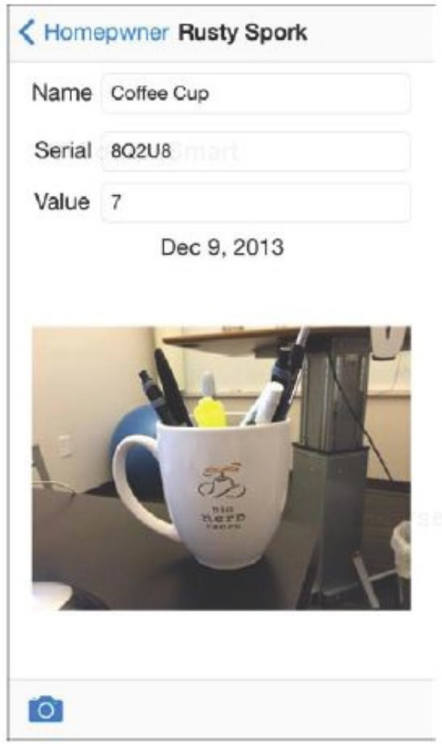
Chapter 11
Camera

- Homeowner with camera addition
- Displaying Images and UIImageView
- UIImagePickerController

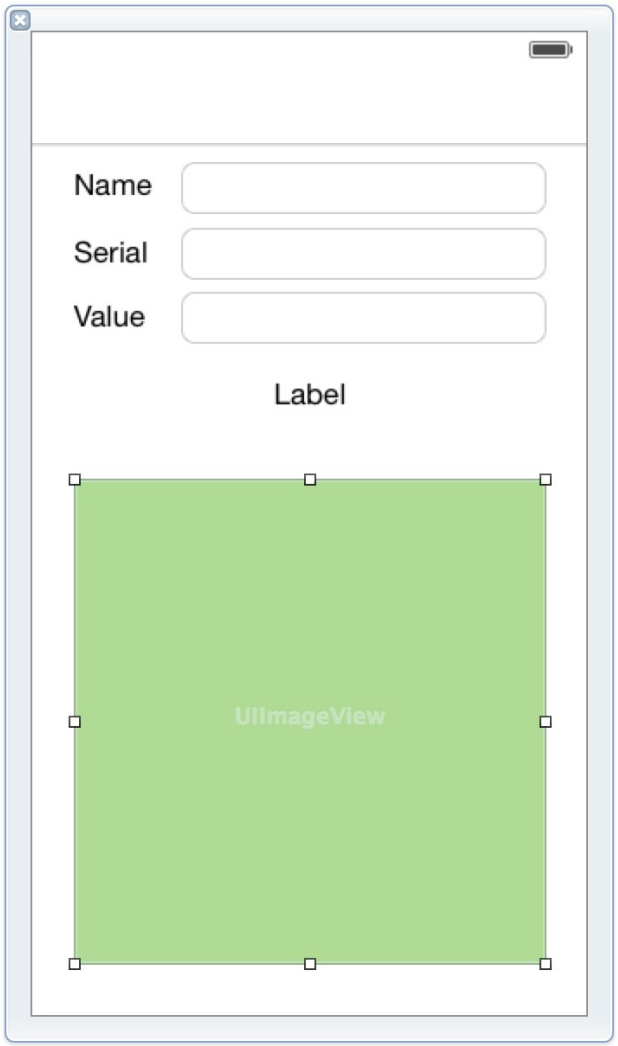
Homeowner with camera addition

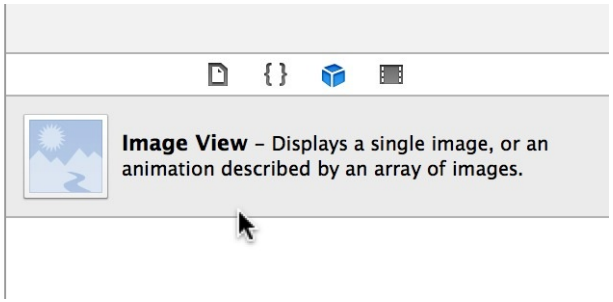
- Images tend to be very large, so it is a good idea to store images separately from other data.
- We are going to create a second store for images we will call INIImageStore
- We will fetch and cache images as they are needed.
- We will also be able to flush the cache when memory runs low.

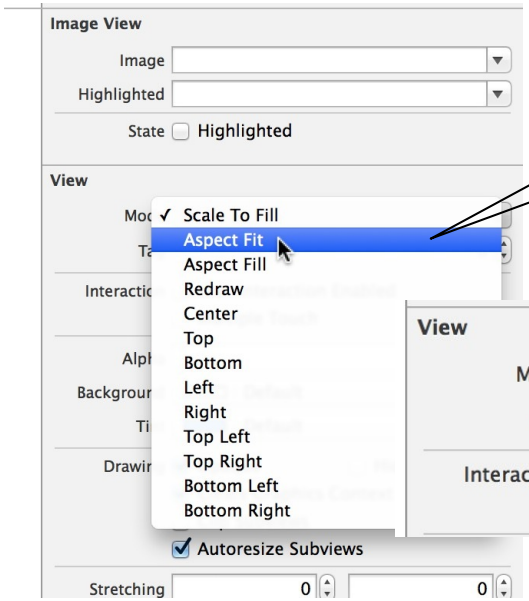


Displaying Image and UIImageView

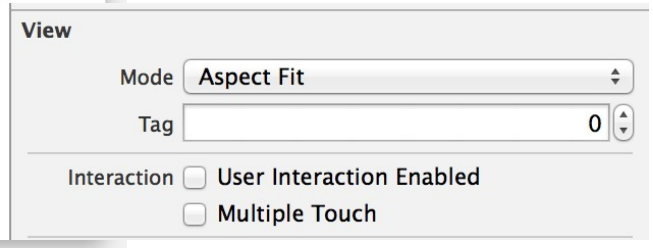
- Add an instance of UIImageView to INIDetailedViewController.xib







