

# Chapter 19

## Subclassing UITableViewCell

- Creating UITableViewCell
- Image Manipulation
- Relaying Actions from UITableViewCells
- Variable Capturing

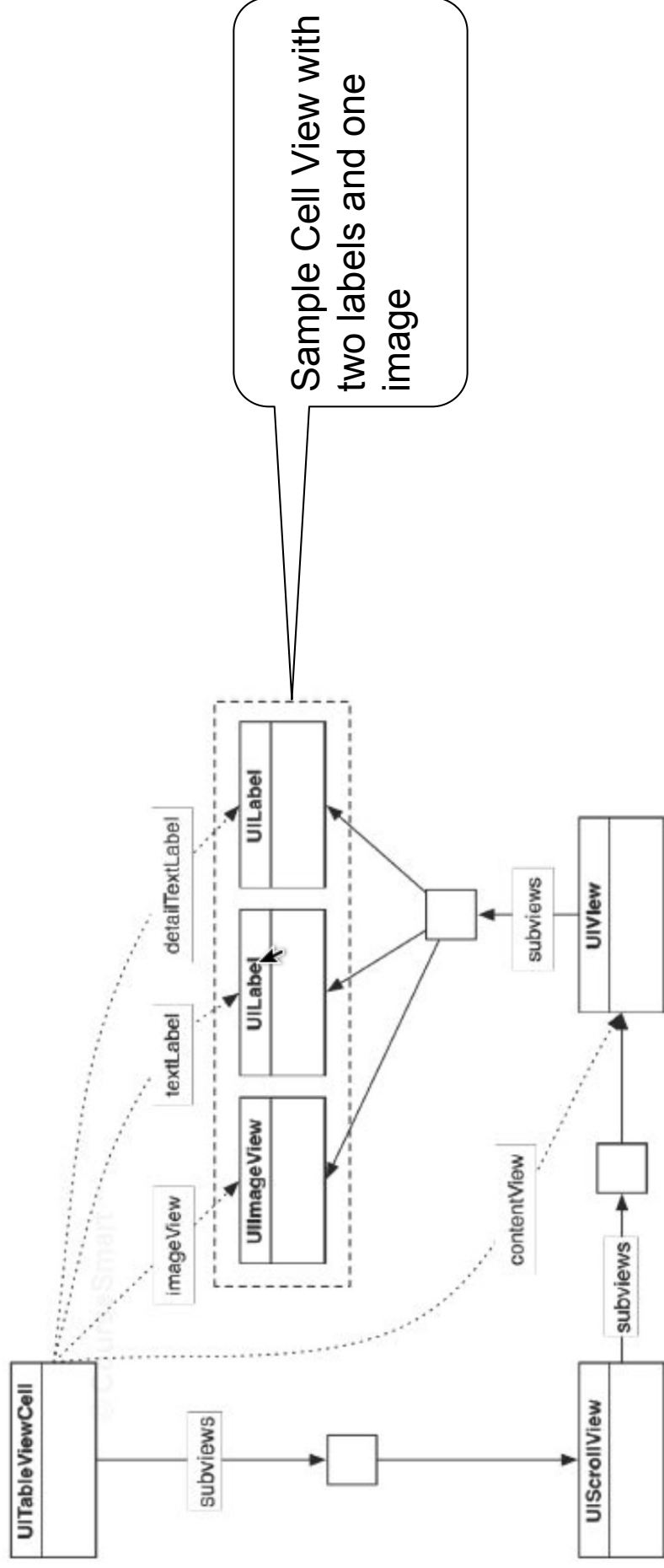
# Subclassing UITableViewCell

- Recall:
  - A UITableView displays a list of UITableViewCell objects.
  - the basic UITableViewCell has a.textLabel, detailedTextLabel, and imageView which is sufficient in most cases.
  - When we need a cell with more detail or a different layout, we subclass UITableViewCell.
  - In this chapter we create a custom UITableViewCell to display INItems instances more effectively.



