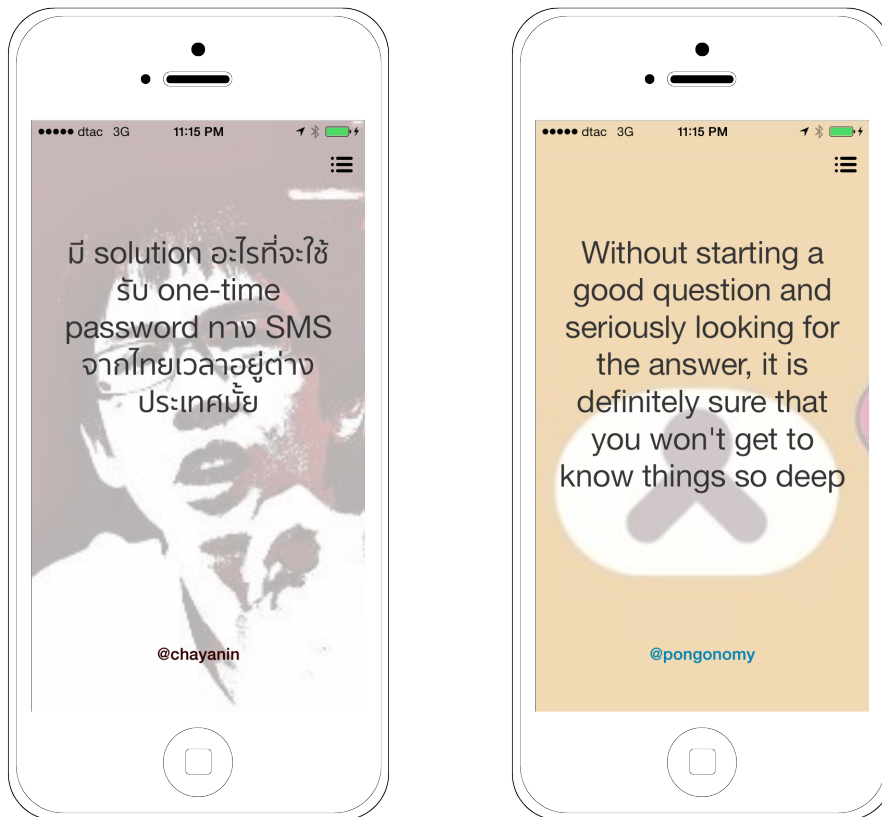


Tweety, one tweet at a time.

เราจะทำ Tweety กัน. Tweety เป็น app สำหรับแสดงข้อความจาก Twitter โดยแสดงทีละ 1 ข้อความ Swipe ไปมาเพื่อแสดงข้อความถัดไป



จุดประสงค์

ทดลองใช้ Component อื่นๆใน UIKit ทำความรู้จักกับ Storyboard และระบบ Auto Layout. การใส่ Animation แบบง่ายๆ รวมถึงการใช้ Thread ด้วย Grand Central Dispatcher

Download Asset ต่างๆที่ใช้ใน Project นี้ได้ที่นี้:

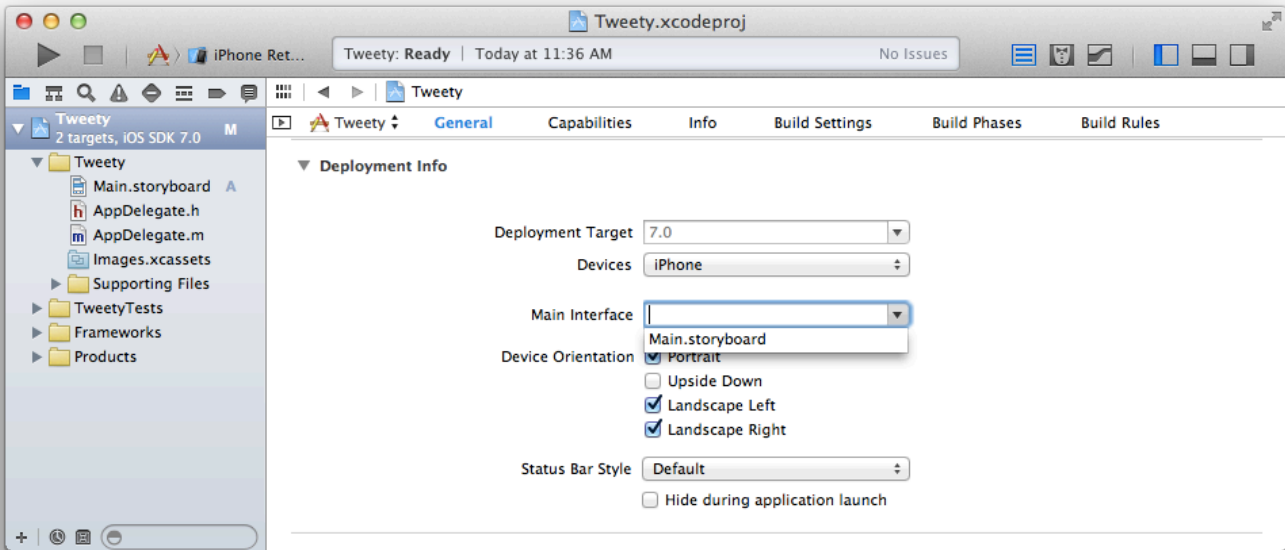
<https://github.com/programmerbird/ios-tweety/raw/master/output/assets.zip>

สร้าง Storyboard

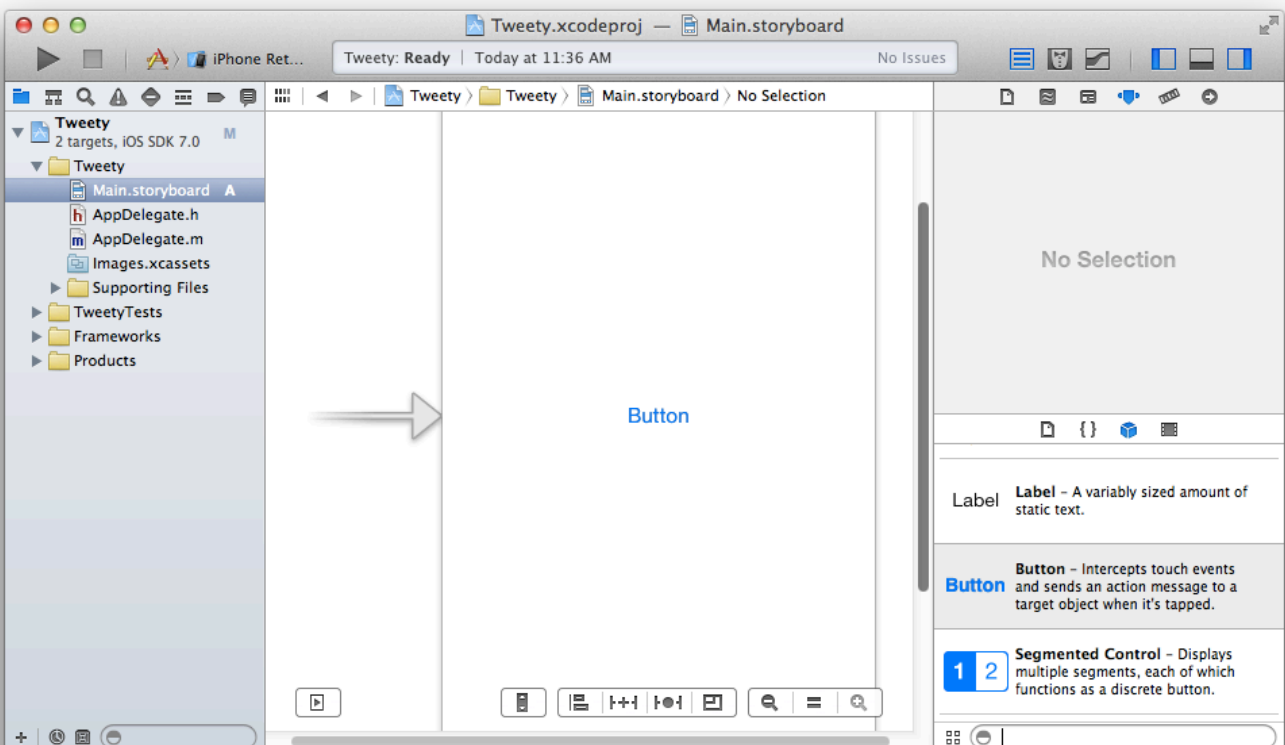
เราจะสร้าง Project ใหม่โดยใช้ Storyboard กัน

1. เปิด Xcode แล้วสร้าง Project ใหม่ชื่อ Tweety เป็นแบบ Empty Application.
2. ลบของใน -application:didFinishLaunchingWithOptions: ทิ้งให้เหลือแต่ `return YES;`

3. เพิ่มไฟล์ ประเภท Storyboard ในหมวด User Interface ชื่อว่า “Main” เข้าไปใน Project
4. เลือก Project จากเมนูด้านซ้าย ที่ Development Info ในช่อง Main Interface เลือก Main.storyboard



5. ใน Main.storyboard ลาก View Controller จากเมนู Object Library ในด้านขวาเข้ามาใส่
6. เราจะเรียกหน้านี้ว่าหน้า Root ให้ใส่ UIButton เข้าไป 1 ปุ่ม



Checkpoint

ใน Storyboard มีลูกศรชี้เข้ามาที่ View Controller และ เมื่อ Run ใน Simulator ควรจะเห็นปุ่มที่ได้ใส่เข้าไปเมื่อสักรู

