"include_entities" : @"0"};
SLRequest *request = [SLRequest requestForServiceType:SLServiceTypeTwitter
                                        requestMethod:SLRequestMethodGET
                                                  URL:url
                                           parameters:params];
[request setAccount:account];
```

TimeLineTableViewController

- さらにviewDidLoadでタイムラインを取得

```
UIApplication *application = [UIApplication sharedApplication];
application.networkActivityIndicatorVisible = YES;

[request performRequestWithHandler:^(NSData *responseData,
                                     NSHTTPURLResponse *urlResponse,
                                     NSError *error) {
    if (responseData) {
        self.httpErrorMessage = nil;
        if (urlResponse.statusCode >= 200 && urlResponse.statusCode < 300) {
            NSError *jsonError;
            self.timeLineData =
            [NSJSONSerialization JSONObjectWithData:responseData
                                            options:NSJSONReadingAllowFragments
                                              error:&jsonError];
            if (self.timeLineData) {
                NSLog(@"Timeline Response: %@\n", self.timeLineData);
                dispatch_async(self.mainQueue, ^{
                    [self.tableView reloadData];
                });
            } else {
                NSLog(@"JSON Error: %@", [jsonError localizedDescription]);
            }
        } else {
            self.httpErrorMessage =
            [NSString stringWithFormat:@"The response status code is %d",
             urlResponse.statusCode];
            NSLog(@"HTTP Error: %@", self.httpErrorMessage);
        }
    }
    dispatch_async(self.mainQueue, ^{
        UIApplication *application = [UIApplication sharedApplication];
        application.networkActivityIndicatorVisible = NO;
    });
}];
```

TimeLineTableViewController

・ツイート本文ラベルの高さを返すメソッドを用意

```
- (CGFloat)labelHeight:(NSString *)labelText
{
    // ラベルの行間設定
    UILabel *aLabel = [[UILabel alloc] init];
    CGFloat lineHeight = 18.0;
    NSMutableParagraphStyle *paragrahStyle = [[NSMutableParagraphStyle alloc] init];
    paragrahStyle.minimumLineHeight = lineHeight;
    paragrahStyle.maximumLineHeight = lineHeight;

    // テキスト属性を付与
    NSString *text = (labelText == nil) ? @"" : labelText;
    UIFont *font = [UIFont fontWithName:@"HiraKakuProN-W3" size:14];
    NSDictionary *attributes = @{NSParagraphStyleAttributeName: paragrahStyle,
                                 NSFontAttributeName: font};
    NSAttributedString *aText = [[NSAttributedString alloc] initWithString:text attributes:attributes];
    aLabel.attributedText = aText;

    // ラベルの高さ計算
    CGFloat aHeight =
    [aLabel.attributedText boundingRectWithSize:CGSizeMake(257, MAXFLOAT)
                                        options:NSStringDrawingUsesLineFragmentOrigin
                                        context:nil].size.height;
    return aHeight;
}
```

TimeLineTableViewController

- セクション数は1、
  セル数はレスポンス取得前後で変化

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    // Return the number of sections.
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    // Return the number of rows in the section.
    if (!self.timeLineData) {
        return 1;
    } else {
        return self.timeLineData.count;
    }
}
```

セルの定義

- cellForRowAtIndexPath その1

```
- (UITableViewCell *)tableView:(UITableView *)tableView
         cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    TimeLineCell *cell = [tableView dequeueReusableCellWithIdentifier:@"TimeLineCell"
                                                         forIndexPath:indexPath];

    // Configure the cell...

    if (self.httpErrorMessage) {
        cell.tweetTextLabel.text = self.httpErrorMessage;
        cell.tweetTextLabelHeight = 24;
    } else if (!self.timeLineData) {
        cell.tweetTextLabel.text = @"Loading...";
        cell.tweetTextLabelHeight = 24;
    } else {
        NSString *name = self.timeLineData[indexPath.row][@"user"][@"screen_name"];
        NSString *text = self.timeLineData[indexPath.row][@"text"];
```

