Chapter 6
View Controllers

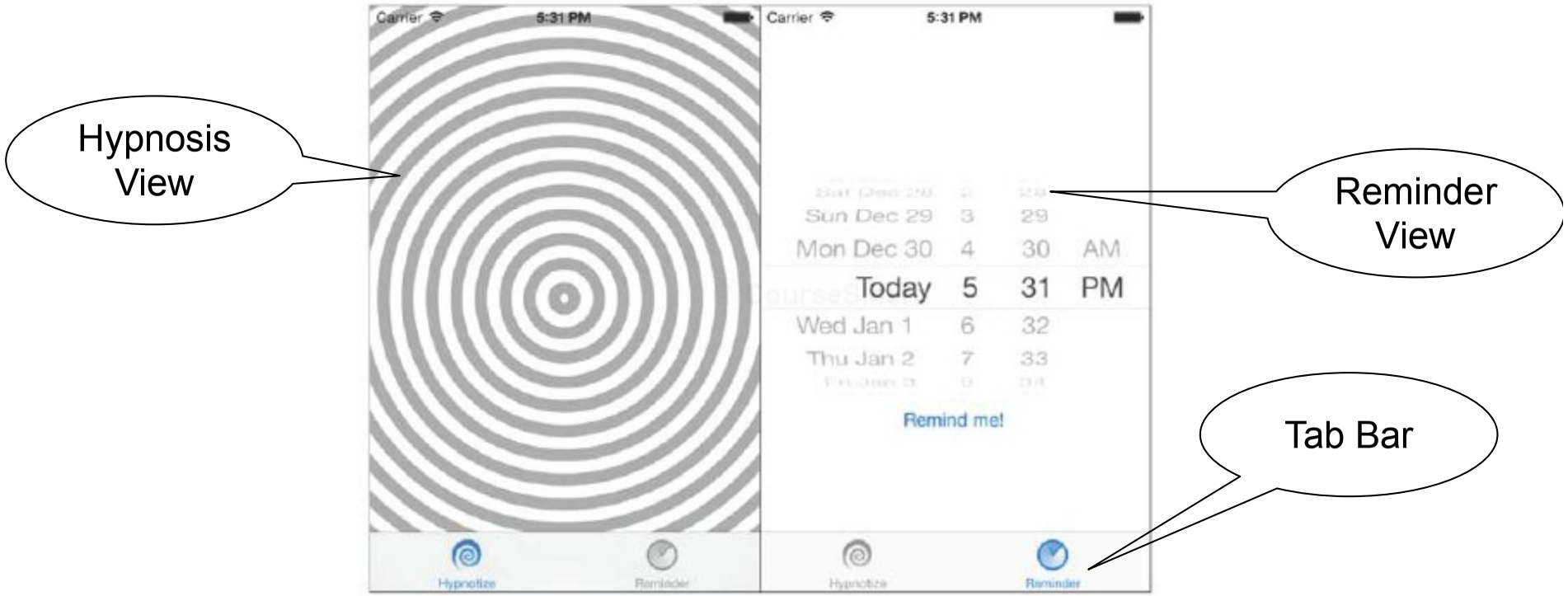
- UIViewController
- UITabBarController
- View Controller Lifecycle
- View Controller Subclasses and Templates

UIViewController

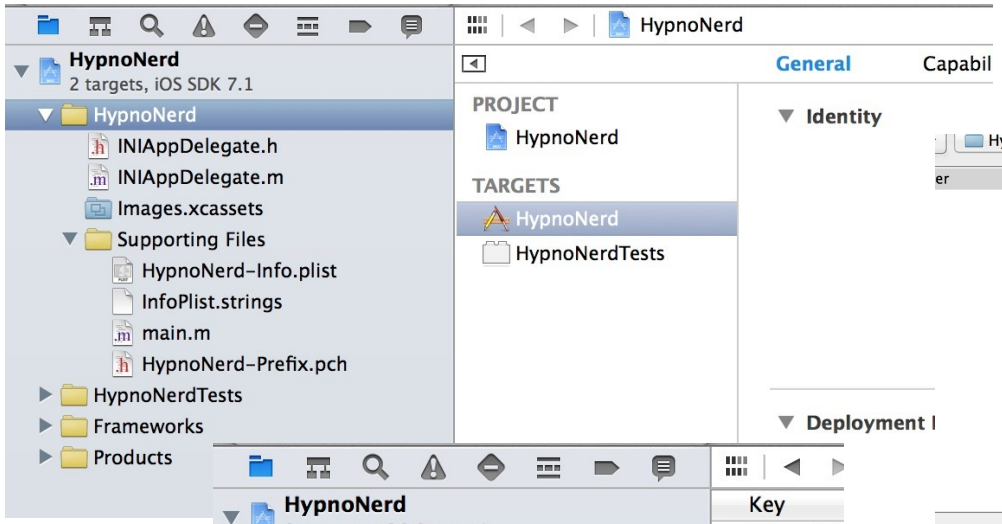
- A view controller is an instance of a subclass of UIViewController.
- It is responsible for creating view objects that make up the hierarchy, for handling events associated with the view objects in its hierarchy, and for adding its hierarchy to the window.
- An instance of UIViewController controls a single view and its hierarchy.
- Every UIViewController has a view property that points to an instance of UIView or it's subclasses and this is the screen corresponding to the view controlled by this controller (typically the full scree)
- If the main view controlled by a view controller has subviews then the controller controls the entire subviews hierarchy.
- *You never create an instance of UIViewController directly, you always subclasses it first, then create an instance of its subclass.*