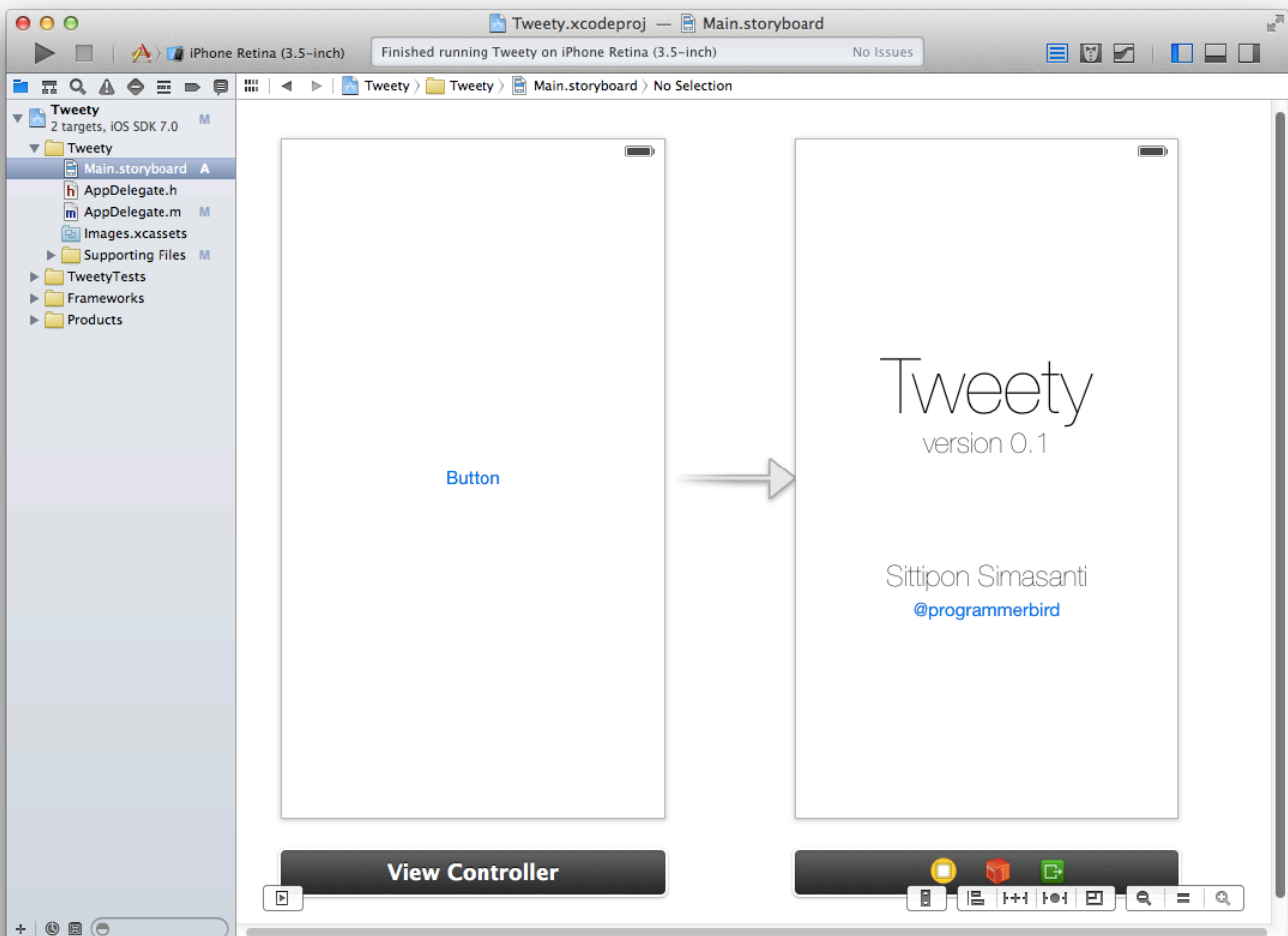
รู้จักกับ Storyboard

Storyboard เป็นระบบ UI ใหม่ ที่นำมาใช้ตั้งแต่ iOS 5 เป็นต้นมา โดยรวมหน้า UI หลายๆ หน้า ของ App เอาไว้ด้วยกัน ประกอบกับโยงความสัมพันธ์ระหว่างหน้า ทำให้เราเห็นภาพรวมการทำงานของ App เหมาะกับการพัฒนา App ขนาดเล็ก และมี Flow การทำงานไม่มากนัก

หน้า About

เราจะลองทำหน้า About เมื่อกดปุ่มที่สร้างขึ้นเมื่อสักครู่นี้ให้เปลี่ยนไปที่หน้า

1. ใน Xcode, เลือก Main.storyboard
2. ลาก View Controller อันใหม่ใส่เข้าไปใน Storyboard ที่ด้านขวามือ ของ View Controller แรก
3. เราจะเรียก View Controller นี้ว่าหน้า About ตกแต่งหน้า About ให้สวยงาม
4. ลากลูกศรชี้เข้าจาก View Controller หน้า Root มาใส่ไว้ที่ View Controller หน้า About อันใหม่



Checkpoint

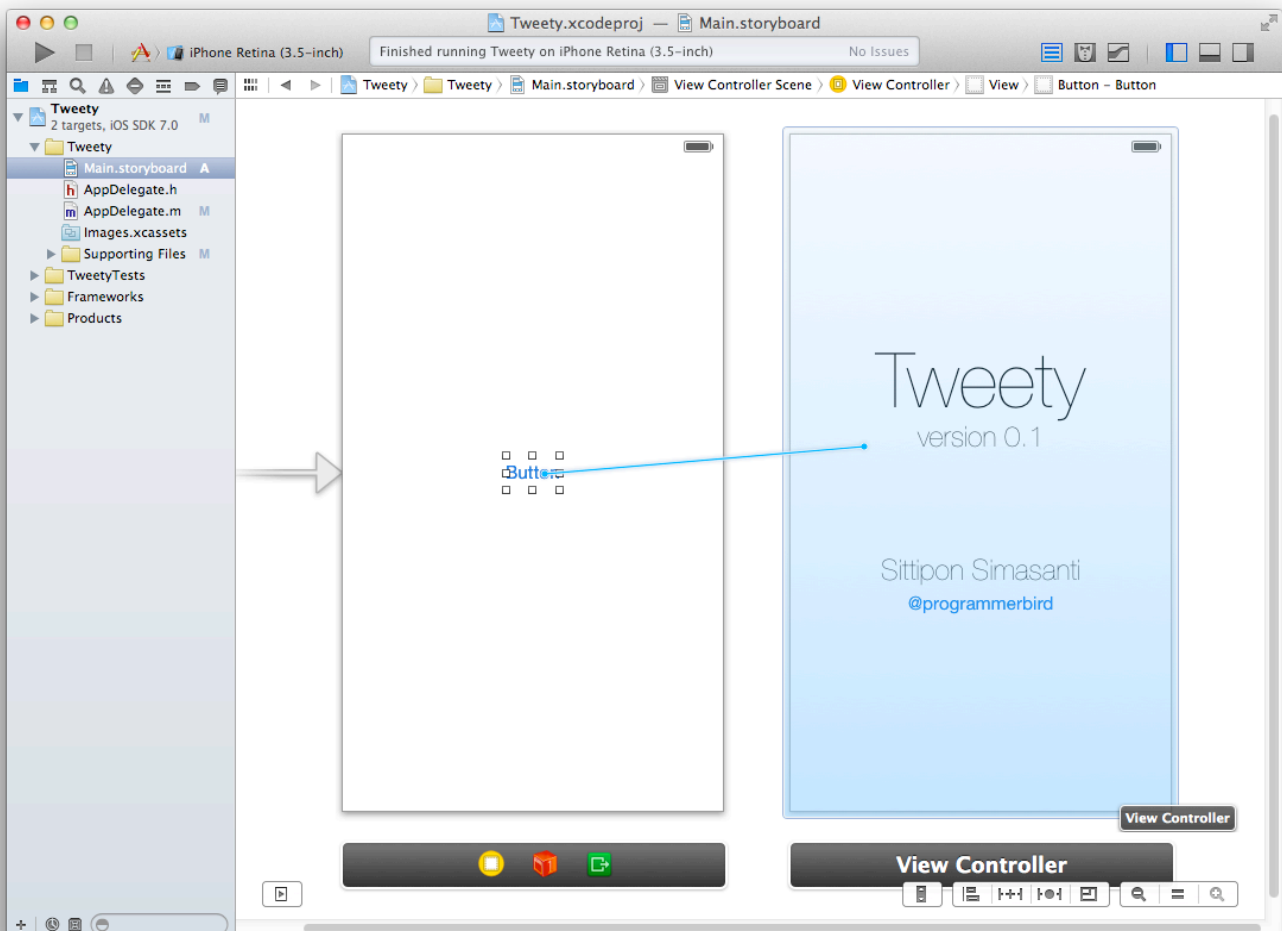
Run ใน Simulator ควรจะเห็นหน้า About แทน แทนหน้า Root อันเดิม

Segue /segwei/

“Segue” เป็นคำที่มาจากภาษาอิตาลี ออกเสียงคล้ายๆ seg-gway (ผม)แปลเป็นภาษาไทยว่า การเปลี่ยนฉาก โดยเราจะกำหนด ให้เมื่อกดปุ่มในฉากหน้า Root เปลี่ยนฉากไปยังหน้า About และจากหน้า About segue กลับไปที่หน้า Root กัน

จากหน้า Root segue ไปหน้า About

1. ย้าย ลูกศรชี้เข้า กลับไปที่หน้า Root เพื่อให้ App เริ่มทำงานที่หน้า Root
2. กด Mouse ขวา ที่ปุ่มในหน้า Root แล้วลากไปที่ หน้า About เพื่อบอกว่าเมื่อกดที่ปุ่มนี้ให้ segue ไปที่หน้า About