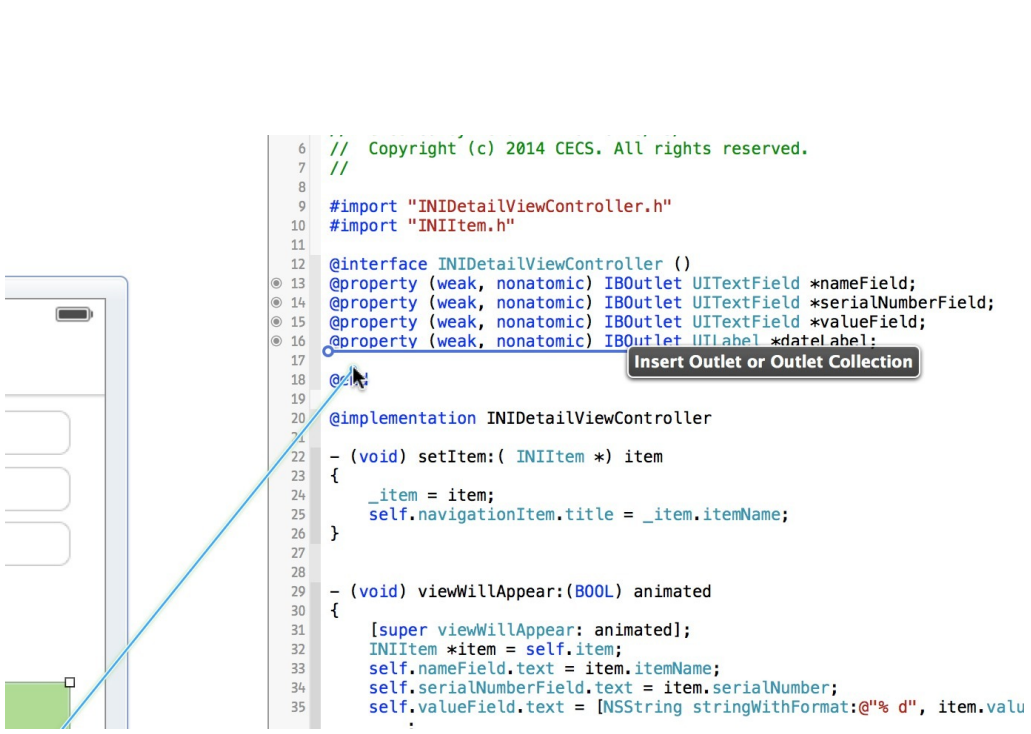
Change the contentMode property of UIImageView

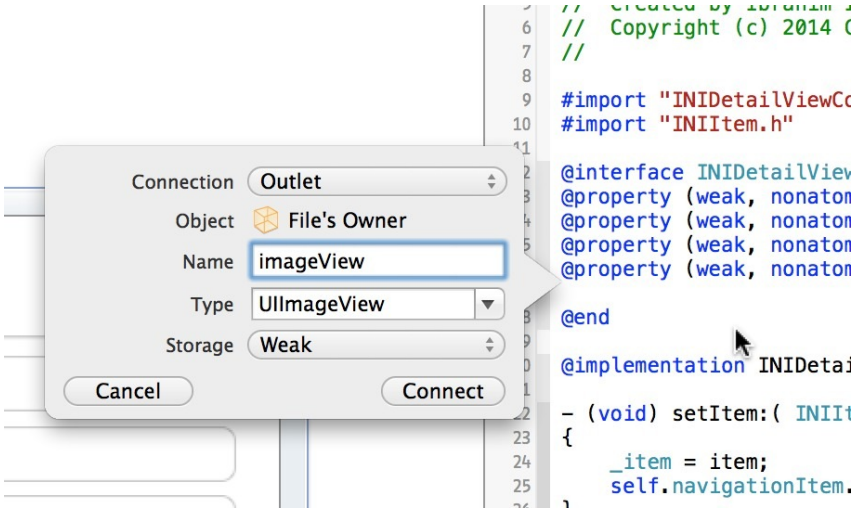


Create UIImageView Outlet

- Use the Assistant editor to create a weak IBOutlet of type UIImageView and call it imageView
- This will create

  @property (weak, nonatomic) IBOutlet UIImageView *imageView;

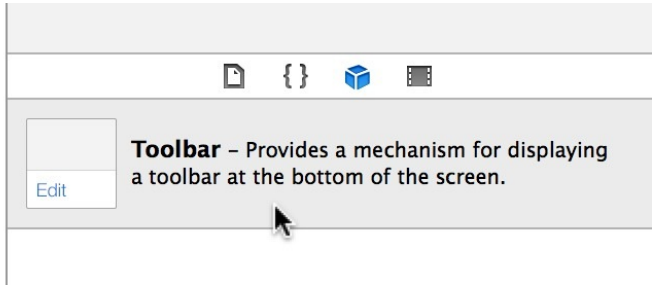


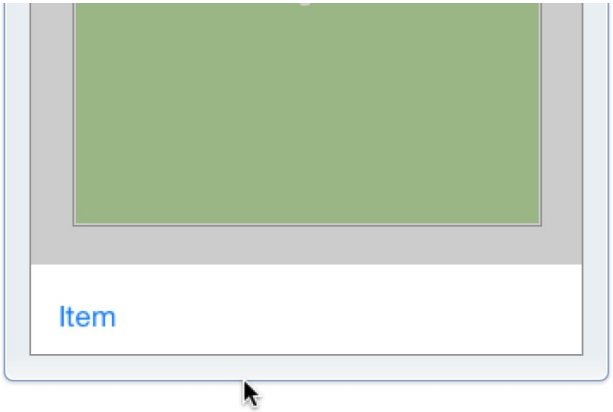



```
// Copyright (c) 2014 CECS. All rights reserved.
//

#import "INIDetailViewController.h"
#import "INIItem.h"

@interface INIDetailViewController ()
@property (weak, nonatomic) IBOutlet UITextField *nameField;
@property (weak, nonatomic) IBOutlet UITextField *serialNumberField;
@property (weak, nonatomic) IBOutlet UITextField *valueField;
@property (weak, nonatomic) IBOutlet UILabel *dateLabel;
@property (weak, nonatomic) IBOutlet UIImageView *imageView;

@end
```

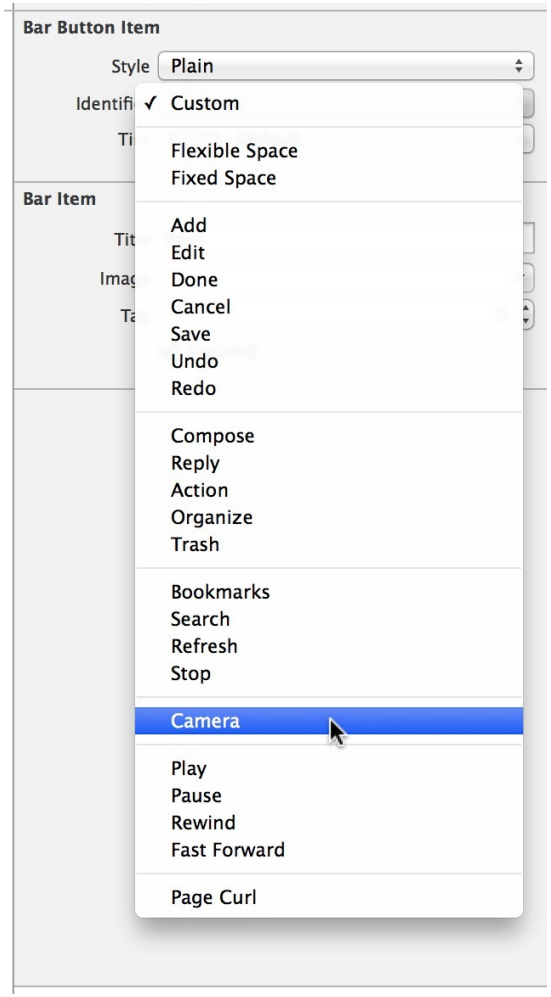
Adding a camera button

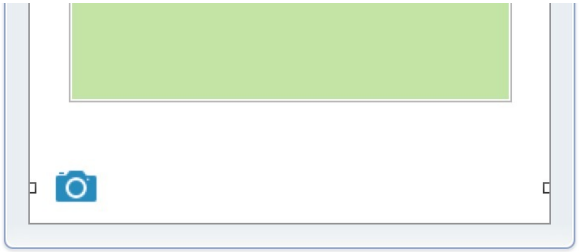
- Now you need a button to initiate the photo- taking process
- Since we are already using the navigation bar, we will create an instance of UIToolbar and place it at the bottom of INIDetailViewController's view.
- A UIToolbar works a lot like a UINavigationBar – you can add instances of UIBarButtonItem to it.
- A toolbar has an array of bar button items. You can place as many bar button items in a toolbar as can fit on the screen.