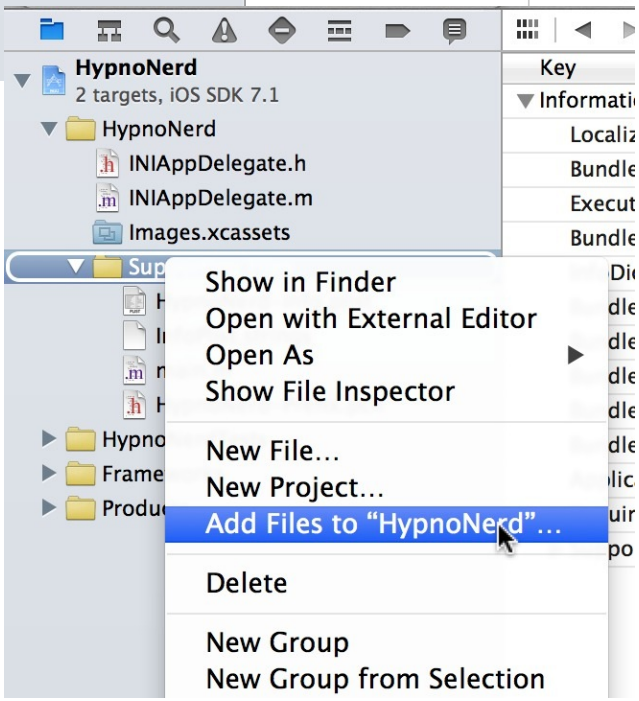
Overview of HypnoNerd

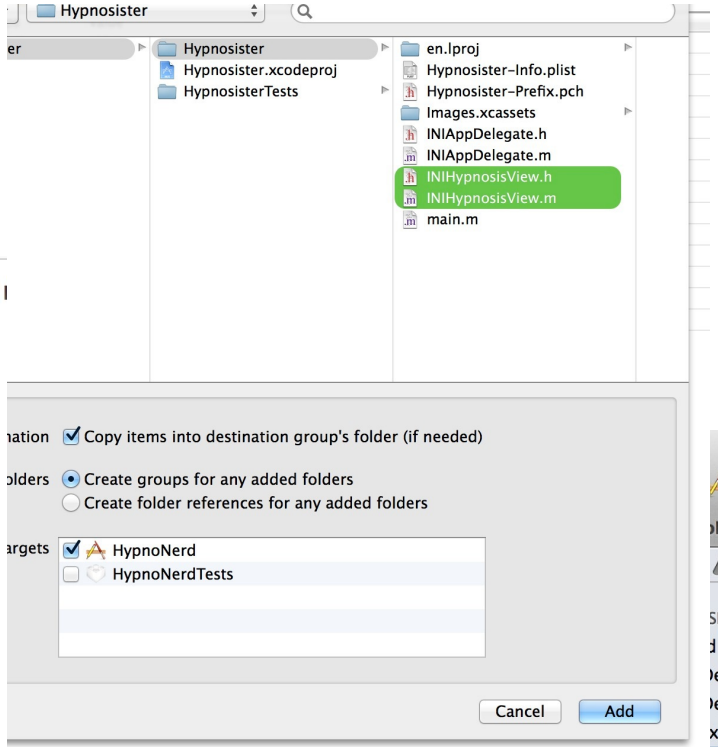
- HypnoNerd is a tabbed application that has two views:
  - HypnosisView (We create programmatically)
  - ReminderView (We create using .xib)
- We create two view controllers one for each of the above views
- We create a tab view as a subview to our main window then we add each of the above views to a tab.