3. เลือก Action Segue เป็นแบบ Modal ไปก่อน

Checkpoint

Run ใน Simulator ควรจะเห็นหน้า Root มี ปุ่ม ที่กดแล้วจะเปลี่ยนไปหน้า About

จากหน้า About segue กลับไปหน้า Root

Segue แบบ Modal ใช้สำหรับแสดง Dialog แบบชั่วคราว โดยนำจาก View Controller อันใหม่มาทับจากเก่า โดยที่จากเก่าก็ยังคงอยู่ใน Memory เหมือนเดิม หากเราใช้ Modal Segue อีกจะทำให้เกิดจากซ้อนกันไปมามากขึ้นเรื่อยๆ ดังนั้นเมื่อตอนจะกลับออกไป เราจะต้อง Segue แบบอื่นเพื่อระบุว่า ให้กลับไปจากที่เคยเรียกมา

Unwind Segue

เราจะใช้ Unwind Segue เพื่อกลับไปจากที่เคยไปหน้า Root โดยในการใช้ Unwind Segue เราจะต้องสร้าง method รูปแบบหนึ่งสำหรับการ Unwinding ดังนั้นเราจะต้องเปลี่ยนหน้า Root จากเดิมที่ใช้ UIViewController เป็น class ใหม่เพื่อจะเพิ่ม method นี้เข้าไป

1. กด Mouse ขวาที่เมนูด้านซ้าย แล้วสร้างไฟล์ใหม่เข้าไปใน Project เป็น Objective-C Class ชื่อ RootViewController โดยเป็น subclass มาจาก UIViewController
2. เพิ่ม method `-(IBAction)returnToRoot:(UIStoryboardSegue *)segue` ให้ RootViewController

RootViewController.h

```
- (IBAction)returnToRoot: (UIStoryboardSegue *)segue;
```

RootViewController.m

```
- (IBAction)returnToRoot: (UIStoryboardSegue *)segue {
}
```

3. ที่ Main.storyboard เลือกจาก Root, ที่แถบสีดำด้านล่าง เลือก Icon สีเหลืองด้านซ้ายสุดที่ ชื่อ View Controller
4. ที่เมนู Inspector ด้านขวา เปลี่ยน Custom Class จาก UIViewController เป็น RootViewController



Checkpoint

เมื่อกด Mouse ขวาที่ Icon Exit ขวาสุดสีเขียว ของจากใดก็ตาม เราจะได้เห็น returnToRoot: เป็น Unwind Segue ที่สามารถใช้ได้

5. ที่หน้า About สร้าง UIButton ชื่อ Done
6. Mouse ขวาที่ปุ่ม Done ลากไปที่ Icon Exit ขวาสุดสีเขียวของจาก About แล้วเลือก returnToRoot:

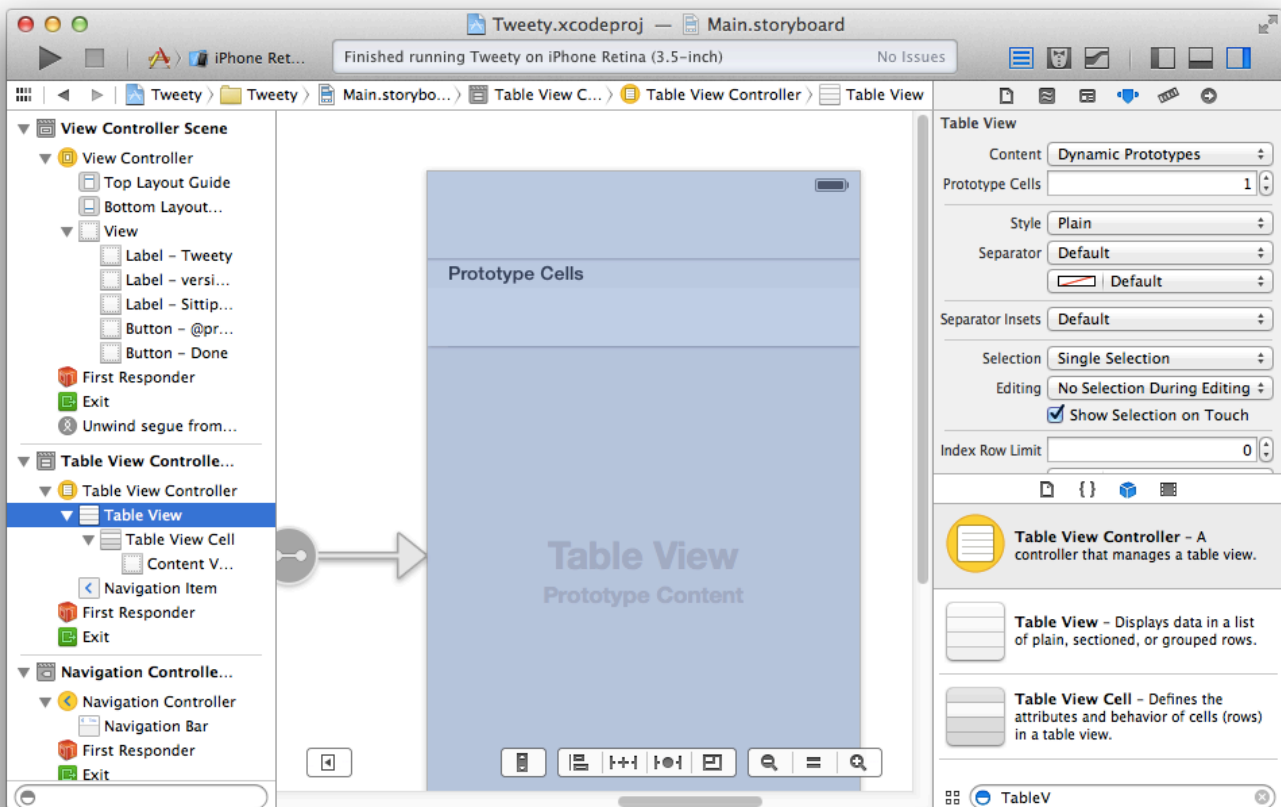
Checkpoint

Run ใน Simulator แล้วสามารถไปกลับ ระหว่างหน้า Root และ About ได้

หน้า Help & Settings

เราจะเปลี่ยนปุ่มในหน้า Root เป็นปุ่มเมนู และเมื่อกดเข้าไปแทนที่จะเปิดหน้า About เข้ามาให้เห็นรายการประกอบด้วย About, Tell Friends และ Email Support

1. เปลี่ยนปุ่มในหน้า Root ให้แสดงรูป menu.png แทนข้อความ
2. เลือก Segue จากหน้า Root ไปยังหน้า About แล้วกด Delete เพื่อลบทิ้ง
3. ลาก Table View Controller เข้ามาใน Storyboard
4. โดยทั่วไปเรานิยมใส่ Table ไว้ใน Navigation Controller เพื่อช่วยจัดการปุ่ม Back และ เมนูด้านบนของ App, ให้เราเพิ่ม Navigation Controller ด้วยการเลือก Table View Controller ที่สร้างมาใหม่ แล้วเลือก Editor > Embed In > Navigation Controller จากเมนูบาร์
5. เลือก Table View ใน Table View Controller เปลี่ยน Content จาก Dynamic Prototypes เป็น Static Cell จากเมนู Inspector ด้านขวา



6. ข้างใต้ Table View จะมี Table View Section, ในเมนู Inspector จะเห็นว่าตั้งไว้ ที่ 3 Rows ลองเปลี่ยน Header และ Footer ดู
7. เลือกที่ Table View Cell, ใน เมนู Inspector เปลี่ยน Style เป็น Basic, Accessory เป็น Disclosure Indicator จากนั้น Double click ที่คำว่า Title แล้วแก้เป็น About
8. เปลี่ยน Table View Cell ที่เหลือเป็น Tell Friends แล้ว Email Support
9. ที่ Navigation Item ของ Table View Controller เปลี่ยน Title เป็น Help & Settings

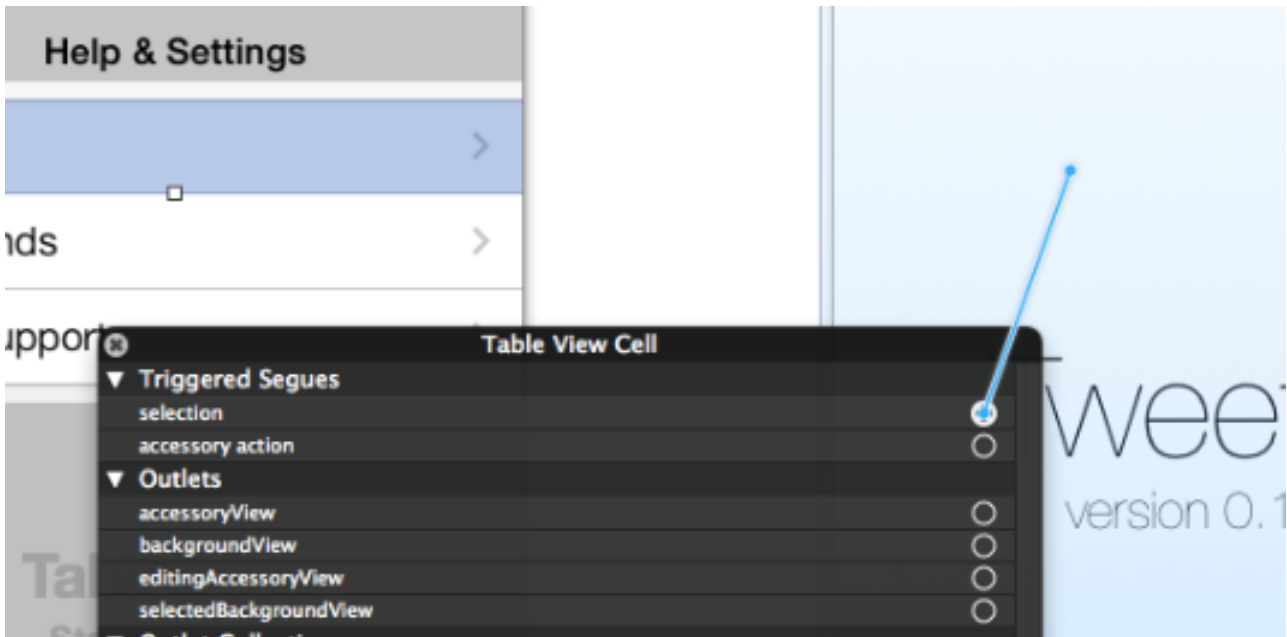
Checkpoint

ลากลูกศรชี้เข้าไปที่ Navigation Controller แล้ว Run ใน Simulator เห็นเป็นหน้า Help & Settings