セルの定義

- cellForRowAtIndexPath その2

```
// カスタムセルを使わない場合は以下の4行
// cell.textLabel.font = [UIFont systemFontOfSize:14];
// cell.textLabel.numberOfLines = 0;
// cell.detailTextLabel.font = [UIFont systemFontOfSize:12];
// cell.detailTextLabel.text = name;

cell.tweetTextLabelHeight = [self labelHeight:text];
cell.tweetTextLabel.text = text;
cell.nameLabel.text = name;

cell.profileImageView.image = [UIImage imageNamed:@"blank.png"];
```

セルの定義

- cellForRowAtIndexPath その3

```
    dispatch_async(self.imageQueue, ^{
        NSString *url;
        NSDictionary *tweetDictionary = self.timeLineData[indexPath.row];

        if ([[tweetDictionary allKeys] containsObject:@"retweeted_status"]) {
            // リツイートの場合はretweeted_statusキー項目が存在する
            url = tweetDictionary[@"retweeted_status"][@"user"][@"profile_image_url"];
        } else {
            url = tweetDictionary[@"user"][@"profile_image_url"];
        }

        NSData *data = [NSData dataWithContentsOfURL:[NSURL URLWithString:url]];
        dispatch_async(self.mainQueue, ^{
            UIApplication *application = [UIApplication sharedApplication];
            application.networkActivityIndicatorVisible = NO;
            UIImage *image = [[UIImage alloc] initWithData:data];
            cell.profileImageView.image = image;
            [cell setNeedsLayout];
        });
    });
}
return cell;
```

セルの高さの定義

- hightForRowAtIndexPath

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    NSString *tweetText = self.timeLineData[indexPath.row][@"text"];
    CGFloat tweetTextLabelHeight = [self labelHeight:tweetText];
    return tweetTextLabelHeight + 35;
}
```

セル選択で詳細表示へ

- didSelectRowAtIndexPath

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    // Navigation logic may go here. Create and push another view controller.
    TimeLineCell *cell = (TimeLineCell *)[tableView cellForRowAtIndexPath:indexPath];

    DetailViewController *detailViewController = [self.storyboard
        instantiateViewControllerWithIdentifier:@"DetailViewController"];
    detailViewController.name = cell.nameLabel.text;
    detailViewController.text = cell.tweetTextLabel.text;
    detailViewController.image = cell.profileImageView.image;
    detailViewController.identifier = self.identifier;
    detailViewController.idStr = [[self.timeLineData objectAtIndex:indexPath.row]
        objectForKey:@"id_str"];

    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
}
```

リツイート、お気に入りなど

- 詳細画面（DetailViewController）に
  リツイートボタン設置（前半）

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.

    self.imageView.image = self.image;
    self.nameView.text = self.name;
    self.textView.text = self.text;
}
```

```
- (IBAction)retweetAction:(id)sender {
    ACAccountStore *accountStore = [[ACAccountStore alloc] init];
    ACAccount *account = [accountStore accountWithIdentifier:self.identifier];

    NSString *urlString = [NSString stringWithFormat:
                           @"https://api.twitter.com/1.1/statuses/retweet/%@.json", self.idStr];
    NSURL *url = [NSURL URLWithString:urlString];
    // NSDictionary *params = @{@"trim_user" : @"1"};

    SLRequest *request = [SLRequest requestForServiceType:SLServiceTypeTwitter
                                            requestMethod:SLRequestMethodPOST
                                                      URL:url
                                               parameters:nil];

    [request setAccount:account];

    UIApplication *application = [UIApplication sharedApplication];
    application.networkActivityIndicatorVisible = YES;
```

リツイート

- 詳細画面その2