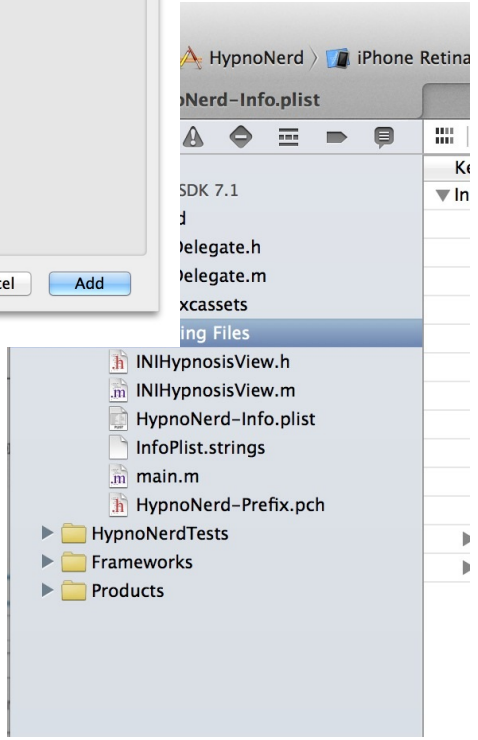


Create HypnoNerd Project and copy INIHyp…



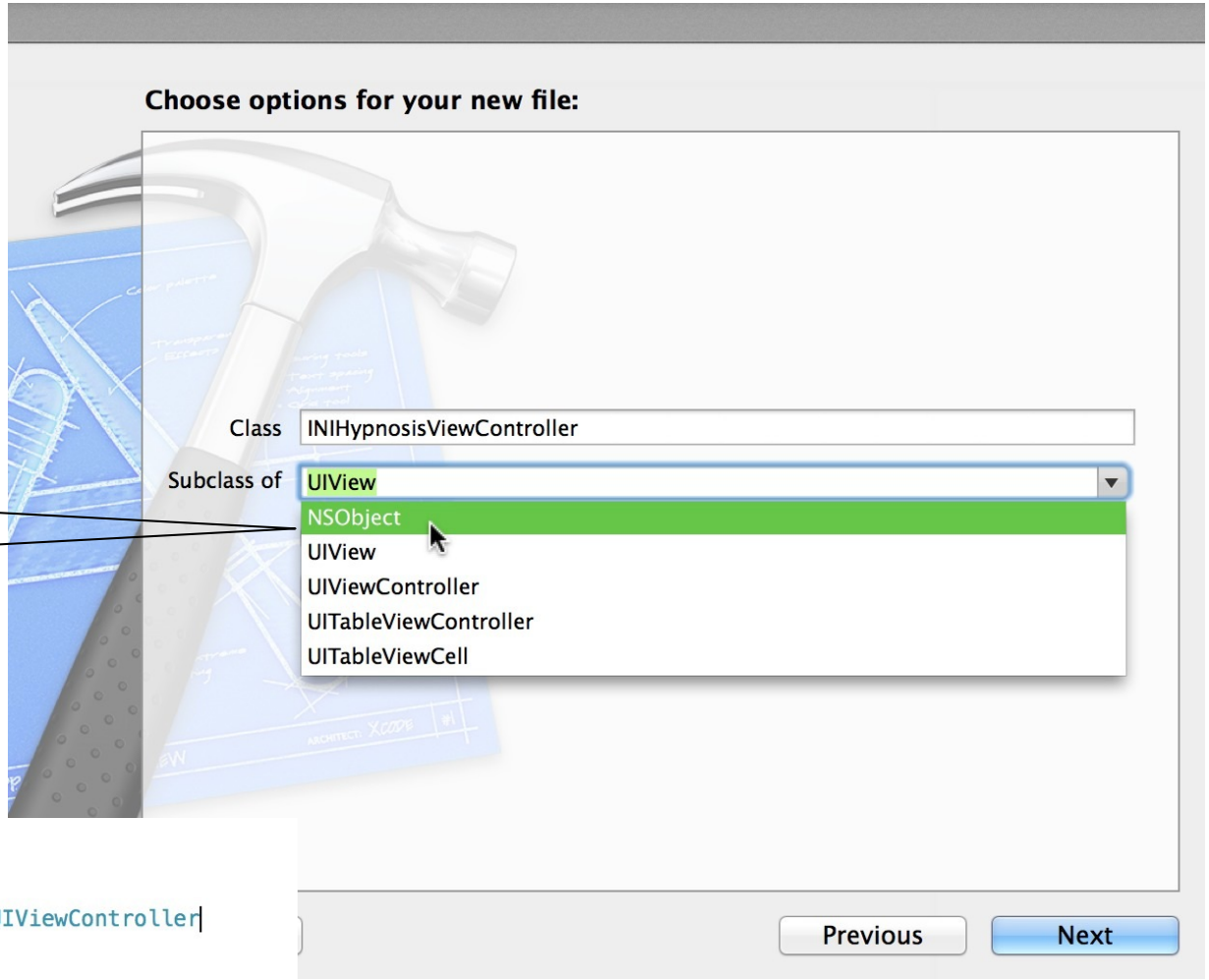






Creating INIHypnosisViewController

Create INIHypnosisViewController as an extension of NSObject.
This is BNR's choice to minimize pre-written code.
We change it later manually



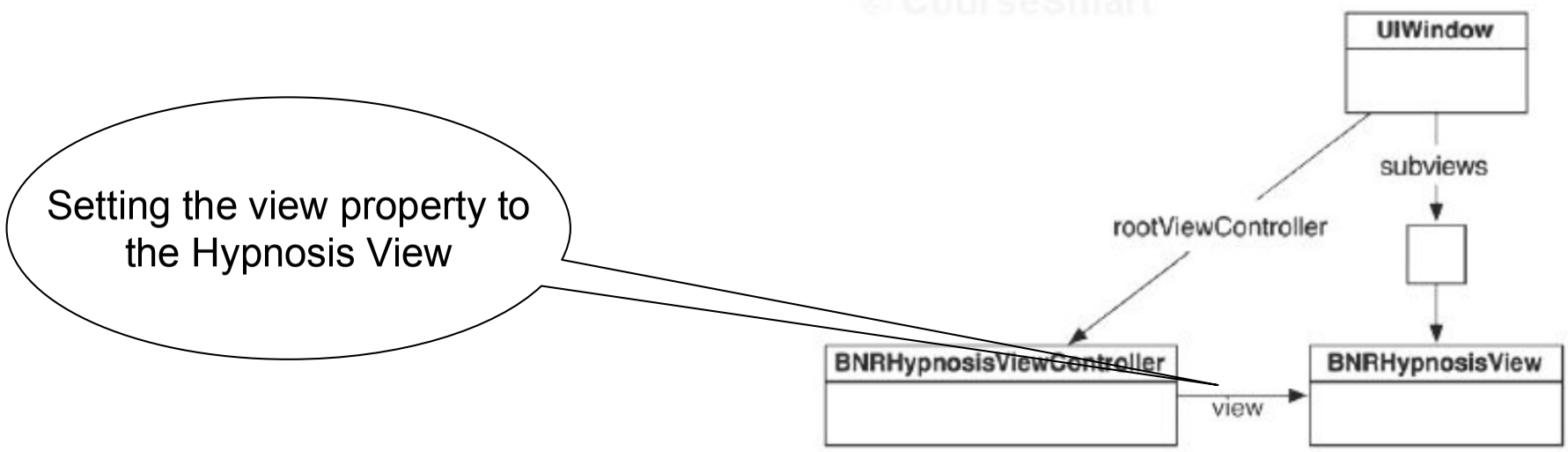

```
8
9   #import <Foundation/Foundation.h>
10
11  @interface INIHypnosisViewController : UIViewController
12
13  @end
14
```

The view of a view controller

- As a subclass of UIViewController, INIHypnosisViewController inherits an important property namely:

  @ property (nonatomic, strong) UIView * view;

- This property points to a UIView instance that is the root of the view controller's view hierarchy.
- When the view of a view controller is added as a subview of the window, the view controller's entire view hierarchy is added.



Creating the view of a view controller

- A view controller's view is not created until it needs to appear on the screen.
- This optimization is called lazy loading, and it can often conserve memory and improve performance.
- There are two ways that a view controller can create its view hierarchy:

  Programmatically, by overriding the UIViewController method loadView.

  Using Interface Builder, by loading a NIB file.
- Because the view hierarchy of INIHypnosisViewController consists of only one view, it is a good candidate for being created programmatically.

Programmatically creating the view of a view controller

- In our App we create two subclasses of UIViewController, INIHypnosisViewController and INIReminderViewController
- The view controlled by INIHypnosisViewController will be created and loaded programmatically.
- This is done by overriding the loadView method in the controller
- When a view controller is created, its view property is nil. If a view controller is asked for its view and its view is nil, then the view controller is sent the loadView message.

```
9  #import "INIHypnosisViewController.h"
10 #import "INIHypnosisView.h"
11
12 @implementation INIHypnosisViewController
13
14 - ( void) loadView
15 {
16     // Create a view
17     CGRect frame = [UIScreen mainScreen]. bounds;
18     INIHypnosisView *backgroundView = [[INIHypnosisView alloc] initWithFrame: frame];
19     // Set it as "the" view of this view controller
20     self.view = backgroundView;
21 }
22
23 @end
```