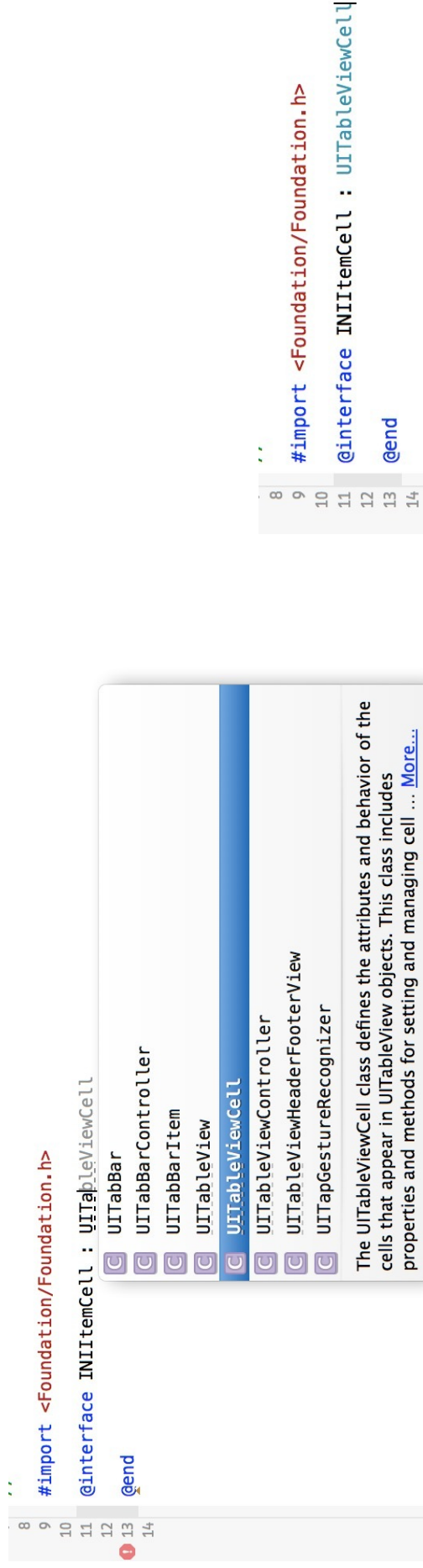
# Creating INIItemCell

- UITableViewCell is a subclass of UIView
- Typically, when subclassing UIView you override its drawRect: as we did in the Hypnosis app.
- For UITableViewCell we customize its appearance by adding subviews to the cell's content view.



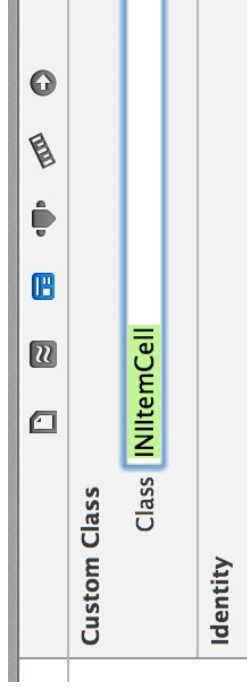
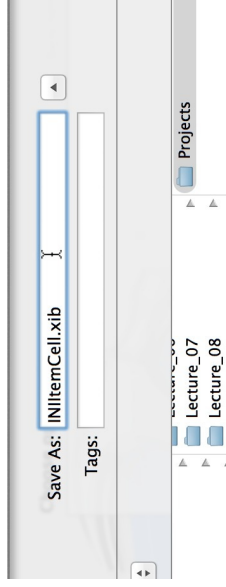
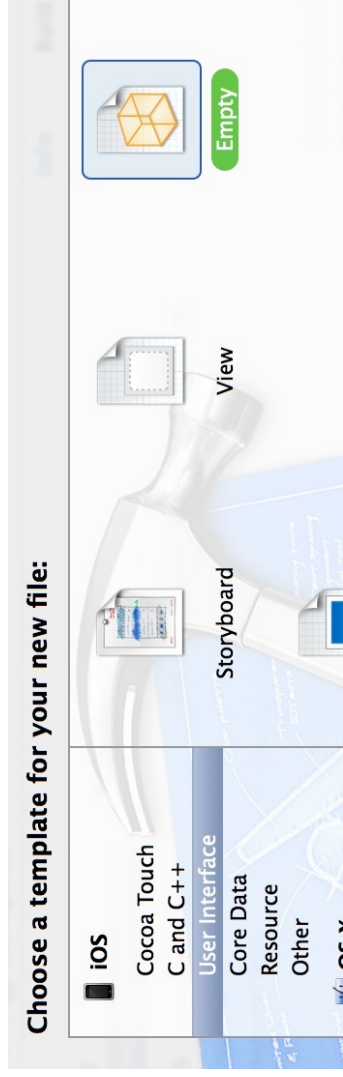
# Creating INIItemCell