Winding All Together

เราจะผูกหน้า Root, Help & Settings และหน้า About เข้าด้วยกัน

1. ย้ายลูกศรชี้เข้ากลับไปหน้า Root
2. กด Mouse ขวาที่ปุ่มเมนูในหน้า Root ลากไปที่หน้า Navigation Controller สร้างเป็น segue แบบ modal ในลักษณะเดียวกันกับหน้า About ก่อนหน้านี้
3. กด Mouse ขวาที่ปุ่ม About ในหน้า Help & Settings ใน Triggered Segues เลือก *selection* แล้วลากไปที่หน้า About สร้างเป็น segue แบบ push เพื่อให้ segue เมื่อเกิด *selection* ที่ Cell



Checkpoint

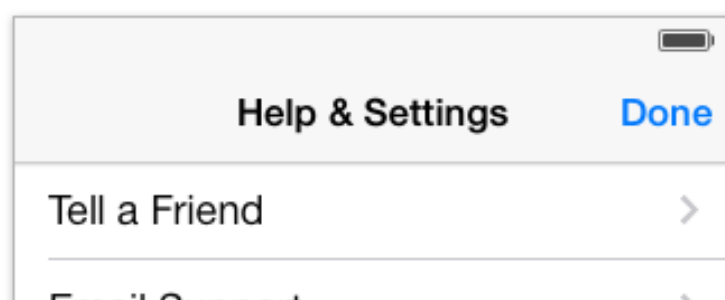
Run ใน Simulator จะเริ่มจากหน้า Root ไปหน้า Help & Settings และ ผ่านไปที่หน้า About ได้

Mission 6.1

1. ถ้าเปลี่ยน Segue จากหน้า Help & Settings ไปหน้า About ให้เป็นแบบ Modal แล้วแตกต่างจากเดิมอย่างไร
2. ลองใช้ปุ่ม Done ที่อยู่ในหน้า About ว่ายังสามารถกลับไปหน้า Root ได้หรือไม่

Mission 6.2

เพิ่มปุ่ม Done ในหน้า Help & Settings segue กลับไปที่หน้า Root



TweetViewController

ต่อจากนี้เราเตรียม View Controller ที่จะใช้แสดง Tweet สำหรับแต่ละข้อความกัน

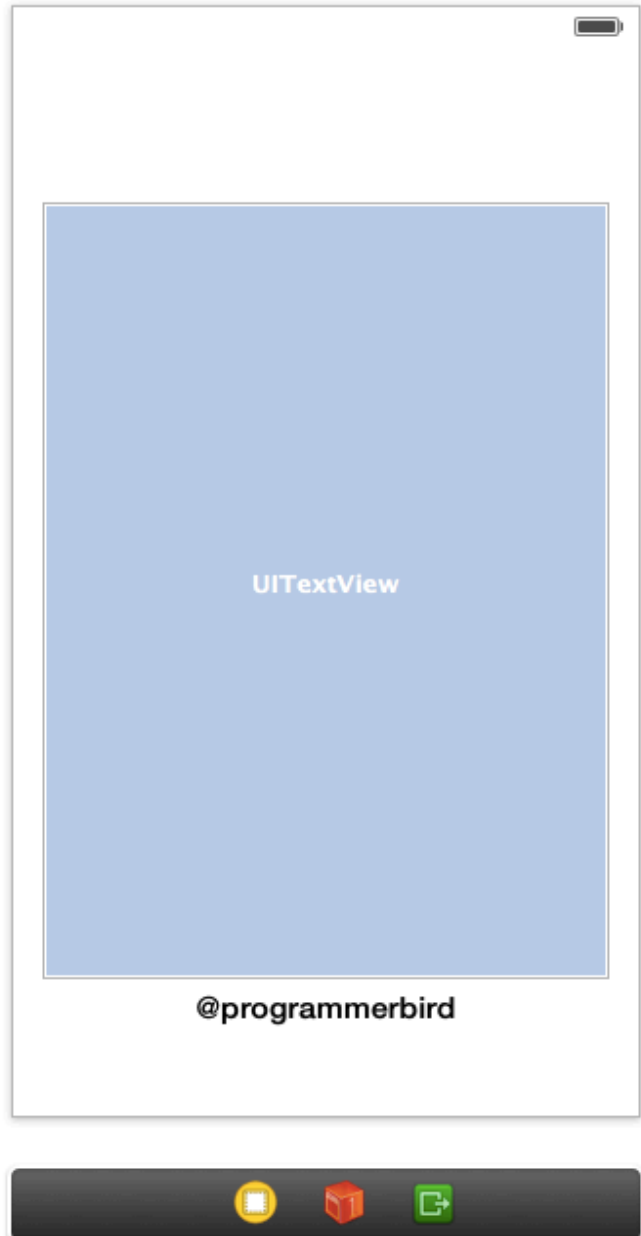
1. กด Mouse ขวาที่เมนูด้านซ้าย แล้วสร้างไฟล์ใหม่เข้าไปใน Project เป็น Objective-C Class ชื่อ TweetViewController โดยเป็น subclass มาจาก UIViewController
2. สร้าง View Controller อันใหม่ใน Storyboard ใหม่ แล้วกำหนด Custom Class ให้เป็น TweetViewController
3. ใส่ UILabel ชื่อ screenNameLabel, และ UITextView ชื่อ textView ผูกเข้ากับ TweetViewController ให้เรียบร้อย
4. จัด TweetViewController ให้สวยงาม โดยให้ screenNameLabel อยู่ด้านล่าง และ UITextView อยู่ตรงกลาง ลองใส่ข้อความลงไปเพื่อทดสอบหน้าตา
5. ย้ายลูกศรชี้เข้า มาชี้ที่ TweetViewController

Checkpoint

Run ใน Simulator ได้หน้าตาที่สวยงาม สามารถแก้ screenNameLabel.text จาก viewDidLoad ใน TweetViewController ได้

เมื่อเปิด Tweety ใน Simulator ของ iPhone Retina 3.5 นิ้ว, และ 4 นิ้ว จะพบว่า ใน Simulator อันหนึ่ง ตำแหน่งของ screenNameLabel อยู่ต่ำหรือสูงเกินไป

เราจะแก้ปัญหานี้ด้วย Auto Layout




รู้จักกับ Auto Layout

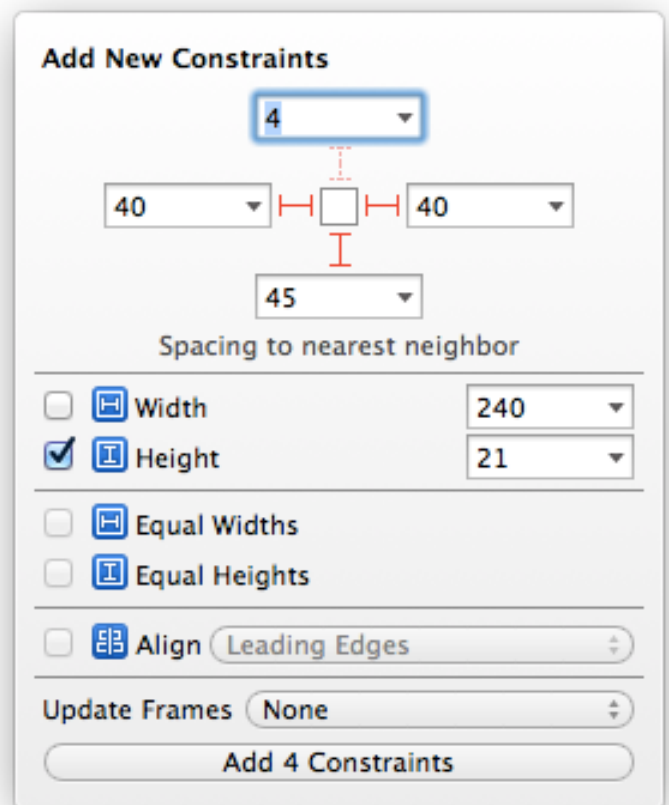
Auto Layout เป็นระบบที่ใช้จัดการ ตำแหน่งและขนาดของ UI เพื่อรองรับ iPhone 5 ที่มีขนาดหน้าจอต่างจาก iPhone 4 แม้จะสามารถนำไปใช้กับ iPad ได้ด้วย แต่ก็ไม่นิยมใช้กัน เพราะขนาดจริงของหน้าจอ iPhone และ iPad ต่างกันมากเกินไป

สำหรับ App ที่ต้องการรองรับ ทั้ง iPhone และ iPad เราจะแยก Storyboard ออกเป็น 2 ไฟล์ ไฟล์หนึ่งออกแบบไว้สำหรับ iPhone โดยเฉพาะ และอีกไฟล์สำหรับ iPad โดยเฉพาะ ซึ่งเราจะไม่พูดถึงในกิจกรรมนี้

TweetViewController (cont.)