loadView in the implementation INIHypnosisViewControlle

Setting the root view controller

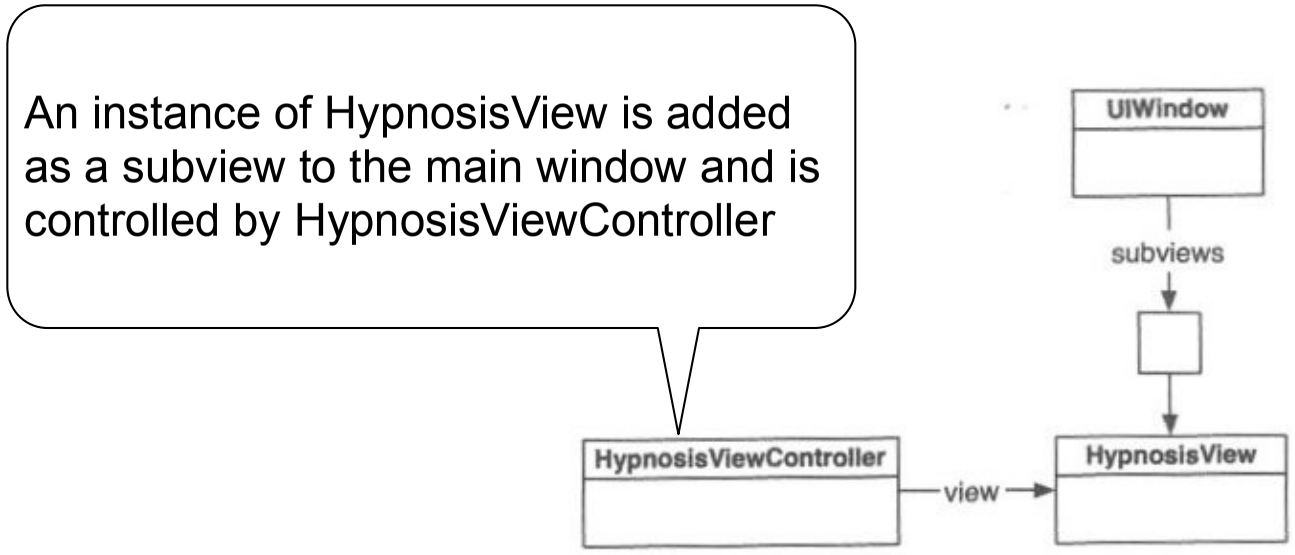
- The next step is to add the view hierarchy of the INIHypnosisViewController to the application window so that it will appear on screen to users.
- This is done in INIAppDelegate.m

```
#import "INIAppDelegate.h"
#import "INIHypnosisViewController.h"

@implementation INIAppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.

    INIHypnosisViewController *hvc = [[INIHypnosisViewController alloc] init];
    self. window.rootViewController = hvc;

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}


- (void)setRootViewController:(UIViewController *)viewController
{
    // Get the view of the root view controller
    UIView *rootView = viewController.view;

    // Make a frame that fits the window's bounds
    CGRect viewFrame = self.bounds;
    rootView.frame = viewFrame;

    // Insert this view as window's subview
    [self addSubview:rootView];

    // Update the instance variable
    _rootViewController = viewController;

}
```

This calls the setter to the left

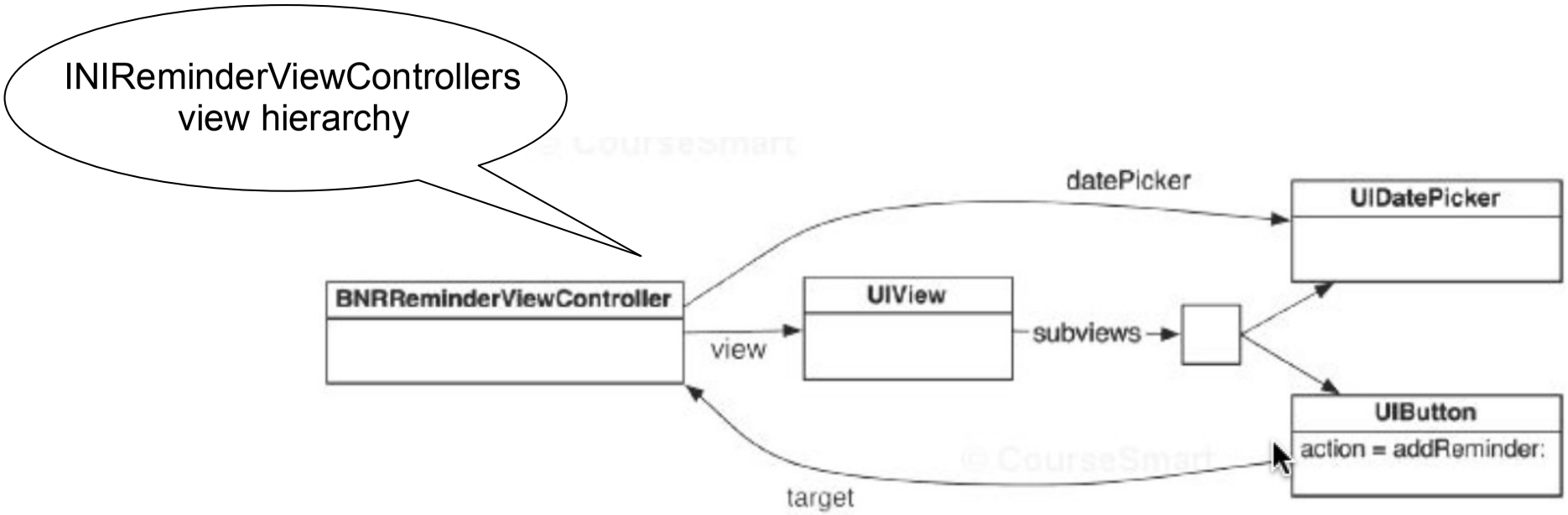
HypnoNerd with one view only

- The first view we add to HypnoNerd is the INIHypnosisView
- We are using a view controller to present the INIHypnosisView instead of adding the view object itself to the window.
- This adds a layer of complexity, which, as we will see later, gives us power and flexibility to do neat things.



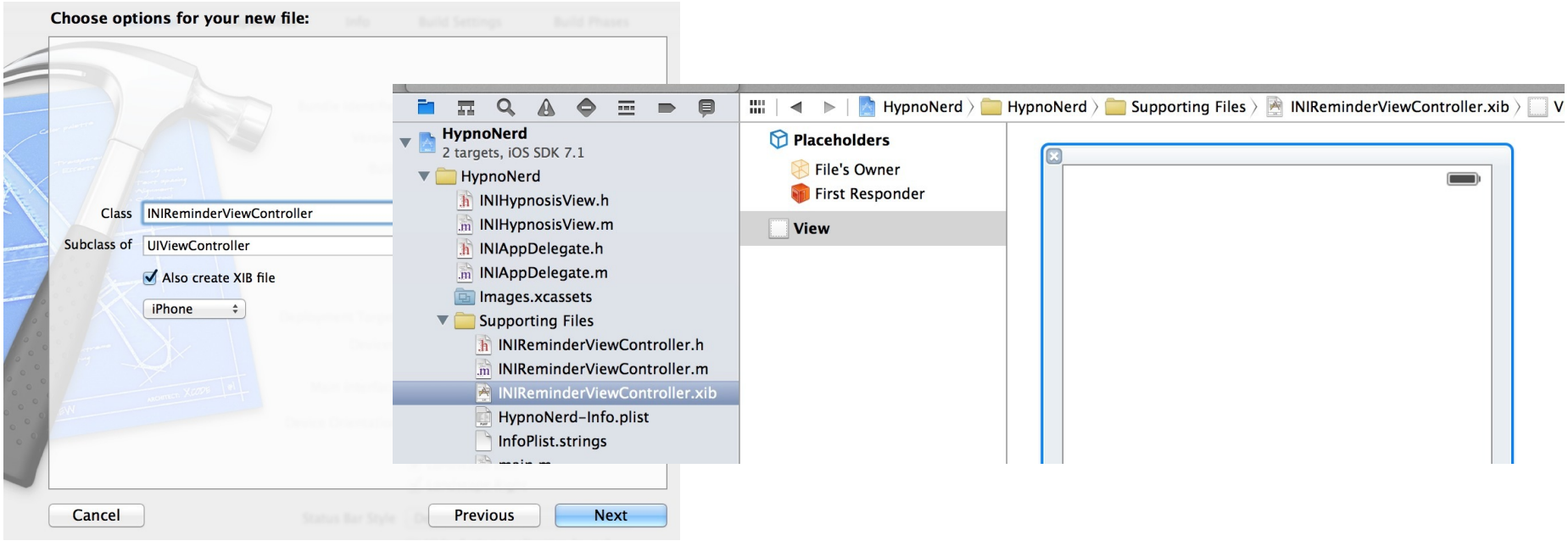
Creating the second view

- The second view is INIReminderViewController class.
- This view controller will enable us to pick a date and time to receive a reminder to be hypnotized.
- This reminder will take the form of a notification that will appear even if HypnoNerd is not running at the time.