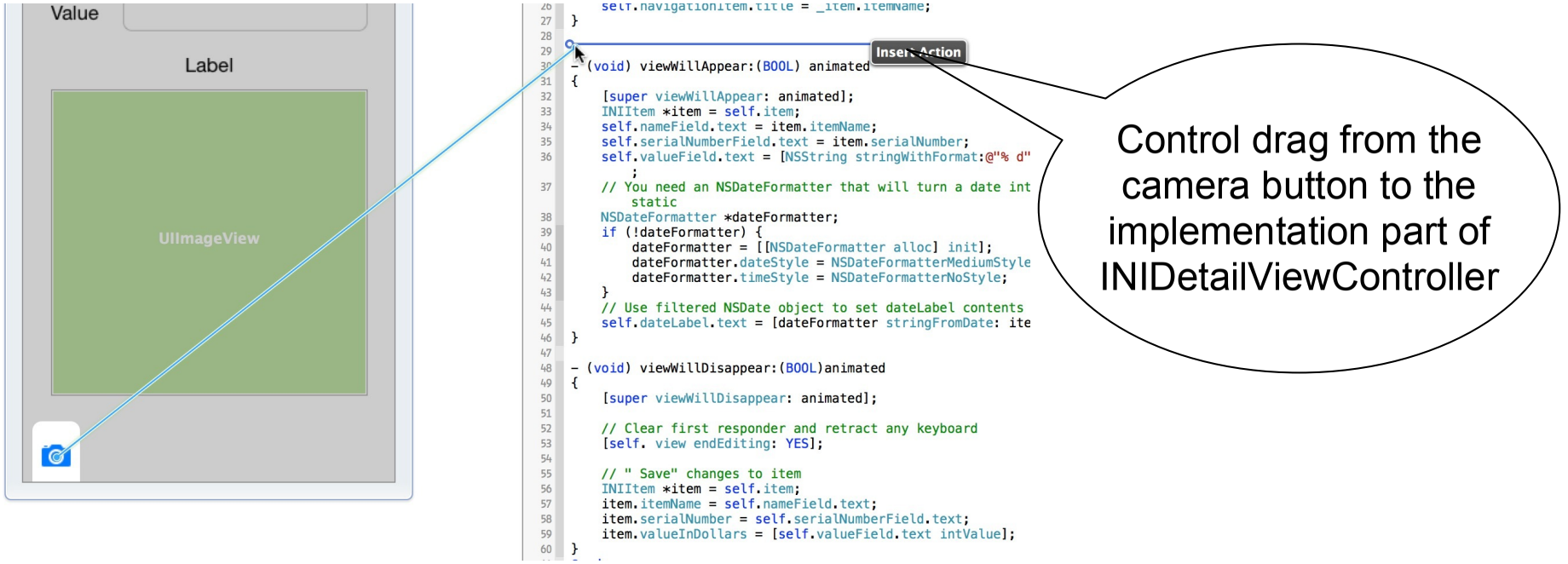
Select Bar Button Item Style

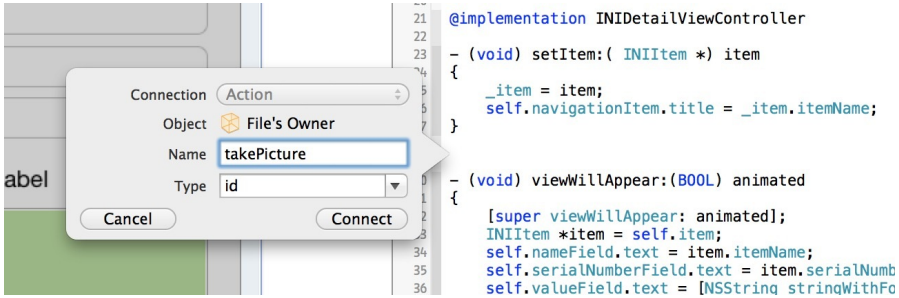


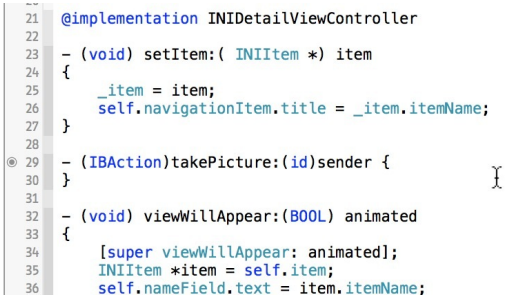


Taking Pictures and UIImagePickerController

- Connect the camera button to a IBAction called takePicture

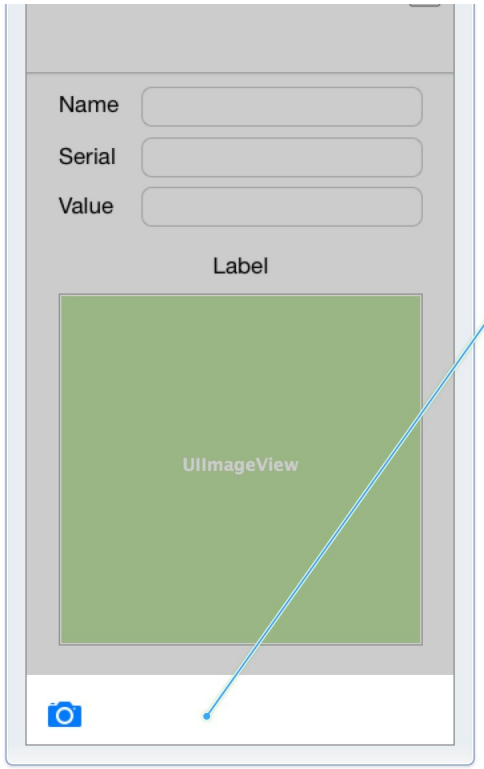


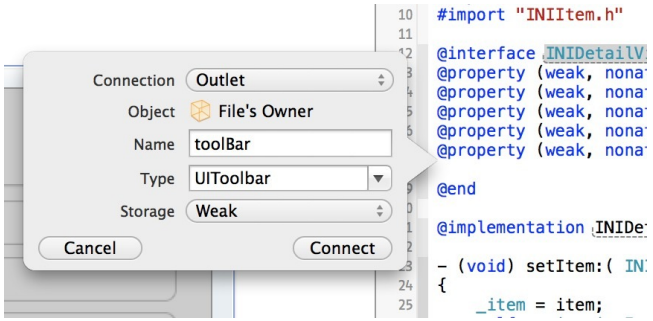


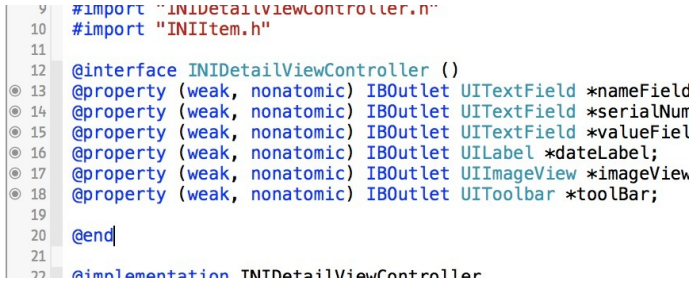


Setting a pointer to the UIToolbar

- Later on we will need a pointer to the tool bar so that we can reference it.
- We will do that now by control dragging