```
[request performRequestWithHandler:^(NSData *responseData,
                                     NSHTTPURLResponse *urlResponse,
                                     NSError *error) {
    if (responseData) {
        NSInteger statusCode = urlResponse.statusCode;
        if (statusCode >= 200 && statusCode < 300) {
            NSDictionary *postResponseData =
            [NSJSONSerialization JSONObjectWithData:responseData
                                            options:NSJSONReadingMutableContainers
                                              error:NULL];
            NSLog(@"[SUCCESS!] Created Tweet with ID: %@", postResponseData[@"id_str"]);
        }
        else {
            NSLog(@"[ERROR] Server responded: status code %ld %@", statusCode,
                  [NSHTTPURLResponse localizedStringForStatusCode:statusCode]);
        }
    }
    else {
        NSLog(@"[ERROR] An error occurred while posting: %@", [error localizedDescription]);
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        UIApplication *application = [UIApplication sharedApplication];
        application.networkActivityIndicatorVisible = NO;
    });
}];
}
```

本文中のURLからWebViewへ

- UITextViewのテキスト中のURLをクリック可能に
  - ★ リンク先を表示するViewControllerを用意（UIWebViewを持つWebViewControllerを新規作成）
  - ★ UIApplicationクラスのサブクラスを作成（MyUIApplicationを新規作成）
  - ★ openURLメソッドをオーバーライド

AppDelegateの編集

- AppDelegateクラスでNavigationControllerを参照可能にする
  - ★ AppDelegate.h

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

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;
@property (nonatomic, strong) UINavigationController *navigationController;

@end
```

  - ★ AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{
    // Override point for customization after application launch.

    self.navigationController = (UINavigationController *)self.window.rootViewController;
    return YES;
}
```

WebViewController

- WebViewController.h

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

@interface WebViewController : UIViewController

@property (strong, nonatomic) IBOutlet UIWebView *webView;
@property (strong, nonatomic) IBOutlet UIActivityIndicatorView *activityIndicator;
@property (nonatomic, strong) NSURL *openURL;

@end
```

- WebViewController.m

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.

    NSURLRequest *myRequest = [NSURLRequest requestWithURL:self.openURL];
    [self.webView loadRequest:myRequest];
}

- (void)webViewDidStartLoad:(UIWebView *)webView {
    [self.activityIndicator startAnimating];
}

- (void)webViewDidFinishLoad:(UIWebView*)webView {
    [self.activityIndicator stopAnimating];
}

- (void)webView:(UIWebView*)webView
didFailLoadWithError:(NSError*)error {
    [self.activityIndicator stopAnimating];
}
```

MyUIApplication

- MyUIApplication.h

```
#import <UIKit/UIKit.h>
#import "AppDelegate.h"
#import "WebViewController.h"

@interface MyUIApplication : UIApplication

@property (nonatomic, strong) NSURL *myOpenURL;

@end
```

- MyUIApplication.m

```
- (BOOL)openURL:(NSURL *)url
{
    if (!url) {
        return NO;
    }

    self.myOpenURL = url;
    AppDelegate *appDelegate = (AppDelegate *)[self delegate];
    UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard"
                                                         bundle:[NSBundle mainBundle]];
    WebViewController *webViewController =
    [storyboard instantiateViewControllerWithIdentifier:@"WebViewController"];
    webViewController.openURL = self.myOpenURL;
    webViewController.title = @"Web View";

    [appDelegate.navigationController pushViewController:webViewController animated:YES];
    self.myOpenURL = nil;

    return YES;
}
```

main.mの編集

- 今回はmain.mも編集
  - UIApplicationMain関数の第3引数をカスタムクラスのクラス名にする

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

#import "AppDelegate.h"
#import "MyUIApplication.h"

int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, NSStringFromClass([MyUIApplication class]),
            NSStringFromClass([AppDelegate class]));
    }
}
```