Creating a view in Interface Builder

- Create the INIReminderViewController as a subclass of UIViewController
- Use the xib editor to create the Hierarchy of the reminder view.
- The view controller will have a datePicker property that points to the UIDatePicker object.
- The view controller will be the target of the UIButton and must implement its action method addReminder:.

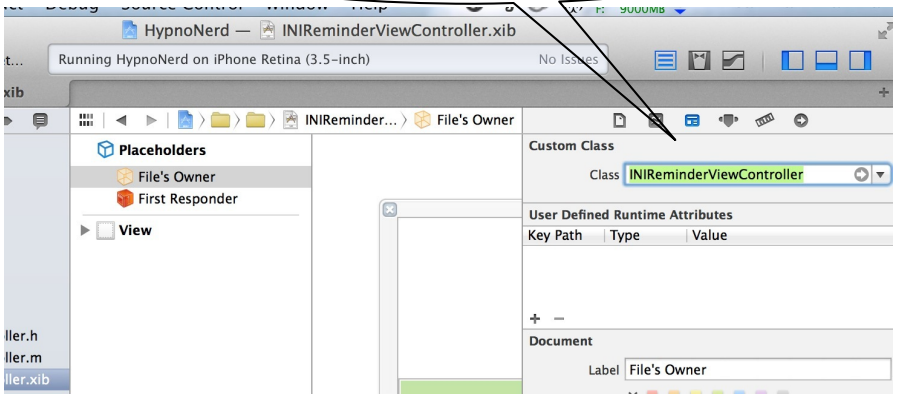


Creating the second view (cont.)

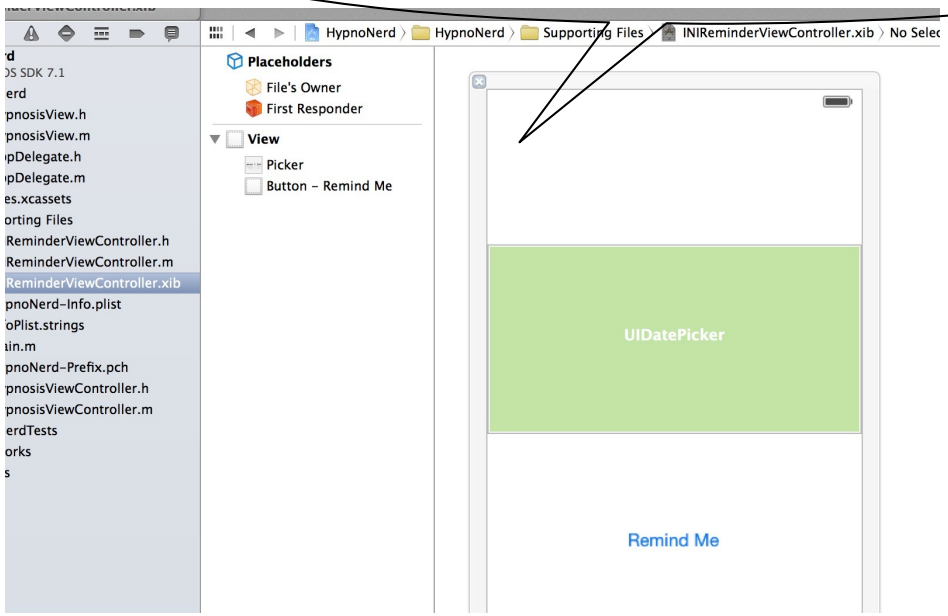
1) INIReminderViewController with @property and IBAction

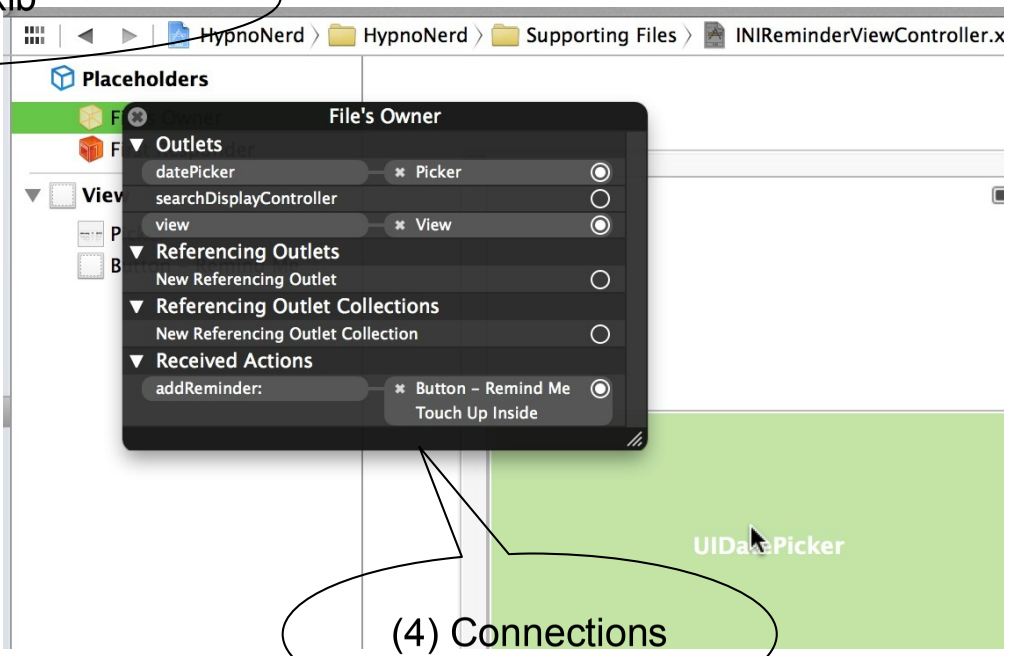
```
@interface INIReminderViewController ()

@property (nonatomic, weak) IBOutlet UIDatePicker *datePicker;

@end

@implementation INIReminderViewController

- (IBAction) addReminder:(id) sender
{
    NSDate *date = self.datePicker.date;
    NSLog(@" Setting a reminder for %@", date);
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
```