- Create new Class named INIItemCell from NSObject and modify it to be a subclass of UITableViewCell



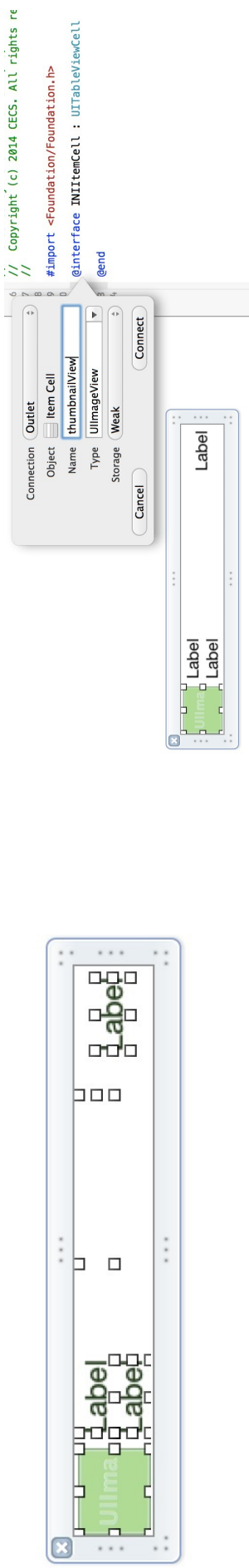
# Creating INItemCell

- The easiest way to configure a UITableViewCell subclass is with a XIB file.
- Create an empty XIB file and name it INItemCell.xib.
- This file will contain a single instance of INItemCell.
- When the table view needs a new cell, it will create one from this XIB file.
- Drag an instance of “UITableViewCell” onto the canvas



# Expose Properties of NSIndexPath

- The NSIndexPath will display three text elements and an image.
- Drag three UILabel objects and one UIImageView object onto the cell.



```
// Created by Ibrahim Imam on 7/13/14.
// Copyright (c) 2014 CECS. All rights reserved.
//
```

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

```
@interface NSIndexPath : UITableViewCell
@property (weak, nonatomic) IBOutlet UIImageView *thumbnailView;
@property (weak, nonatomic) IBOutlet UILabel *nameLabel;
@property (weak, nonatomic) IBOutlet UILabel *serialNumberLabel;
@property (weak, nonatomic) IBOutlet UILabel *valueLabel;
@end
```

editingAccessoryView	nameLabel	* Label - La
selectedBackgroundView	serialNumberLabel	* Label - La
	thumbnailView	* Image Vie
	valueLabel	* Label - La
Outlet Collections		
gestureRecognizers		
Referencing Outlets		
New Referencing Outlet		
Referencing Outlet Collections		

## Using INItemCell

- In INItemsViewController's method

`tableView: cellForRowAtIndexPath:`

we will create an instance of INItemCell for every row in the table.