ตำแหน่งของ `screenNameLabel`

1. เลือกที่ `screenNameLabel`
2. กดที่เมนู Pin ด้านล่าง
3. เราจะยึดให้ `screenNameLabel` อยู่ห่างจาก ขอบล่าง, ขอบซ้าย, ขอบขวา ตามเดิม ใน Add new Constraints กดขอบซ้าย ขวา ล่าง ให้เป็นเส้นทึบสีแดง (ในตัวอย่างคือ 45, 40, 40 ตามลำดับ)
4. ขอบบนปล่อยไว้ แต่กำหนดความสูงเป็น 21 เท่าเดิม ด้วยการติ๊กถูกที่หน้าช่อง Height
5. กด Add 4 Constraints เพื่อเพิ่ม Constraints จะมี Constraints เพิ่มมาในเมนูด้านขวา
6. กด  เพื่อลองเปลี่ยนไปดูในจอ iPhone แบบ 3 นิ้ว จะพบว่า `screenNameLabel` เลื่อนตามขึ้นมา



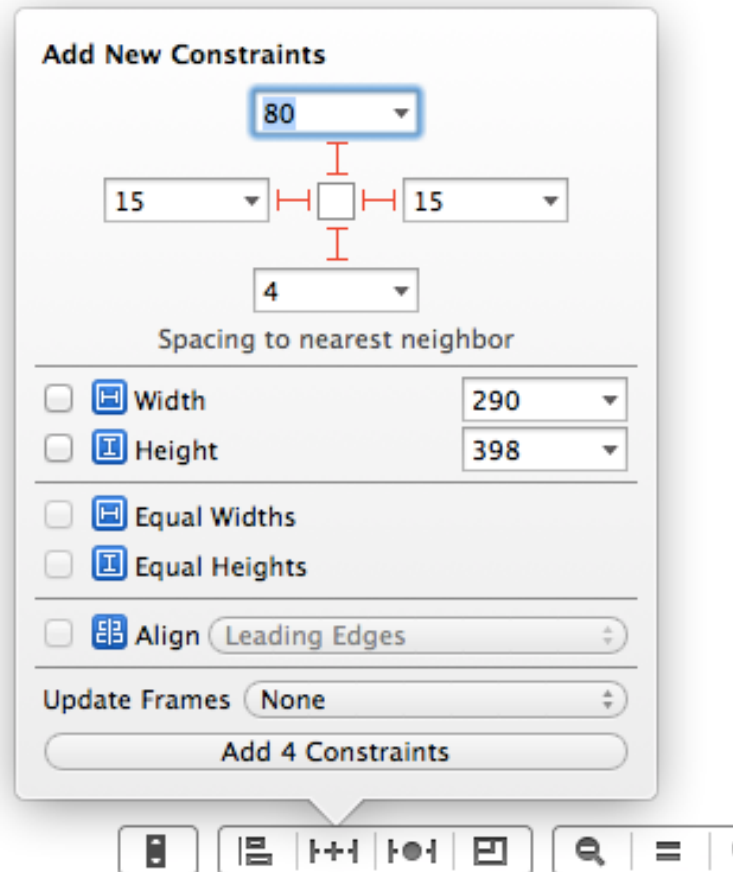
หลังจากนี้หากเราเลื่อนตำแหน่งของ `screenNameLabel` มันจะฟ้องขึ้นเป็นเส้นสีส้ม บอกว่าไม่ตรงตาม Constraints ที่ตั้งไว้ในเมนูด้านขวา ข้างชื่อ Tweet View Controller จะมี ลูกศรขาวในวงกลมสีส้มขึ้นมา กดเข้าไปจะมีรายการของ Component ที่อยู่ติดอยู่ กดที่สามเหลี่ยมสีส้ม เพื่อแก้ Constraints

ตำแหน่งของ textView

1. เลือกที่ textView
2. กดที่เมนู Pin ด้านล่าง
3. เราจะยึด textView ให้อยู่สูงจาก screenNameLabel, ขอบบน, ขอบซ้าย และ ขอบขวา ตามเดิม กดขอบ บน ซ้าย ขวา ล่าง ให้เป็นเส้นทึบสีแดง
4. กด Add 4 Constraints
5. ลองเปลี่ยนไปดู จอ iPhone แบบ 3 นิ้ว จะพบว่า textView ปรับขนาดให้หดลงตามหน้าจอ

Checkpoint

เมื่อ Run ใน Simulator ตำแหน่ง screenNameLabel และ textView อยู่ในตำแหน่งที่เหมาะสม



Swipe เปลี่ยนหน้า

ต่อจากนี้เราจะผูกหน้า Tweet เข้ากับ RootViewController ผ่าน UIPageViewController เพื่อให้ Swipe ไปมาได้ เสียดยที่เราไม่สามารถโยก จาก Storyboard เราจึงต้องสร้าง UIPageViewController ขึ้นมาเองผ่าน Code

Learning By Example

แต่ก็โชคดีที่ Apple ได้ใส่ตัวอย่างมาให้ด้วย

Code ในหน้าต่อจากนี้ดัดแปลง มาจาก Project ประเภท Page-based Application ที่อยู่ตอน Create New Project โดยเปลี่ยน DetailViewController เป็น TweetViewController ของเรา และรวม ModelController เข้ามาใส่ใน RootViewController เพื่อความกระชับ

ส่วนที่ต้องพิมพ์นั้น จัดว่าไม่ได้มีเยอะมาก ฉะนั้นพิมพ์ตามดูไปเลย

IOS Application Development - Tweety!

สร้าง property pageController ให้ RootViewController

RootViewController.h

```
@interface RootViewController : UIViewController
    <UIPageViewControllerDelegate, UIPageViewControllerDataSource>

@property (nonatomic, strong) IBOutlet UIPageViewController *pageController;

-(IBAction)returnToRoot:(UIStoryboardSegue *)segue;
@end
```

สร้าง property index ให้ TweetViewController

TweetViewController.h

```
@property (assign) int index;
```

สร้าง method tweetControllerAtIndex: สำหรับสร้าง TweetViewController จาก Storyboard

RootViewController.m (cont.)

```
-(TweetViewController*)tweetControllerAtIndex: (int)index
{
    TweetViewController *tweetController = [self.storyboard instantiateViewControllerWithIdentifier:
                                                @"TweetViewController"];
    [tweetController setIndex: index];
    return tweetController;
}
```

สร้าง **UIPageViewController** ใน **viewDidLoad** ของ **RootViewController** (อย่าลืม #import “TweetViewController.h” ด้านบน)

RootViewController.m

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view. Configure the page view controller and add it
    // as a child view controller.
    UIPageViewController *pageController = [[UIPageViewController alloc]
        initWithTransitionStyle: UIPageViewControllerTransitionStyleScroll
        navigationOrientation: UIPageViewControllerNavigationOrientationHorizontal
        options: nil];

    TweetViewController *firstPage = [self tweetControllerAtIndex: 0];
    NSArray *pages = @[firstPage];
    [pageController setViewControllers: pages
                     direction: UIPageViewControllerNavigationDirectionForward
                     animated: NO
                     completion: nil];
    [self addChildViewController: pageController];
    [self.view addSubview: pageController.view];

    // We don't want page view to stay above the settings button. Otherwise, our button will not tappable.
    [self.view sendSubviewToBack: pageController.view];

    // Set the page view controller's bounds using an inset rect so that self's view is visible around
    // the edges of the pages.
    [pageController.view setFrame: self.view.bounds];
    [pageController setDelegate: self];
    [pageController setDataSource: self];
    [pageController didMoveToParentViewController: self];

    // Add the page view controller's gesture recognizers to the book view controller's view so that
    // the gestures are started more easily.
    [self.view setGestureRecognizers: pageController.view.gestureRecognizers];
    self.pageController = pageController;
}
```

สร้าง method ที่จำเป็นให้ **pageController** ทำงานได้ ตาม protocol ของ **UIPageViewControllerDelegate**, และ **UIPageViewControllerDataSource** ใน **RootViewController**

RootViewController.m (cont. 2)

```
-(UIViewController*)pageViewController: (UIPageViewController *)pageViewController
    viewControllerAfterViewController: (UIViewController *)viewController
{
    TweetViewController *tweetController = (TweetViewController*)viewController;
    return [self tweetControllerAtIndex: tweetController.index + 1];
}