(3) Check the File's Owner Custom Class



(2) INIReminderViewController.xib





(4) Connections

Demystifying the NIB: File's Owner

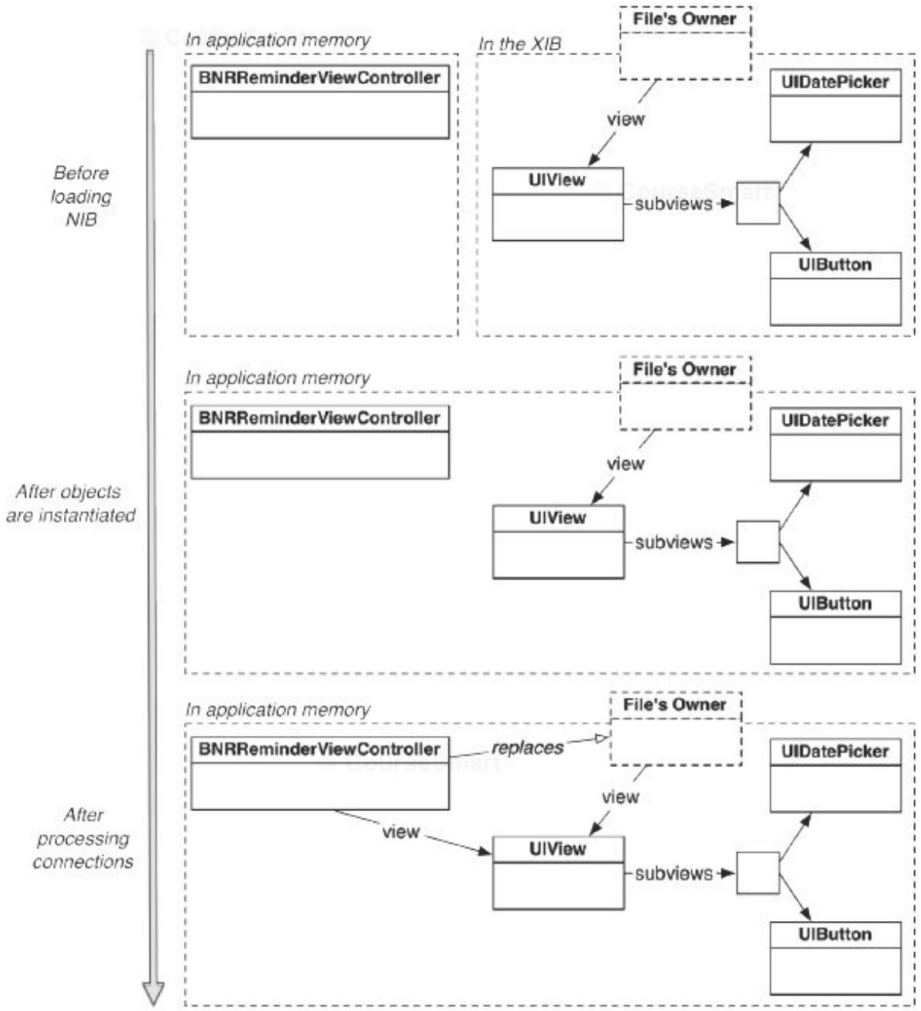
When a XIB file created from an empty XIB template, it has no idea who is going to be the owner (controller) of the view that will be generated from the archived view it contains.

This will remain as an empty place holder inside the XIB file.

When the controller loads its view from the XIB file with the same name, it needs to drop itself in this empty place holder.

To tell the XIB file what goes in the place holder:

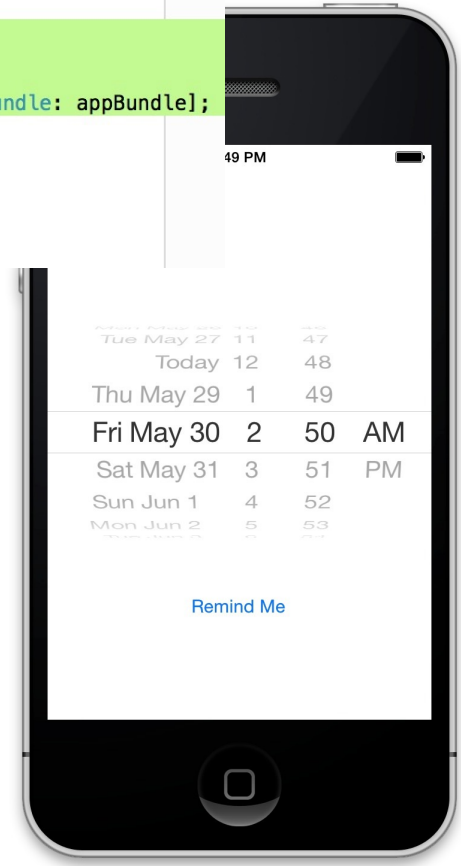
- Use the Identity Inspector to set the class of the File's Owner to the controller class
- Connect the view of the view of view you created in your XIB file



HypnoNerd with reminder view only

```objc
#import "INIHypnosisViewController.h"
#import "INIReminderViewController.h"

@implementation INIAppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.

//    INIHypnosisViewController *hvc = [[INIHypnosisViewController alloc] init];
//    self. window.rootViewController = hvc;

    // This will get a pointer to an object that represents the app bundle
    NSBundle *appBundle = [ NSBundle mainBundle];
    // Look in the appBundle for the file INIReminderViewController.xib
    INIReminderViewController *rvc = [[INIReminderViewController alloc] initWithNibName:@"INIReminderViewController" bundle: appBundle];
    self. window. rootViewController = rvc;

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