参考：カスタムツイートシート

- オリジナルのツイートシートのサンプル

  - ★ TweetSheetViewController.h

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

@interface TweetSheetViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextView *tweetTextView;
@property (nonatomic, strong) ACAccountStore *accountStore;

- (IBAction)editEndAction:(id)sender;
- (IBAction)tweetAction:(id)sender;

@end
```

カスタムツイートシート

- ★ TweetSheetViewController.m その 1

```
- (IBAction)editEndAction:(id)sender {
    [self.tweetTextView resignFirstResponder];
}

- (IBAction)cancelAction:(id)sender {
    [self dismissViewControllerAnimated:YES completion:NULL];
}

- (IBAction)tweetAction:(id)sender {
    ACAccountStore *accountStore = [[ACAccountStore alloc] init];
    ACAccount *account = [accountStore accountWithIdentifier:self.identifier];
    NSString *tweetString = self.tweetTextView.text;

    NSURL *url = [NSURL URLWithString:@"https://api.twitter.com"
                  @"/1.1/statuses/update.json"];
    NSDictionary *params = @{@"status" : tweetString};
    SLRequest *request = [SLRequest requestForServiceType:SLServiceTypeTwitter
                                            requestMethod:SLRequestMethodPOST
                                                      URL:url
                                               parameters:params];
    //UIImage *image = [UIImage imageNamed:@"testIcon.png"];
    //NSData *imageData = UIImageJPEGRepresentation(image, 1.f);
    //NSData *imageData = UIImagePNGRepresentation(image);
    //[request addMultipartData:imageData
    //                 withName:@"media[]"
    //                     type:@"image/png"
    //                 filename:@"testIcon.png"];
    request.account = account;
```

カスタムツイートシート

- ★ TweetSheetViewController.m その２

```
    UIApplication *application = [UIApplication sharedApplication];
    application.networkActivityIndicatorVisible = YES;

    [request performRequestWithHandler:^(NSData *responseData,
                                         NSHTTPURLResponse *urlResponse,
                                         NSError *error) {
        if (responseData) {
            self.httpErrorMessage = nil;
            if (urlResponse.statusCode >= 200 && urlResponse.statusCode < 300) {
                NSDictionary *postResponseData =
                [NSJSONSerialization JSONObjectWithData:responseData
                                                options:NSJSONReadingMutableContainers
                                                  error:NULL];
                NSLog(@"[SUCCESS!] Created Tweet with ID: %@", postResponseData[@"id_str"]);
            }
            else {
                self.httpErrorMessage =
                [NSString stringWithFormat:@"The response status code is %ld",
                 urlResponse.statusCode];
                NSLog(@"HTTP Error: %@", self.httpErrorMessage);
            }
        }
        dispatch_async(dispatch_get_main_queue(), ^{
            UIApplication *application = [UIApplication sharedApplication];
            application.networkActivityIndicatorVisible = NO;
            [self dismissViewControllerAnimated:YES completion:^{
                NSLog(@"Tweet Sheet has been dismissed.");
            }];
        });