-(UIViewController *)pageViewController: (UIPageViewController *)pageViewController
    viewControllerBeforeViewController: (UIViewController *)viewController
{
    TweetViewController *tweetController = (TweetViewController*)viewController;
    return [self tweetControllerAtIndex: tweetController.index - 1];
}
```

Checkpoint

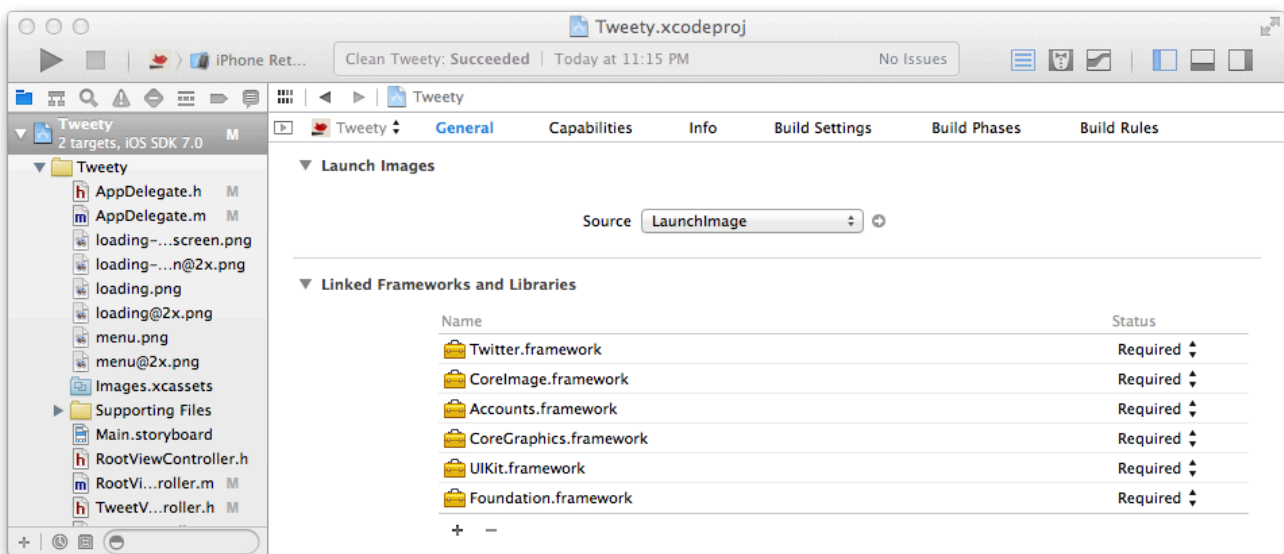
ย้ายลูกศรชี้เข้าไปที่หน้า Root แล้ว Run ใน Simulator เราจะสามารถ Swipe เพื่อเปลี่ยนหน้า Tweet ไปมาได้ หากกดที่ปุ่มเมนู ก็จะพบหน้า Help & Settings ทำงานได้ถูกต้อง

Twitter SDK

ก่อนจะใช้ Twitter SDK ได้ เราจะต้องเพิ่ม Framework ที่ Twitter SDK ที่จำเป็นต้องใช้เข้าไปใน Project ก่อน

เพิ่ม **Accounts.framework**, และ **Twitter.framework** เข้าไปใน **Project**

1. ที่เมนูด้านซ้าย เหลือ Project Tweety
2. ด้านล่างสุดในหมวด Linked Frameworks and Libraries กดที่เครื่องหมาย +
3. เลือก Accounts.framework แล้วกด Add
4. เลือก Twitter.framework แล้วกด Add



Connect to Twitter

ตอนนี้เราพร้อมแล้วที่จะใช้ Twitter SDK ถึงเวลาเขียน Code เพื่อต่อกับ Twitter แล้ว!

It's a good day to learn Objective-C!

แต่ก็โชคดีที่ Twitter ได้เตรียมตัวอย่างมาให้เช่นกัน

<https://dev.twitter.com/docs/ios/making-api-requests-slrequest>

จากตัวอย่างเป็นการดึง User Timeline, (เรา Tweet อะไรบ้าง) แต่สิ่งที่เราต้องการคือ Home Timeline (คนที่เรา Follow อยู่ Tweet อะไรบ้าง) เราจึงต้องแก้ code ในส่วนที่เป็น url โดยเราจะนำ Code มาใส่ไว้ใน AppDelegate สำหรับในส่วนนี้แนะนำให้ไปที่หน้า Web ของ Twitter ด้านบนแล้ว copy code มาแก้ตามหน้าต่อจากนี้

เพิ่ม property **accountStore** ตามตัวอย่างจากหน้าเว็บ Twitter

เพิ่ม property **timeline** เพื่อเก็บที่ Twitter return ออกมา

AppDelegate.h

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

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;
@property (strong, nonatomic) ACAccountStore *accountStore;
@property (strong, nonatomic) NSArray *timeline;

@end
```

เพิ่ม method **twitterDidGetTimeline** เพื่อจัดการ timeline ที่ Twitter return ออกมา

AppDelegate.m (cont. 2)

```
-(void)twitterDidGetTimeline: (NSArray*)timelineData {
    NSLog(@"twitterDidGetTimeline!");
    self.timeline = timelineData;
}
```

ดึง Home Timeline จาก Twitter

AppDelegate.m

```
-(BOOL)userHasAccessToTwitter
{
    return [SLComposeViewController
        isAvailableForServiceType: SLServiceTypeTwitter];
}

- (void)fetchHomeTimeline
{
    // Step 0: Check that the user has local Twitter accounts
    if ([self userHasAccessToTwitter]) {

        // Step 1: Obtain access to the user's Twitter accounts
        ACAccountType *twitterAccountType =
            [self.accountStore accountTypeWithAccountTypeIdentifier:
                ACAccountTypeIdentifierTwitter];

        [self.accountStore
            requestAccessToAccountsWithType:twitterAccountType
            options:NULL
            completion:^(BOOL granted, NSError *error) {

                if (granted) {
                    // Step 2: Create a request
                    NSArray *twitterAccounts =
                        [self.accountStore accountsWithType:twitterAccountType];
                    NSURL *url = [NSURL URLWithString:@"https://api.twitter.com"
                        @"1.1/statuses/home_timeline.json"];
                    NSDictionary *params = @{@"include_rts": @"0",
                        @"trim_user": @"0",
                        @"count": @"30"};

                    SLRequest *request =
                        [SLRequest requestForServiceType:SLServiceTypeTwitter
                            requestMethod:SLRequestMethodGET
                            URL:url
                            parameters:params];

                    // Attach an account to the request
                    [request setAccount:[twitterAccounts lastObject]];
                }
            }];
    }
}
```



```

        // Step 3: Execute the request
        [request performRequestWithHandler:
        ^(NSData *responseData,
        NSHTTPURLResponse *urlResponse,
        NSError *error) {

            if (responseData) {
                if (urlResponse.statusCode >= 200 &&
                urlResponse.statusCode < 300) {

                    NSError *jsonError;
                    NSArray *timelineData =
                    [NSJSONSerialization
                    JSONObjectWithData:responseData
                    options:NSJSONReadingAllowFragments error:&jsonError];
                    if (timelineData) {
                        NSLog(@"Timeline Response: %@\n", timelineData);
                        [self twitterDidGetTimeline: timelineData];
                    }
                    else {
                        // Our JSON deserialization went awry
                        NSLog(@"JSON Error: %@", [jsonError localizedDescription]);
                    }
                }
                else {
                    // The server did not respond ... were we rate-limited?
                    NSLog(@"The response status code is %d",
                    urlResponse.statusCode);
                }
            }
        }];

    }
    else {
        // Access was not granted, or an error occurred
        NSLog(@"%@", [error localizedDescription]);
    }
}];
}
}

```

เรียก `fetchHomeTimeline` เมื่อ App เริ่มทำงาน

AppDelegate.m (cont. 3)

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.accountStore = [[ACAccountStore alloc] init];
    [self fetchHomeTimeline];

    return YES;
}
```

Checkpoint

เปิด Simulator แล้วเข้าไปใน Settings ตั้ง Account Twitter ให้เรียบร้อย จากนั้นสั่ง Run ถ้าทุกอย่างเรียบร้อยเราควรจะได้เห็น Dictionary ขนาดใหญ่ในหน้า Output จาก NSLog ตามด้วยประโยคสุดท้าย twitterDidGetTimeline! ในบรรทัดสุดท้าย

เกี่ยวกับ `fetchHomeTimeline`

ยังมีอยู่หลายจุดที่ยังทำให้ดีกว่านี้ได้ ได้แก่

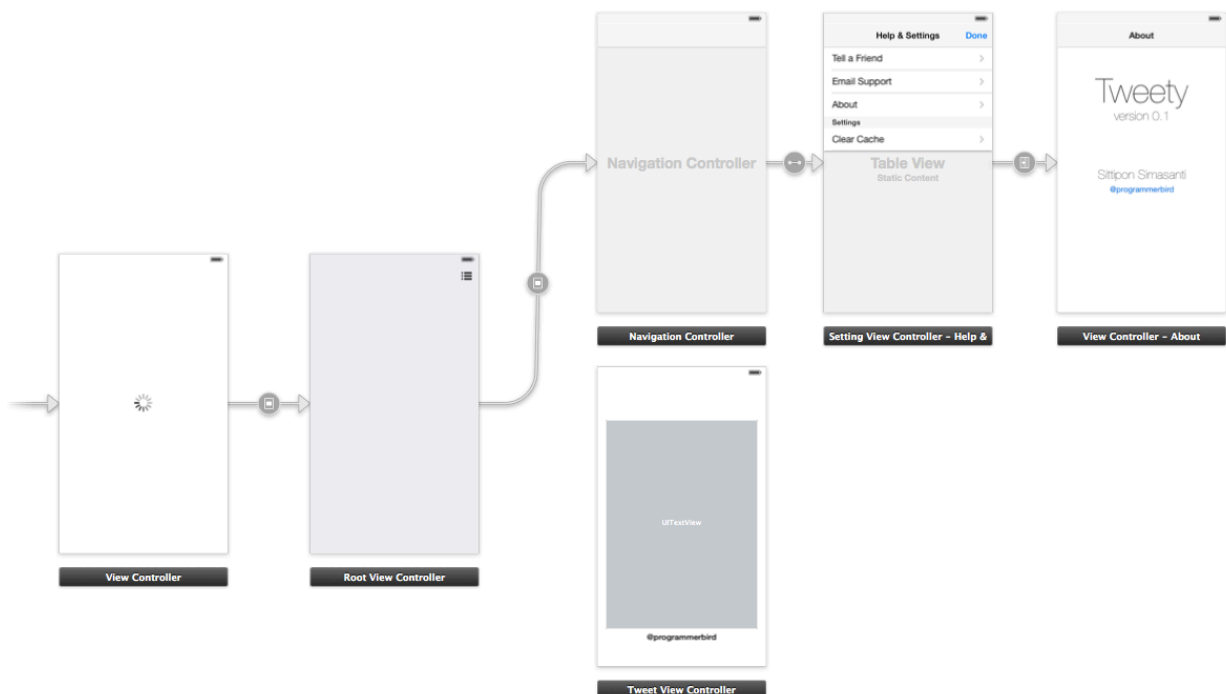
- Step 1 และ Step 2 ควรจะอยู่แยกเป็นคนละ method กัน เพราะหากต้องการ Refresh timeline เราไม่จำเป็นต้องทำ Step ที่ 1 ซ้ำอีกรอบ
- ในกรณีที่มี Twitter หลาย Accounts อยู่ในเครื่อง เราควรมีทางให้ User เปลี่ยน Username ที่จะให้ดึงข้อมูลมาได้ มากกว่าเลือกอันสุดท้ายตลอด
- Error ไม่ควรจะมีพิมพ์ใส่ NSLog อย่างเดียว เราควรจะแจ้ง User ด้วยว่าเกิดผิดพลาดอะไรขึ้นบ้าง
- Timeline ของคนส่วนใหญ่ประกอบด้วย user ซ้ำๆกัน เราควรจะตั้ง trim_user เป็น 1 แล้ว cache ข้อมูลของ User ไว้ในเครื่องมากกว่า

Loading Screen

เราใช้เวลาดึงข้อมูลจาก Twitter ประมาณ 2-3 วินาที เราควรมี Loading Screen ก่อนแสดงหน้า Root เป็นครั้งแรก

เพิ่มหน้า Loading Screen

1. ลาก View Controller เข้ามาใส่ด้านซ้ายมือของหน้า Root
2. ตกแต่ง ด้วยการใส่ UIActivityIndicator แล้วติ๊กถูกที่ Animating ในเมนู Inspector ด้านขวา
3. กด Mouse ขวาที่หน้า Loading ลากไปใส่ที่ หน้า Root เพื่อสร้าง Segue ขึ้นมาใหม่เป็นแบบ push โดย segue นี้ไม่ได้ผูกอยู่กับปุ่ม แต่เราจะเรียกจาก Code เมื่อทำการโหลด Timeline เสร็จ
4. เพราะต้องเรียกจาก code เราจะต้องตั้งชื่อ Segue นี้ ให้กดที่ Segue ใหม่ ที่เมนู Inspector ด้านขวาเลือก ตั้ง Identifier ใน Storyboard Segue ให้เป็น "finishLoading", เลือก Transition เป็น Cross Dissolve
5. ลากลูกศรชี้เข้ามาใส่ที่ Loading Screen ตอนนี้ Storyboard ของเราจะเป็นแบบนี้



6. เพิ่มคำสั่งให้ ทำ segue "finishLoading" เมื่อได้ timeline AppDelegate.m