Picking the source of the picture

- To be able to take a picture we need an instance of UIImagePickerController.
- UIImagePickerController allow for one of three sources to pick a picture:

  Take it with the camera,
  UIImagePickerControllerSourceTypeCamera

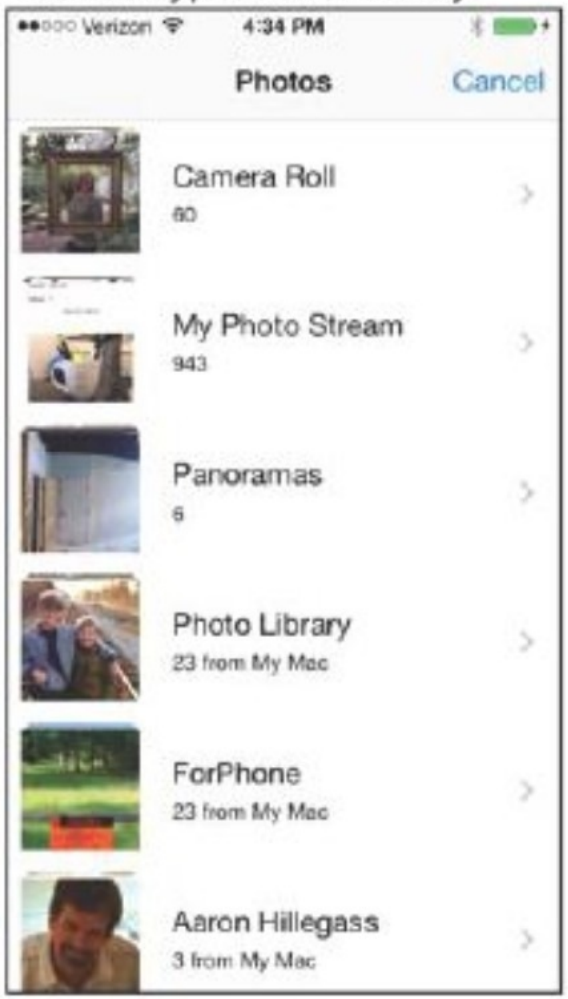
  Select a picture from an album from the Photo Library
  UIImagePickerControllerSourceTypePhotoLibrary

  Select a picture from t the Saved Photos Album
  UIImagePickerControllerSourceTypeSavedPhotosAlbum
- We also need to appoint a delegate for UIImagePickerController by conforming to UIImagePickerControllerDelegate

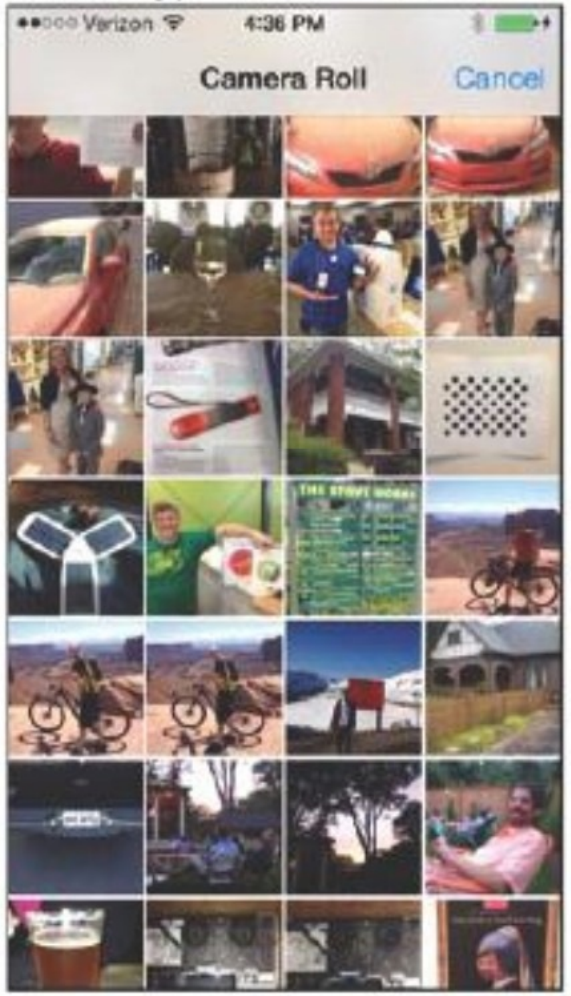
Examples of three source types

SourceTypeCamera

SourceTypePhotoLibrary



SourceTypeSavedPhotosAlbum



Implement takePicture:

- Implement takePicture:

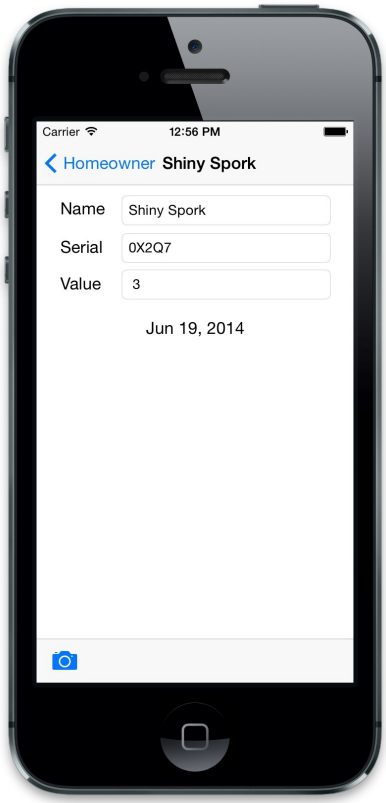
  Test if the device has a camera by sending the message isSourceTypeAvailable: to the UIImagePickerController class:

  Then set the source to be the camera or the photo library accordingly

```
29
30  - (IBAction)takePicture:(id)sender {
31
32      UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
33      // If the device has a camera, take a picture, otherwise,
34      // just pick from photo library
35      if ([UIImagePickerController isSourceTypeAvailable: UIImagePickerControllerSourceTypeCamera]) {
36          imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
37      } else {
38          imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
39      }
40  }
41
```

Testing if I have a camera

```
29
30  - (IBAction)takePicture:(id)sender {
31
32      UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
33      // If the device has a camera, take a picture, otherwise,
34      // just pick from photo library
35      if ([UIImagePickerController isSourceTypeAvailable: UIImagePickerControllerSourceTypeCamera]) {
36          NSLog(@"I have a camera");
37          imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
38      } else {
39          NSLog(@"I do not have a camera");
40          imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
41      }
42  }
43
```




```
tem.valueInDollars = [self.valueField.text intValue];
```

```
2014-06-19 12:52:34.269 Homepwner[1290:60b] I do not have a camera
```