```
erd
2014-05-28 13:50:04.016 HypnoNerd[10579:60b]  Setting a reminder for 2014-05-30 06:50:18 +0000
```



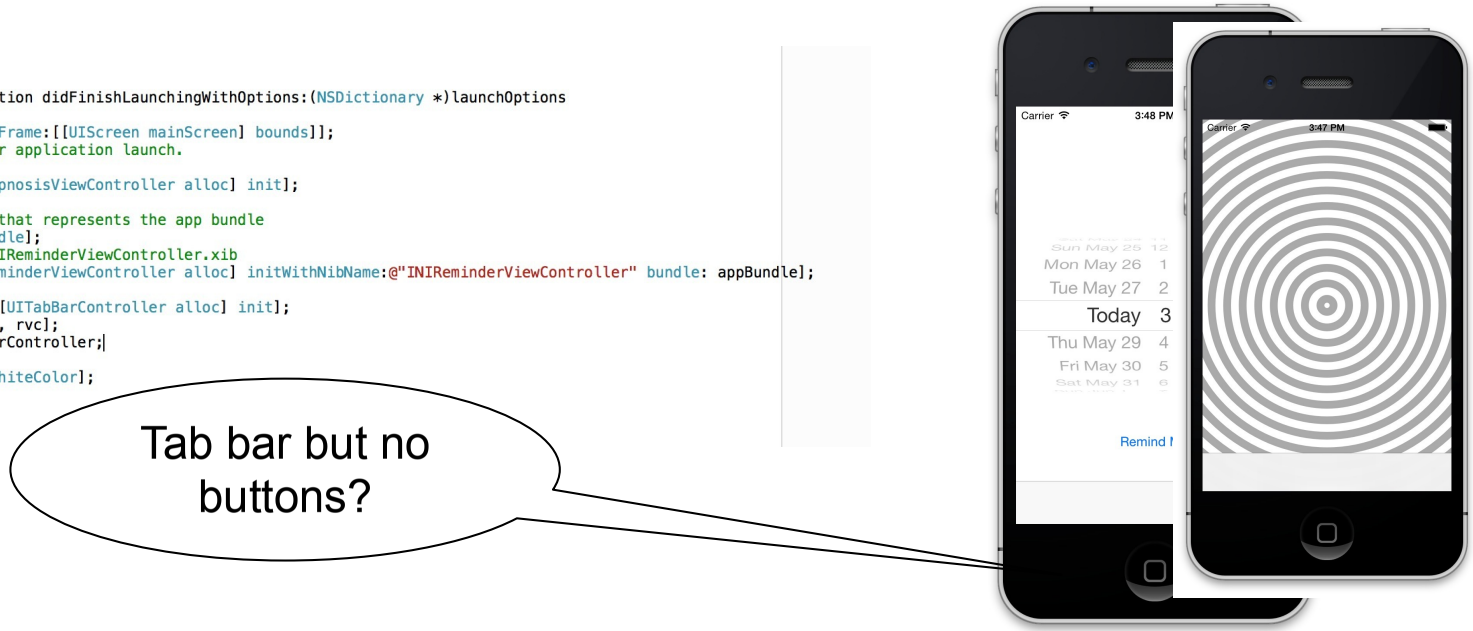
HypnoNerd with the two views

- There are three additional changes we need to make for our app:
  - Create and instance of UITabBarController and set it to be the root view controller
  - Create an instance of NSArray to hold a pointer to each of our INIReminderViewController and INIHypnosisViewController
  - Modify the INIReminderViewController and INIHypnosisViewController so that they know their own tab label string and their tab icon image

UITabBarController

- UITabBarController that will allow us to swap between instances of INIHypnosisViewController and INIReminderViewController.
- UITabBarController keeps an array of view controllers.
- It also maintains a tab bar at the bottom of the screen with a tab for each view controller in this array.
- Tapping on a tab results in the presentation of the view of the view controller associated with that tab.
- We will construct it programmatically by modifying INIAppDelegate.m

```
@implementation INIAppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.

    INIHypnosisViewController *hvc = [[INIHypnosisViewController alloc] init];

    // This will get a pointer to an object that represents the app bundle
    NSBundle *appBundle = [ NSBundle mainBundle];
    // Look in the appBundle for the file INIReminderViewController.xib
    INIReminderViewController *rvc = [[INIReminderViewController alloc] initWithNibName:@"INIReminderViewController" bundle: appBundle];

    UITabBarController *tabBarController = [[UITabBarController alloc] init];
    tabBarController.viewControllers = @[hvc, rvc];
    self. window. rootViewController = tabBarController;