```
-(void)twitterDidGetTimeline: (NSArray*)timelineData {
    NSLog(@"twitterDidGetTimeline!");
    self.timeline = timelineData;
    [self.window.rootViewController
        performSegueWithIdentifier: @"finishLoading" sender:nil];
}
```

Checkpoint

เมื่อกด Run ใน Simulator จะขึ้นหน้าจอ Loading Screen สักครู่ก่อนเปลี่ยนไปหน้า Root แต่ บางครั้งหน้าจอแรกของ Tweet แสดงเป็นหน้าเปล่าๆ บางครั้งก็ Crash เราจะแก้ปัญหาในอีก 2 หัวข้อถัดไป

แสดง Tweet จาก Timeline

เราจะเป็นไม่เห็นปัญหา จากหัวข้อที่แล้วสักครู่หนึ่ง และเอา Timeline มาแสดงในหน้า TweetViewController โดยผ่านตัวแปร tweetEntry ตอน viewDidLoad

ใส่ข้อมูลให้ TweetViewController

1. ที่ TweetViewController เตรียม property สำหรับเก็บข้อมูล Tweet

TweetViewController.h

```
@property (nonatomic, strong) NSDictionary *tweetEntry;
```

2. ใส่ข้อมูลให้ tweetEntry ให้ TweetViewController (อย่าลืม import AppDelegate.h ที่หัว RootViewController.m)

RootViewController.m

```
-(TweetViewController*)tweetControllerAtIndex: (int)index
{
    AppDelegate *appDelegate = [[UIApplication sharedApplication]
                                delegate];
    NSArray *timeline = [appDelegate timeline];
    if(0 <= index && index < [timeline count])
    {
        TweetViewController *tweetController =
            [self.storyboard instantiateViewControllerWithIdentifier:
             @"TweetViewController"];
        [tweetController setTweetEntry: timeline[index]];
        [tweetController setIndex: index];
        return tweetController;
    }
    return nil;
}
```

3. นำข้อมูลจาก tweetEntry มาแสดงเมื่อ viewDidLoad

TweetViewController.m

```
-(void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    if(self.tweetEntry){
        NSDictionary *user = self.tweetEntry[@"user"];
        self.screenNameLabel.text =
            [NSString stringWithFormat: @"%@", user[@"screen_name"]];
        self.textView.text = self.tweetEntry[@"text"];
    }
}
```

Checkpoint
Crash!

Crash!

จากการตรวจสอบ Log ในหน้าจอ Output เราจะพบบรรทัดหนึ่งหน้าตาลักษณะนี้

```
2013-10-11 07:23:29.871 Tweety[15872:1403] *** Terminating app due to
uncaught exception 'NSInternalInconsistencyException', reason: 'Only
run on the main thread!'
```

Only run on the main thread!

โดยปกติแล้ว App iOS จะทำงานอยู่บน Thread เดียวเสมอ ไม่ว่าจะเป็นคำนวณค่าต่างๆ หรือ แสดงผลทั้งหมดอยู่ใน Thread ที่เรียกว่า Main Thread. Component ต่างๆ ถูก Optimize อย่างเต็มที่เพื่อให้กินทรัพยากรน้อยที่สุด จึงไม่ได้ทำเป็น thread safe ไว้เลย. เวลาแก้ค่าต่างๆ ให้ Component ไม่ว่าจะเป็น UILabel, UITextView, เปลี่ยนสีพื้นหลัง จึงต้องทำตอนที่อยู่บน Main Thread เสมอ

เราเปลี่ยน Thread ไปตอนไหน ?

อีกคำถามหนึ่งที่อธิบายได้ง่ายกว่า คือ ถ้า App ของเราไม่ได้เปลี่ยนไป Thread อื่นเลยจะเป็นอย่างไร ?

ในระหว่างที่เราทำการ Load Timeline จาก Twitter มา จะไม่มีอะไรมา Update UI ให้ Activity Indicator ที่เราใส่ใน Load Screen จะไม่สามารถหมุนได้ ฉะนั้นเมื่อต้องทำงานหนักๆ หรืองานที่ใช้เวลานาน เช่นการ Load Timeline, เรานิยมแตก Thread ไปทำเป็นงานด้านหลัง และปล่อยให้ Main Thread จัดการ UI ต่อเอง. หากเราสังเกต Code ของ Twitter เราจะเห็นว่าส่วนใหญ่ที่เราใส่ไปเป็นลักษณะของ Callback เมื่อ method ทำงานเสร็จ

ในเมื่อ Library เป็นตัวเปลี่ยนไปใช้ Thread อื่น, ตอน Callback ทำไมถึงไม่นิยมเปลี่ยนกลับเป็น Main Thread ?

Library ส่วนใหญ่ของ iPhone เน้นประสิทธิภาพก่อนความสะดวกของโปรแกรมเมอร์เสมอมา เพราะการเปลี่ยน Thread มีต้นทุน และหลังจากทำงานหนักๆ เสร็จ มักจะต้องทำงานหนักๆ ต่อ เช่น บันทึกผลลัพธ์ที่ได้ลงไฟล์ ซึ่งก็ไม่ควรอยู่ใน Main Thread อยู่ดี Library ส่วนใหญ่ของ iOS จึงปล่อยให้เป็นที่ของโปรแกรมเมอร์ที่จะเปลี่ยนเป็น Main Thread กลับมาเอง

จากเหตุผลที่กล่าวมาทั้งหมด หลัง Callback ของ Twitter เราจะต้องเปลี่ยนเป็น Main Thread กลับมา

เปลี่ยนกลับเป็น Main Thread

โชคดีที่การเปลี่ยน Thread ในปัจจุบันทำได้ง่ายมาก โดยอาศัยระบบจัดการ Thread ที่ชื่อว่า Grand Central Dispatcher, สำหรับ Library นี้เราจะเรียก Thread ต่างๆ ว่า Queue (มันมีเหตุผลที่ดีอยู่)

AppDelegate.m ใส่ dispatch_async ตามด้วย queue ที่จะเปลี่ยน คร่อมการ segue