Setting the image picker's delegate

- In addition to a source type, the UIImagePickerController instance needs a delegate.
- When the user selects an image from the UIImagePickerController's interface, the delegate is sent the message

  imagePickerController: didFinishPickingMediaWithInfo:.
- If the user taps the cancel button, then the delegate receives the message

  imagePickerControllerDidCancel:
- The image picker's delegate will be the instance of INIDetailViewController.
- Declare that INIDetailViewController conforms to

  UINavigationControllerDelegate protocol and

  UIImagePickerControllerDelegate protocol
- Why UINavigationControllerDelegate? UIImagePickerController's delegate property is actually inherited from its superclass, UINavigationController, and while UIImagePickerController has its own delegate protocol, its inherited delegate property is declared to point to an object that conforms to UINavigationControllerDelegate.

```
    imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
}
imagePicker.delegate = self;
```

⚠ Assigning to 'id<UINavigationControllerDelegate,UIImagePickerControllerDelegate>' from incompatible type 'INIDetailViewController *c

Setting delegates

UIImagePickerController is a subclass of UINavigationController. Thus we need both delegates.

```
10  #import "INIItem.h"
11
12  @interface INIDetailViewController () < UINavigationControllerDelegate, UIImagePickerControllerDelegate>
13
14  @property (weak, nonatomic) IBOutlet UITextField *nameField;
```

```
41          imagePicker.sourceType = UIImagePi
42      }
43      imagePicker.delegate = self;
44  }
45
```

The warning is gone

```
2014-06-20 11:13:07.622 Homepwner[2338:60b] I do not have a camera
```

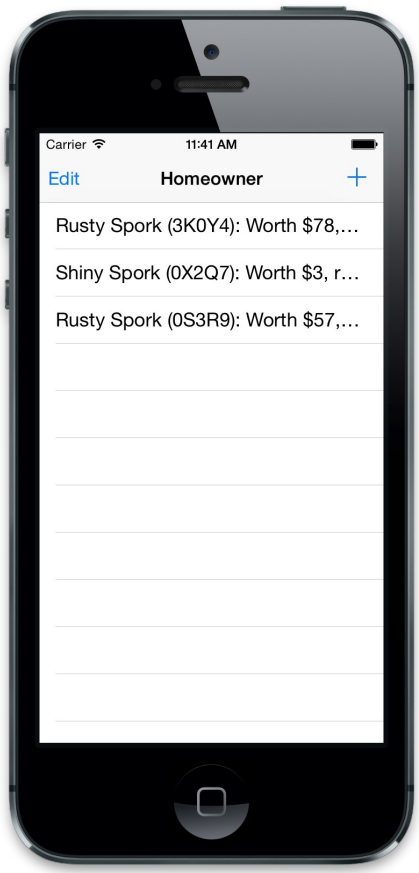
Presenting the image picker modally

- Unlike other UIViewControllers subclasses, UIImagePickerController instances are presented on the screen modally.
- A modal view controller takes over the screen until it finishes is's work and it gets dismissed.
- To present a view controller modally, you send the message ***presentViewController: animated: completion:*** to the UIViewController whose view is on the screen.
- The view controller to be presented is passed to it, and this view controller's view slides up from the bottom of the screen.
- In our application we ask the INIDetailViewController to present modally the imagePicker which is an instance of UIImagePickerController

Acquiring and displaying the image

```
41          imagePicker.sourceType = UIImagePickerControllerSourceTypePhotoLibrary;
42      }
43      imagePicker.delegate = self;
44
45      // Place image picker on the screen
46      [self presentViewController: imagePicker animated: YES completion: NULL];
47  }
48
49  - (void) viewWillAppear:(BOOL) animated
```

Testing the app



Associating the Image with the Item

- If we build and run the application and select an INIItem to see its details and then tap the camera button on the UIToolbar we have:

  UIImagePickerController's interface will appear on the screen and you can take a picture or choose an existing image if your device does not have a camera.

  If you are working on the simulator, you can open Safari in the simulator and navigate to a page with an image. Click and hold the image and then choose Save Image to save it in the simulator's photo library.

  Selecting an image dismisses the UIImagePickerController and returns you to the detail view. However, you do not have a reference to the photo once the image picker disappears and it will not show with the item

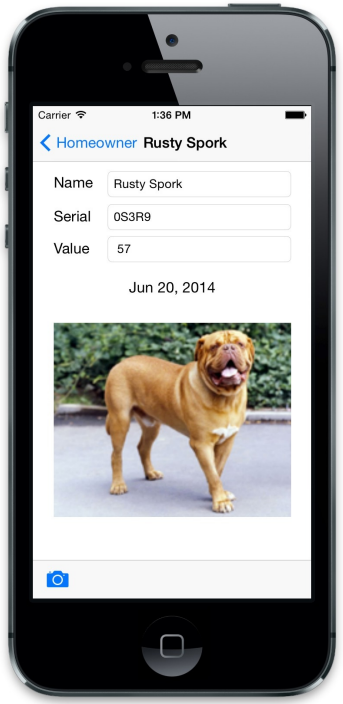
Associating the Image with the Item (cont.)

- In INIDetailViewController implement the method

  - (void) imagePickerController:(UIImagePickerController *) picker didFinishPickingMediaWithInfo:(NSDictionary *) info

- This will associate the image with the item and show it but will not save it

```
48
49 - (void) imagePickerController:( UIImagePickerController *) picker didFinishPickingMediaWithInfo:( NSDictionary *) info
50 {
51     // Get picked image from info dictionary
52     UIImage *image = info[ UIImagePickerControllerOriginalImage];
53     // Put that image onto the screen in our image view
54     self.imageView.image = image;
55     // Take image picker off the screen -
56     // you must call this dismiss method
57     [self dismissViewControllerAnimated: YES completion: NULL];
58 }
59
```