    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
    return YES;
}
```

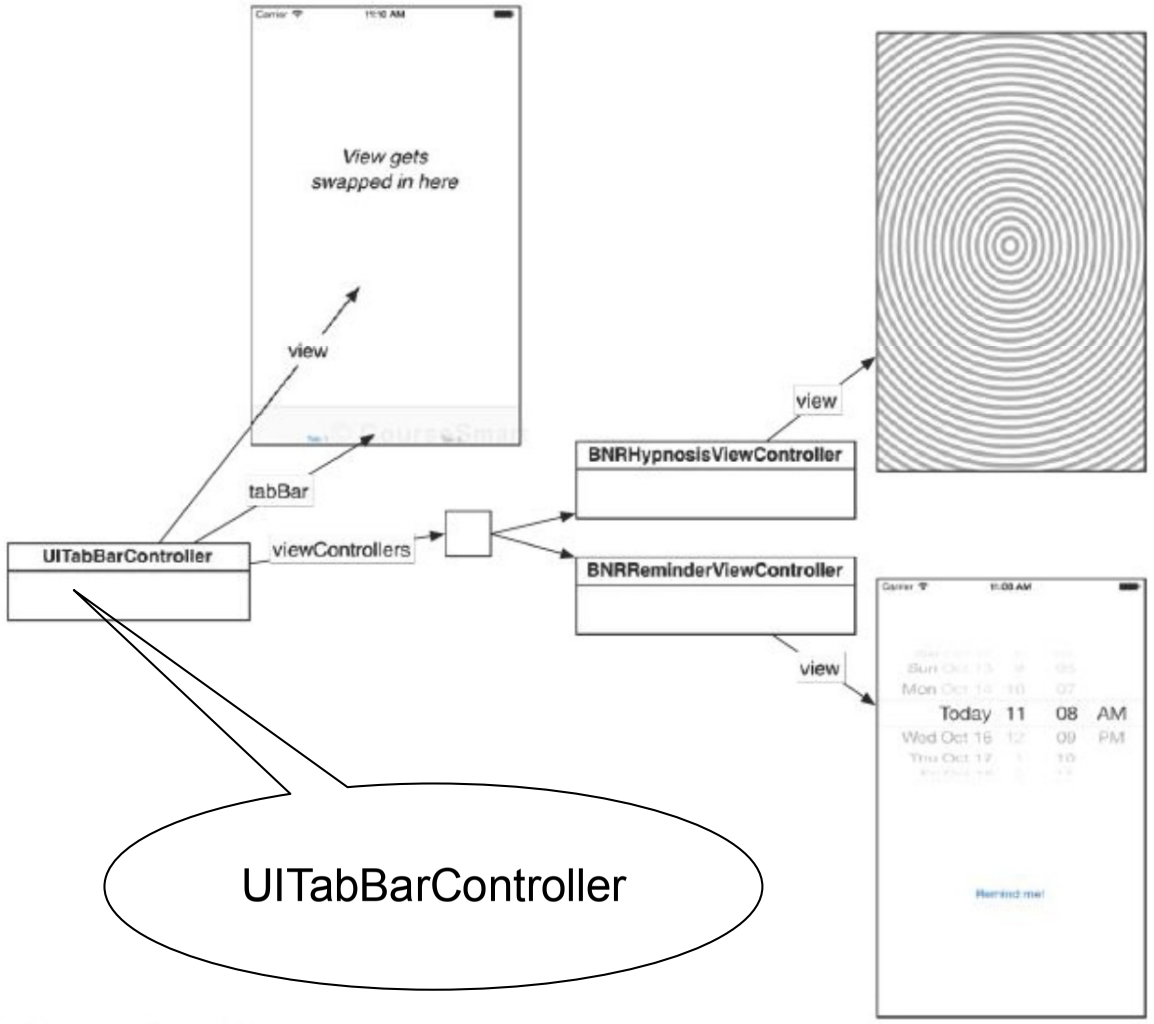


Anatomy of a Tab Bar Controller

UITabBarController is itself a subclass of UIViewController.

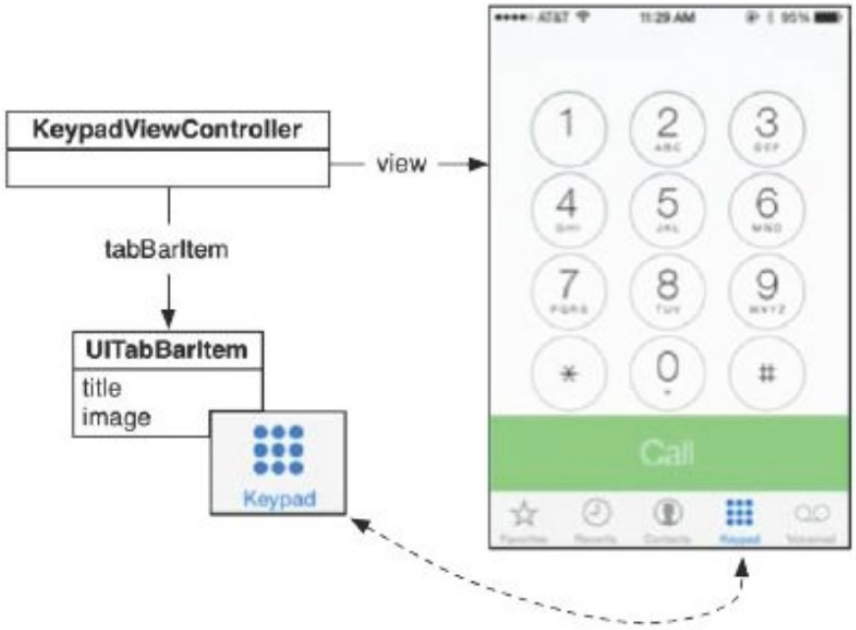
A UITabBarController's view is a UIView with two subviews:

- The tab bar
- The view of the selected view controller



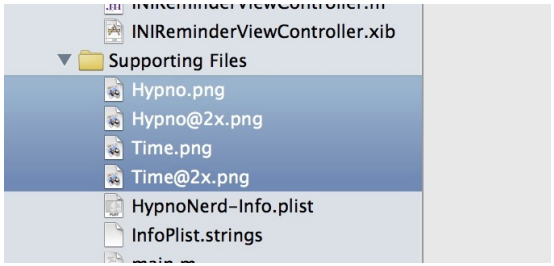
Tab bar items

- Each tab on the tab bar can display a title and an image.
- Each view controller maintains a tabBarItem property for this purpose.
- When a view controller is contained by a UITabBarController, its tab bar item appears in the tab bar.



Adding Tab Bar Items

```objc
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Set the tab bar item's title
        self.tabBarItem.title = @"Hypnotize";
        // Create a UIImage from a file
        // This will use Hypno@ 2x.png on retina display devices
        UIImage *image = [UIImage imageNamed:@"Hypno.png"];
        // Put that image on the tab bar item
        self.tabBarItem.image = image;
    }
    return self;
}
```




```objc
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Set the tab bar item's title
        self.tabBarItem.title = @"Reminder";
        // Create a UIImage from a file
        // This will use Time@2x.png on retina display devices
        UIImage *image = [UIImage imageNamed:@"Time.png"];
        // Put that image on the tab bar item
        self.tabBarItem.image = image;
    }
    return self;
}
```

Adding a Local Notification

```
17  @implementation INIReminderViewController
18
19  - (IBAction) addReminder:(id) sender
20  {
21      NSDate *date = self.datePicker.date;
22      NSLog(@" Setting a reminder for %@", date);
23
24      UILocalNotification *note = [[UILocalNotification alloc] init];
25      note.alertBody = @"Hypnotize me!";
26      note.fireDate = date;
27      [[UIApplication sharedApplication] scheduleLocalNotification: note];
28  }
```

Initializing the view controller

- initWithNibName:bundle: is the designated initializer for UIViewController
- Using self = [super initWithNibName:nil bundle:nil]; means use the xib file with the same name as the controller and the default app bundle created for this target
- You only need to override this method if you need additional initialization such as the ones we did here namely specifying the tab image or label
- There is no mention of the view in the initializer at all

Loaded and Appearing Views (Lazy Loading)

- The view controller will create or load the view only when it is about to be displayed on the screen.
- This is intended to preserver memory usage.
- The initializer gets called only once.
- Use viewDidLoad to update the view if you need to and not the initializer.
- Never access the view property of the view controller in the initializer so that you would not defeat this measure

```
37
38  - (void)viewDidLoad
39  {
40      [super viewDidLoad];
41      // Do any additional setup after loading the view from its nib.
42      NSLog(@" INIHypnosisViewController loaded its view.");
43  }
44
```

```
44
45  - (void)viewDidLoad
46  {
47      [super viewDidLoad];
48      // Do any additional setup after loading the view from its nib.
49      NSLog(@" INIReminderViewController loaded its view.");
50  }
51
```


```
2014-05-29 12:38:27.150 HypnoNerd[13004:60b]  INIHypnosisViewController loaded its view.
```

Accessing subviews

- Lazy loading necessitates special consideration in accessing or customizing subviews (Their pointers will be nil until loaded).
- There are two main options to accessing subviews:

  You should override viewDidLoad if the configuration/customization only needs to be done once during the run of the app.

  You override viewWillAppear: if you need the configuration/ customization to be done and redone every time the view controller appears on screen.

  Example: You are going to configure the date picker to only allow users to select a time that is at least 60 seconds in the future. This