    }];
}
```

6.　　ググり方を知る

- ググり方で差がつく！
- キーワードの選び方が重要
  - ～とは
- 検索時の条件指定
  - 対象としないキーワード
  - 期間指定

頼りになるサイト

- Stack Overflow
- Qiita
- クラスメソッドのiOS7特集
- Apple公式ドキュメント
- iOS Dev Center（WWDC資料）

ググり方演習

- エラーメッセージで検索
  - どれがエラーメッセージか
  - 英語をいかに読むか
    - 問いは自分と同じか
    - どれが答えか
    - どれくらいあてになるか

7. 作品を作りこむ

- 完成度を上げるには
  - ★ 画面レイアウト
  - ★ 画像パーツ調整
  - ★ ユーザインターフェース調整
  - ★ エラー対応

画面レイアウトの調整

- 画面の向きの対応
  - Portrait
  - Landscape（Left、Right）
  - Upside Down

画像パーツの調整

- キーカラー、背景色とのコントラスト
- 画像ファイルの解像度、ファイル形式、圧縮
- 機種ごとの画像の用意（iPad、Retina対応）

ユーザインターフェースの調整

- Apple公式ドキュメント
  「ヒューマンインターフェースガイドライン」

- iOS7の目指すところ
  - UIは脇役
  - ユニバーサルデザイン
  - アクセシビリティの考え方

エラー対応

- どのエラーまで対応するか
- ユーザのエラー操作を未然に防ぐ
- 通信のエラー対応はしつこいくらいに
- ユーザ向けのメッセージ
- 開発者向けのメッセージ
- ユーザからのフィードバックを受け取る
  （永遠のベータ版？）

テストの取り組み

- まずテストを書く
- カバレッジより「できるところからやる」
- 振り返りで頭を冷やして軌道修正

8. GitHub

- バージョン管理
- Gitコマンド基礎
- GitHubにアカウント作成
- GitHubでのバージョン管理
- GitHubでの分散バージョン管理

バージョン管理

- ソースプログラムのライフサイクル
- 終わりなき更新
- 過去に戻れるか
- 本体 + 差分 + 管理情報

Gitコマンド基礎

- まずはローカルから
  - じっくり取り組むならここ（宣伝）
    http://engineer-intern.jp/archives/11957
- ３つの段階を理解
  - ワーキングツリー
  - インデックス
  - リポジトリ

ローカルでGitコマンド

- git config で初期設定
- git init でリポジトリ作成
  - 取り消しは rm -r .git
- git status で状態確認
- ファイル作成（修正）でワーキングツリーへ
  - 取り消しは
    rm ファイル名（作成時）または
    git chekout --ファイル名（修正時）

ローカルでGitコマンド

- git add でワーキングツリーからインデックスへ
  - 取り消しは
    git rm -- cached ファイル名（作成時）または
    git reset HEAD ファイル名（修正時）
- git commit でインデックスからリポジトリへ
  - 取り消しは
    git reset -- soft HEAD^（修正時のみ）

状態を見るGitコマンド

- git log でオブジェクトIDと日付を確認
- git diff で差分を確認
- gitk で経過を確認

GitHubにアカウント作成

- ソーシャルコーディング
- 無料のアカウントはソースが常にさらされる
  （かえって勉強になる）
- 他の人のソースもじっくり見る
- git clone でリポジトリごとコピー

GitHubでのバージョン管理

- リモートリポジトリ作成（New repositoryボタン）
  - じっくり取り組むならここ（宣伝）
    http://engineer-intern.jp/archives/12305
- 公開されることを覚悟して名前をつける
- 後々のことも考えて設定する
- git remote でリモートリポジトリの設定
- git push でリモートリポジトリにプッシュ

GitHubで分散バージョン管理

- もう一人になり切って分散開発の疑似体験
- 2人目が git clone でリポジトリコピー
- git add、git commit 後 git push
- 1人目が git pull で2人目の修正取り込み
- わざとコンフリクト発生
- git pull、修正、git push でコンフリクト解消

9. 発表準備

- 発表のために必要なこと
- ライトニングトークについて
- 振り返りの必要性

発表のために必要なこと

- リハーサル
  - なかなか重要性を理解できない
- 機器チェック
  - いざという時の逃げ道確保
- 時間の管理
  - 発表を邪魔せずに

ライトニングトークについて

- 時間制限が厳しい
- 多くの発表を聴く人への優しさ
  - テンポの良さ
  - ウケるプレゼン増加傾向
- プレッシャーで発表者が成長
  - 簡潔な箇条書きへシフト
  - 「読ませる」より「見せる」
  - スライド１枚当たりの情報と説明時間を減らす

振り返りの必要性

- 発表直後にこそ最大の成長可能性
- 冷静に見てくれている人の存在
- 自分が客観的になれるかどうか