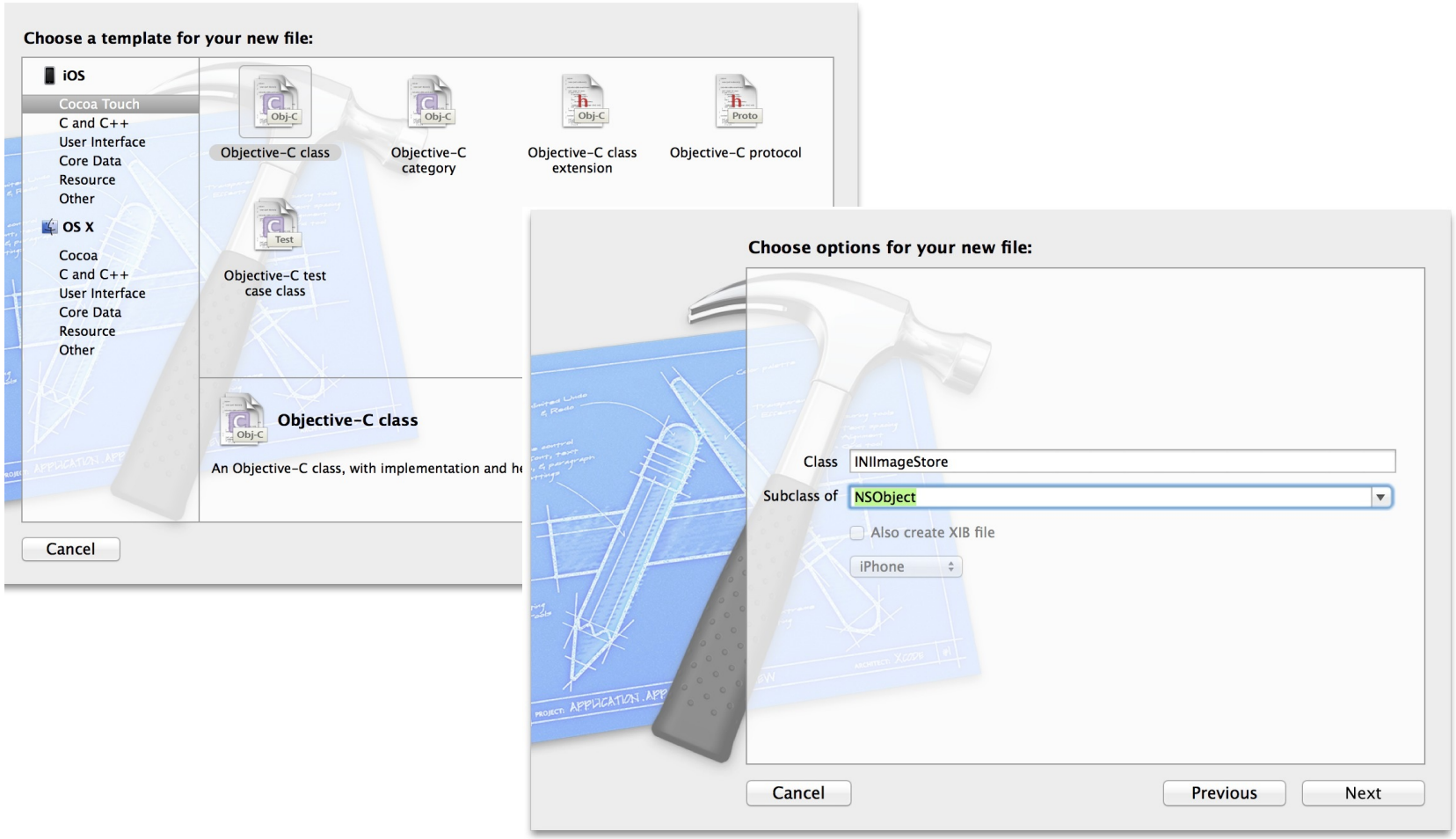


The message
- (void) imagePickerController:(UIImagePickerController *) picker didFinishPickingMediaWithInfo:(NSDictionary *) info
is sent to the image picker's delegate when a photo has been selected.

Creating the Image Shared Store and linking it to INIItem

- We must store images separately for memory and management issues.
- We store images in a dictionary
- We generate a random key that associate the image with the INIItem

Create Image Store



Implementing the image shared store

```
//  Copyright (c) 2014 CECS. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface INIImageStore : NSObject

+ (instancetype) sharedStore;
- (void) setImage:(UIImage *) image forKey:(NSString *) key;
- (UIImage *) imageForKey:(NSString *) key;
- (void) deleteImageForKey:( NSString *) key;

@end
```

Create the image store with a static sharedStore

```
//  Copyright (c) 2014 CECS. All rights reserved.
//

#import "INIImageStore.h"

@interface INIImageStore ()

@property (nonatomic, strong) NSMutableDictionary * dictionary;

@end

@implementation INIImageStore

+ (instancetype) sharedStore
{
    static INIImageStore *sharedStore;
    if (!sharedStore) {
        sharedStore = [[self alloc] initPrivate];
    }
    return sharedStore;
}

// No one should call init
- (instancetype) init
{
    [NSException raise:@"Singleton" format:@"Use +[INIImageStore sharedStore]"];
    return nil;
}

// Secret designated initializer
- ( instancetype) initPrivate
{
    self = [super init];
    if (self) {
        _dictionary = [[ NSMutableDictionary alloc] init];
    }
    return self;
}
```

```
- (void) setImage:(UIImage *) image forKey:(NSString *) key
{
    [self.dictionary setObject: image forKey: key];
}

- (UIImage *) imageForKey:( NSString *) key
{
    return [self.dictionary objectForKey: key];
}

- (void) deleteImageForKey:( NSString *) key
{
    if (!key) {
        return;
    }
    [self.dictionary removeObjectForKey: key];
}
@end
```

Creating and Using Keys

- When an image is added to the store, it will be put into a dictionary under a unique key, and the associated INIItem object will be assigned that key.
- When the INIDetailViewController wants an image from the store, it will ask its item for the key and search the dictionary for the image.
- Add a property to INIItem.h to store the key using itemKey as an NSString property toINIItem
- The image keys need to be unique in order for your dictionary to work.
- We will use the Cocoa Touch mechanism for creating universally unique identifiers (UUIDs).
- Objects of type NSUUID represent a UUID and are generated using the time, a counter, and a hardware identifier,
- When represented as a string, UUIDs look something like this:

4A73B5D2- A6F4- 4B40- 9F82- EA1E34C1DC04

Item Key Creation

```
13      int _valueInDollars;
14      NSDate *_dateCreated;
15  }
16  |
17  @property (nonatomic, copy) NSString * itemKey;
18
19  +(instancetype) randomItem;
```

```
 8
 9  #import "INIDetailViewController.h"
10  #import "INIItem.h"
11  #import "INIImageStore.h"
12
13  @interface INIDetailViewController () < UINavigationControllerDelegate, UIImagePickerControllerDelegate>
14
```

```
-(instancetype) initWithName:(NSString *)name valueInDollars:(int)value serialNu
{
    // Call The "Designated" initializer for the super class
    self = [super init];
    if (self)
    {
        // initialize the instant variables
        [self setItemName: name];
        [self setValueInDollars: value];
        [self setSerialNumber: sNumber];
        _dateCreated = [[NSDate alloc] init];

        // Create an NSUUID object - and get its string representation
        NSUUID *uuid = [[NSUUID alloc]init];
        NSString *key = [uuid UUIDString];
        _itemKey = key;
    }
    return self;
}
```