```
dispatch_async(dispatch_get_main_queue(), ^{
    [self.window.rootViewController
        performSegueWithIdentifier:@"finishLoading" sender:nil];
});
```

Checkpoint

Tweety ของเรา ไม่ Crash แล้ว สามารถดึงข้อมูลจาก Twitter มาแสดงผลได้, Swipe ไปมาเพื่อดู Tweet ถัดไป, ก่อนหน้าได้ปกติ, กดที่ Settings เข้าไปดู About ได้ปกติ

ดึง Profile Picture มาใส่เป็น Background

ต่อจากนี้เราจะดึงรูป Profile Picture มาใส่เป็นพื้นหลัง ถ้าเราดูในหน้าต่าง Output จะเห็น profile_image_url มาด้วย ลองโหลดมาดูจะได้เป็นภาพ profile ที่มีขนาดเล็กเกินกว่าที่จะทำเป็นพื้นหลัง แต่ถ้าตัด _normal ออกที่ท้าย url จะได้รูปขนาดต้นฉบับที่ upload ขึ้นมา เราจะใช้รูปนี้ทำเป็น Background กัน

Profile Picture

1. ในหน้า Root

- เพิ่ม UIImageView ชื่อ profileBackView ปรับ Layout ให้ขนาดเต็มจอ และตั้ง mode ให้เป็น Aspect Fill
- เพิ่ม UIImageView ชื่อ profileForeView ปรับ Layout ให้ขนาดเต็มจอ และตั้ง mode ให้เป็น Aspect Fill เช่นกัน
- เพิ่ม UIImageView ชื่อ overlayView ปรับ layout ให้ขนาดเต็มจอ เช่นกัน ตั้ง background color เป็นสีเทา, Opacity ตั้งเป็น 70% วางทับ profileForeView

2. ใน viewDidLoad ของ RootViewController ย้าย 3 รูปนี้ไปไว้ข้างหลัง pageController.view

RootViewController.m ใน method viewDidLoad

```
[self.view addSubview: pageController.view];

[self.view addSubview: self.overlayView];
[self.view addSubview: self.profileForeView];
[self.view addSubview: self.profileBackView];
```

3. เพิ่ม method สำหรับโหลดรูป ตอนนี้ให้เปลี่ยนสี Background ไปก่อน

RootViewController.m (อย่าลืมเพิ่ม method ใน header file)

```
-(void)reloadUserPicture: (NSDictionary *)user {
    [self.profileBackView setBackgroundColor: [UIColor blueColor]];
}
```

4. เพิ่ม RootViewController เข้าไปใน AppDelegate เพื่อเตรียมจะเรียกจากหน้า Tweet

AppDelegate.h (อย่าลืม import "RootViewController.h" ที่หัวไฟล์)

```
@property (weak, nonatomic) RootViewController *rootController;
```

3. ใส่ค่า `appDelegate.rootController` ตอนที่ `RootViewController` ถูกโหลดขึ้นมา

`RootViewController.m` ใน method `viewDidLoad`

(อย่าลืม import “AppDelegate.h” ที่หัวไฟล์)

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

    AppDelegate *appDelegate = [[UIApplication sharedApplication]
delegate];
    [appDelegate setRootController: self];

    ...
}
```

4. ใน `TweetViewController` สั่ง `reloadUserPicture:` เมื่อ view แสดงขึ้นมา

`TweetViewController.m`

```
-(void)viewDidAppear:(BOOL)animated {
    if(self.tweetEntry)
    {
        NSDictionary *user = self.tweetEntry[@"user"];
        AppDelegate *appDelegate = [[UIApplication sharedApplication]
delegate];
        [appDelegate.rootController reloadUserPicture: user];
    }
}
```

Checkpoint

เมื่อ Run ขึ้นมาจะได้หน้า Tweet สีฟ้า, ปุ่มเมนูยังใช้ได้ปกติ

ในส่วน ต่อจากนี้เราจะเขียน `reloadUserPicture:` ให้ดีขึ้น

reloadUserPicture:

แบบที่ 1: โหลดรูปมาแล้วแสดงผลทันที

```
-(void)reloadUserPicture: (NSDictionary *)user {
    NSString *profileImageString = [user[@"profile_image_url"]
                                     stringByReplacingOccurrencesOfString:@"_normal." withString:@"."];
    NSURL *profileImageUrl = [NSURL URLWithString: profileImageString];
    NSData *urlData = [NSData dataWithContentsOfURL: profileImageUrl];
    UIImage *image = [UIImage imageWithData: urlData];
    [self.profileBackView setImage: image];
}
```

แบบที่ 2: ดูนใน cache ก่อน ถ้าไม่มีค่อยโหลดรูปมา

```
-(void)reloadUserPicture: (NSDictionary *)user {

    NSString *cacheFileName = [NSString stringWithFormat: @"%@.png", user[@"screen_name"]];
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSString *cachePath = [documentsDirectory stringByAppendingPathComponent: cacheFileName];

    NSFileManager *fileManager = [NSFileManager defaultManager];
    if(![fileManager fileExistsAtPath: cachePath])
    {
        NSString *profileImageString = [user[@"profile_image_url"]
                                         stringByReplacingOccurrencesOfString:@"_normal." withString:@"."];
        NSURL *profileImageUrl = [NSURL URLWithString: profileImageString];
        NSData *urlData = [NSData dataWithContentsOfURL: profileImageUrl];
        [urlData writeToFile:cachePath atomically:YES];
    }
    UIImage *image = [UIImage imageWithContentsOfFile: cachePath];
    [self.profileBackView setImage: image];
}
```


แบบที่ 3: ดูปใน cache ก่อน ถ้าไม่มีค่อยโหลดรูปมา ผ่าน Thread อื่น, แล้วเปลี่ยนกลับไป Main Thread หลังได้รูปมาแล้ว

```
-(void)reloadUserPicture: (NSDictionary *)user {

    NSString *cacheFileName = [NSString stringWithFormat:@"%s.png", user[@"screen_name"]];
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSString *cachePath = [documentsDirectory stringByAppendingPathComponent: cacheFileName];

    NSFileManager *fileManager = [NSFileManager defaultManager];
    if([fileManager fileExistsAtPath: cachePath]){

        UIImage *image = [UIImage imageWithContentsOfFile: cachePath];
        [self.profileBackView setImage: image];

    }else {

        dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{

            NSString *profileImageString = [user[@"profile_image_url"]
                                             stringByReplacingOccurrencesOfString:@"_normal." withString:@"."];
            NSURL *profileImageUrl = [NSURL URLWithString: profileImageString];
            NSData *urlData = [NSData dataWithContentsOfURL: profileImageUrl];
            [urlData writeToFile:cachePath atomically:YES];

            dispatch_async(dispatch_get_main_queue(), ^{
                UIImage *image = [UIImage imageWithContentsOfFile: cachePath];
                [self.profileBackView setImage: image];
            });

        });

    }

}
```

Core Animation

แทนที่เราจะเปลี่ยนรูปทันทีที่เราจะค่อยๆ fade รูปใหม่ขึ้นมาผ่าน method fadeUserPicture: ผ่าน Core Animation โดยการกำหนด เวลาที่จะใช้ในการ animate กับสถานะปลายทางที่ต้องการ Core Animation จะค่อยๆ Tween ไปตามเวลาที่เรากำหนด

```
[UIView animateWithDuration: <วินาที>
    animations:^(
        <ปลายทาง>
    )];
```

Fade Background

1. สร้าง method fadeUserPicture ใน RootViewController โดยเราจะ swap เอารูปเก่าไปใส่ใน profileBackView แล้วใส่รูปใหม่ไว้ที่ profileForeView แล้วค่อยๆ fade รูปใหม่ขึ้นมา

RootViewController.m

```
-(void)fadeUserPicture: (UIImage*)image
{
    [self.profileBackView setImage: self.profileForeView.image];
    [self.profileForeView setAlpha: 0.0f];
    [self.profileForeView setImage: image];

    [UIView animateWithDuration:0.8f
        animations:^(
            [self.profileForeView setAlpha: 1.0f];
        )];
}
```

2. ใน reloadUserPicture: เปลี่ยนการ set รูปตรงๆ เป็น

```
[self.profileBackView setImage: image];
```

เป็น

```
[self fadeUserPicture: image];
```

Checkpoint

Run ใน Simulator ได้ Tweety ที่เปลี่ยน Background ได้สวยงาม

นี่เป็น Checkpoint สุดท้ายของ เนื้อหาในคอร์สนี้แล้ว ในกิจกรรมนี้ เราได้ทำความรู้จักกับ ระบบ Storyboard, การทำ Segue, ระบบ Auto Layout, Grand Central Dispatcher, Core Animation และเรื่องอื่นๆอีกมากมาย นับเป็นก้าวแรกที่ไม่เลวเลยทีเดียว

ยินดีด้วยครับ !

Code Listings

สามารถ Download source code ของ กิจกรรมนี้ได้ที่นี่

```
git clone git://github.com/programmerbird/ios-tweety.git
```

Finishing Touch

สำหรับคนที่ยังต้องการทำต่อ

หน้า Settings

สำหรับปุ่มที่เหลือนในหน้า Settings เราจะดักการกดได้ ด้วยการ subclass

UITableViewController ออกมา แล้ว override method ชื่อ

tableView:didSelectRowAtIndexPath: ซึ่งจะส่ง Argument ชื่อ indexPath ให้ระบุ row กับ section ที่ user กดเข้ามา

ปุ่ม Email Support

```
[[UIApplication sharedApplication] openURL:  
[NSURL URLWithString: @"mailto:support@google.com"]];
```

ปุ่ม Tell Friends

ใช้ UIActivityViewController

หน้า Tweet

Links ใน UITextView

UITextView สามารถแสดง link เป็นปุ่มกดได้ใน การเลือกให้ Detect Links ใน Inspector

profile_text_color, และ profile_link_color

Twitter ได้ส่งข้อมูลมาด้วยว่า user นี้ ใช้ link และ text สีอะไร, หาทางเปลี่ยน ข้อความ

TweetViewController ให้เป็นสีตามนั้น

เมื่อต้องการ Load Tweets เพิ่ม

ดู parameter ชื่อ since_id ใน

https://dev.twitter.com/docs/api/1.1/get/statuses/home_timeline

หากต้องการ Reload UIPageViewController ให้สั่ง [UIPageViewController
setViewControllers:direction:animated:completion:] ใหม่อีกครั้ง

Load Tweets เมื่อเปิด App เข้ามาใหม่

ดู applicationDidBecomeActive: ใน AppDelegate