Saving and removing the image in the image store

- Update INIDetailViewController.m

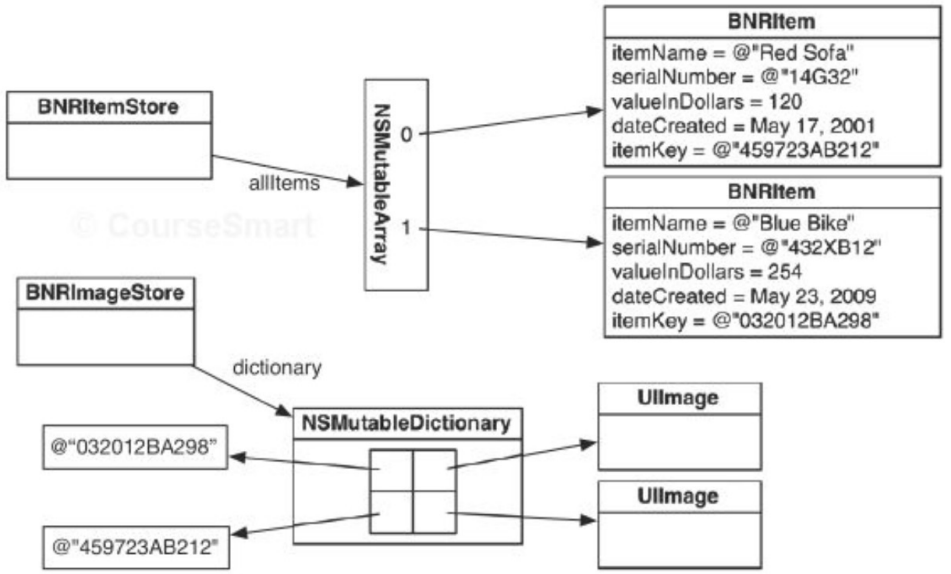
```
49
50  - (void) imagePickerController:( UIImagePickerController *) picker
        didFinishPickingMediaWithInfo:( NSDictionary *) info
51  {
52      // Get picked image from info dictionary
53      UIImage *image = info[UIImagePickerControllerOriginalImage];
54
55      //Store the image in the INIImageStore for this key
56      [[INIImageStore sharedStore] setImage: image forKey: self.item.itemKey];
57
58      // Put that image onto the screen in our image view
59      self.imageView.image = image;
60
61      // Take image picker off the screen -
62      // you must call this dismiss method
63      [self dismissViewControllerAnimated: YES completion: NULL];
64  }
65
```

- Update INIItemStore

```
8
9   #import "INIItemStore.h"
10  #import "INIImageStore.h"
11
```

```
65  - (void) removeItem:(INIItem *) item
66  {
67      NSString *key = item.itemKey;
68      [[INIImageStore sharedStore] deleteImageForKey: key];
69      [self.privateItems removeObjectIdenticalTo: item];
70  }
```

Wrapping it up for now

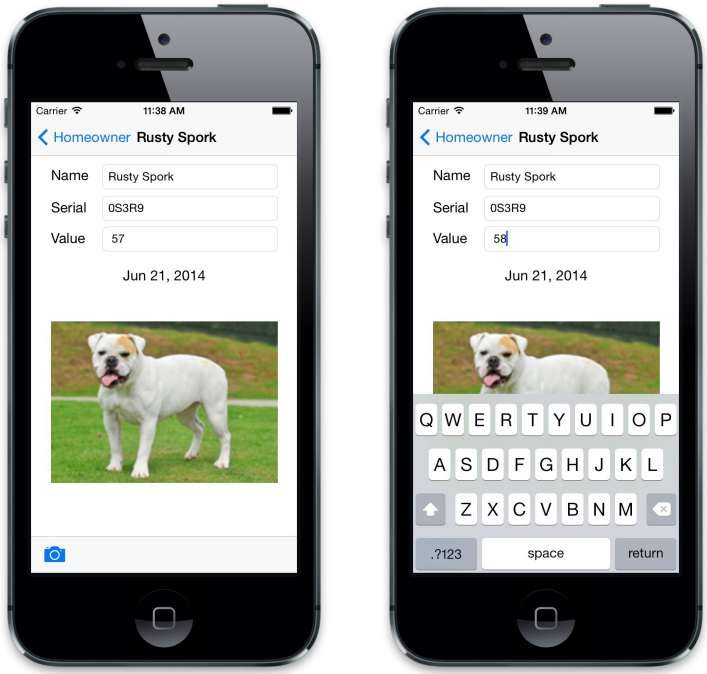


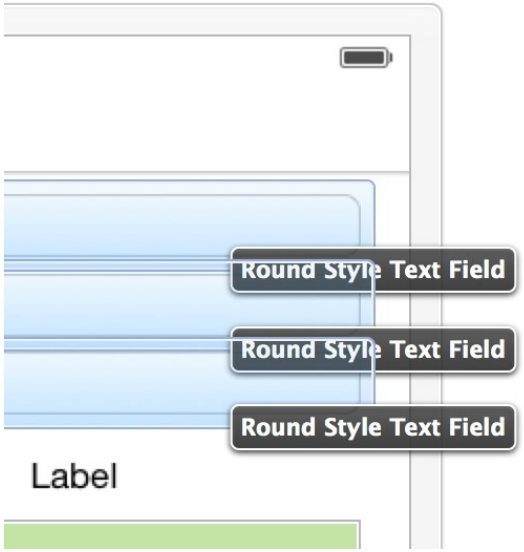
Use imageKey to retrieve the image from the imageStore and display it

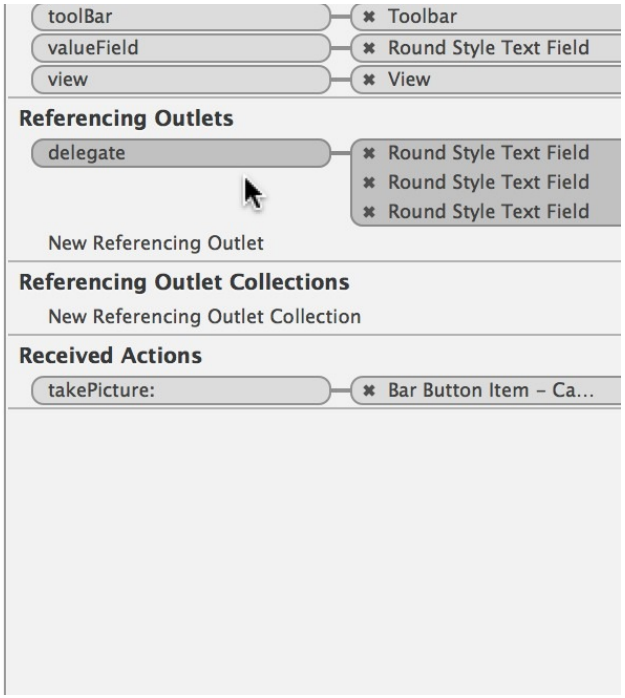
```
65
66  - (void) viewWillAppear:(BOOL) animated
67  {
68      [super viewWillAppear: animated];
69      INIItem *item = self.item;
70      self.nameField.text = item.itemName;
71      self.serialNumberField.text = item.serialNumber;
72      self.valueField.text = [NSString stringWithFormat:@"% d", item.valueInDollars];
73      // You need an NSDateFormatter that will turn a date into a simple date string static
74      NSDateFormatter *dateFormatter;
75      if (!dateFormatter) {
76          dateFormatter = [[NSDateFormatter alloc] init];
77          dateFormatter.dateStyle = NSDateFormatterMediumStyle;
78          dateFormatter.timeStyle = NSDateFormatterNoStyle;
79      }
80      // Use filtered NSDate object to set dateLabel contents
81      self.dateLabel.text = [dateFormatter stringFromDate: item.dateCreated];
82
83      NSString *itemKey = self.item.itemKey;
84      // Get the image for its image key from the image store
85      UIImage *imageToDisplay = [[INIImageStore sharedStore] imageForKey: itemKey];
86      // Use that image to put on the screen in the imageView
87      self.imageView. image = imageToDisplay;
88  }
89
```

Dismissing the Keyboard

- When the keyboard appears on the screen in the item detail view, it obscures INIDetailViewController's imageView.
- We need to figure out a way to make the text fields relinquish the first responder status.
- Recall, in chapter 10 we connected the text fields to the delegate outlet of the file's owner.
- INIDetailedViewController is the delegate for these text fields







Dismissing the Keyboard (cont.)

- We need to implement the delegate method

  textFieldShouldReturn:

  to have the text field resign its first responder status when the return key is tapped.
- First, in INIDetailViewController.m, have INIDetailViewController conform to the UITextFieldDelegate protocol.

```
#import "INIImageStore.h"

@interface INIDetailViewController () < UINavigationControllerDelegate, UIImagePickerControllerDelegate,
    UITextFieldDelegate>

```

This message is sent when we hit the “return” key

```
- (BOOL)textFieldShouldReturn:(UITextField *) textField
{
    [textField resignFirstResponder];
    return YES;
}
```

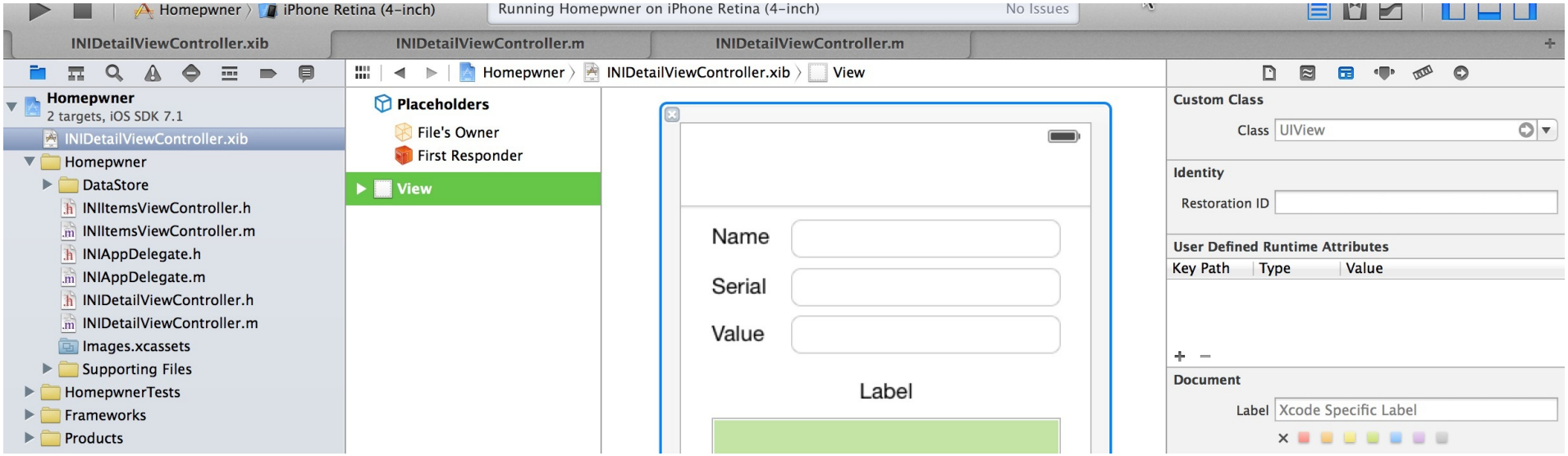
More On Dismessing The Keyboard

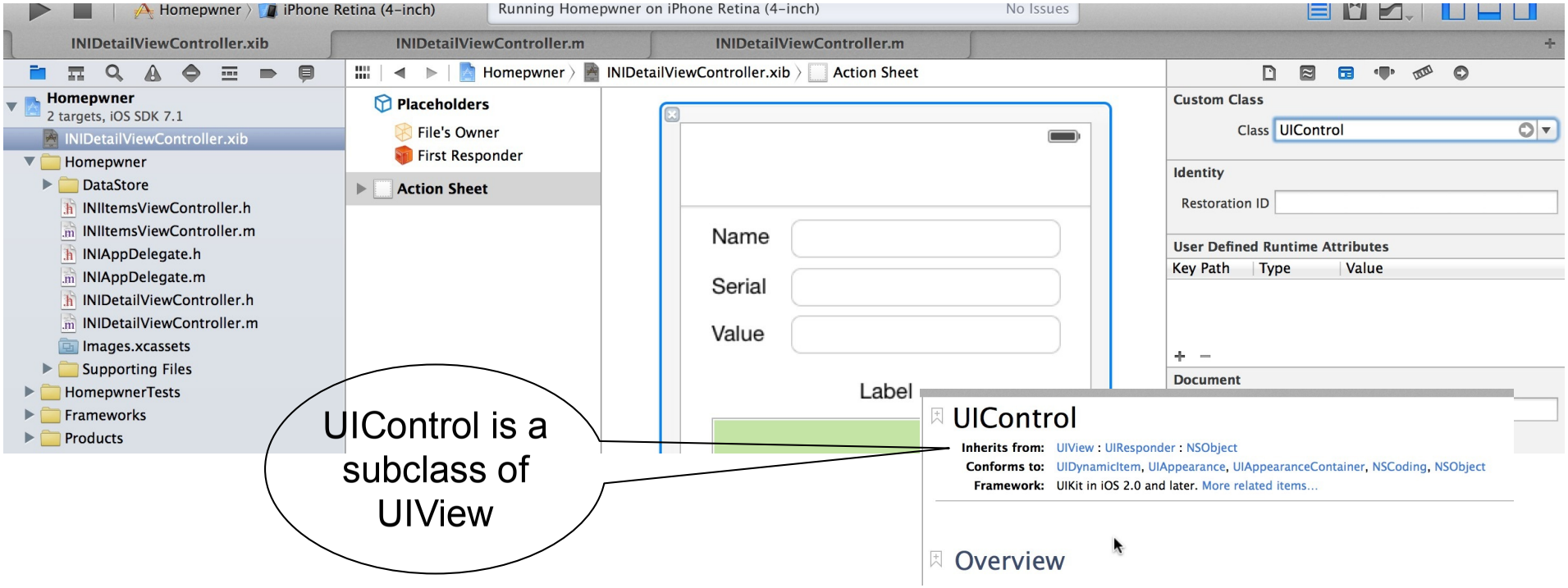
- It would be stylish to also dismiss the keyboard if the user taps anywhere else on INIDetailViewController's view.
- We can dismiss the keyboard by sending the view the message

  endEditing:

  which will cause the text field (as a subview of the view) to resign as first responder.
- Recall that classes like UIButton can send an action message to a target when tapped.
- Buttons inherit this target-action behavior from their superclass, UIControl.
- We are going to change the view of INIDetailViewController from an instance of UIView to an instance of UIControl so that it can handle touch events.
- In INIDetailViewController.xib, select the main View object. Open the identity inspector and change the view's class to UIControl

Changing The Class Of The View





Implement “endEditing:”