- In INItemsViewController.m, import the header file for INItemCell so that INItemsViewController knows about it.
- Register the nib file in INItemsViewController's `viewDidLoad`
- Modify ItemsViewController's `tableView:cellForRowAtIndexPath` to use the newly registered nib



```

- (void) viewDidLoad
{
    [super viewDidLoad];
    [self.tableView registerClass:[UITableViewCell class] forCellReuseIdentifier:@"UITableViewCell"];
}

```

```

- (void) viewDidLoad
{
    [super viewDidLoad];
    // Load the NIB file
    UINib *nib = [UINib nibWithNibName:@"INIItemCell"bundle: nil];
    // Register this NIB, which contains the cell
    [self.tableView registerNib: nib forCellReuseIdentifier:@"INIItemCell"];
}

```

```

- (UITableViewCell *) tableView:(UITableView *) tableView cellForRowAtIndexPath:(NSIndexPath *) indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"UITableViewCell" forIndexPath: indexPath];
    NSArray *items = [[INIItemStore sharedStore] allItems];
    INIItem * item = items[indexPath.row];
    cell.textLabel.text = [ item description]; return cell;
}

```

```

- (UITableViewCell *) tableView:(UITableView *) tableView cellForRowAtIndexPath:(NSIndexPath *) indexPath
{
    INIItemCell *cell = [tableView dequeueReusableCellWithIdentifierWithIdentifier:@"INIItemCell"forIndexPath: indexPath];
    NSArray *items = [[INIItemStore sharedStore] allItems];
    INIItem *item = items[indexPath.row];

    //Configure the cell with the INIItem
    cell.textLabel.text = item.itemName;
    cell.serialNumberLabel.text = item.serialNumber;
    cell.valueLabel.text = [NSString stringWithFormat:@"%d", item.valueInDollars];
    return cell;
}

```

Edit		Homeowner	+
Shiny Spork	0X2Q7	\$ 3	
Rusty Spork	0S3R9	\$ 57	
My Car	1234	\$ 10000	
My Bike	4321	\$ 2000	



# Image Manipulation

- To avoid the overhead of compressing the full images into small images to fit in the item's cell view we will be creating and using thumbnails for these images.
- To create a thumbnail of an `INItem` image, we will draw a scaled-down version of the full image to an offscreen context and keep a reference to that new image inside the `INItem` instance.
- We will also need a place to store this thumbnail image so that it can be reloaded when the application launches again.

# Creating the thumbnail

```

16  @property (nonatomic, copy) NSString * itemKey;
17
18
19  @property (strong, nonatomic) UIImage *thumbnail;
20  -(void) setThumbnailFromImage:( UIImage *) image;
21

```

```

111 -(void)setThumbnailFromImage:(UIImage *)image
112 {
113     CGSize origImageSize = image.size;
114     //The rectangle of the thumbnail
115     CGRect newRect = CGRectMake(0, 0, 40, 40);
116
117     //Figure out a scaling ratio to make sure we maintain the same aspect ratio
118     float ratio = MAX(newRect.size.width /origImageSize.width, newRect.size.height /origImageSize.height);
119
120     //Create a transparent bitmap context with a scaling factor
121     //equal to that of the screen
122     UIGraphicsBeginImageContextWithOptions(newRect.size, NO, 0.0);
123
124     //Create a path that is a rounded rectangle
125     UIBezierPath *path = [UIBezierPath bezierPathWithRoundedRect: newRect cornerRadius: 5.0];
126
127     //Make all subsequent drawing clip to this rounded rectangle
128     [path addClip];
129
130
131     //Center the image in the thumbnail rectangle
132     CGRect projectRect;
133     projectRect.size.width = ratio *origImageSize.width;
134     projectRect.size.height = ratio *origImageSize.height;
135     projectRect.origin.x = (newRect.size.width -projectRect.size.width)/2.0;
136     projectRect.origin.y = (newRect.size.height -projectRect.size.height)/2.0;
137
138     //Draw the image on it
139     [image drawInRect: projectRect];
140
141     //Get the image from the image context; keep it as our thumbnail
142     UIImage *smallImage = UIGraphicsGetImageFromCurrentImageContext();
143     self.thumbnail = smallImage;
144
145     //Cleanup image context resources; we're done
146     UIGraphicsEndImageContext();
147 }

```

Compression  
ratio and clipping

Thumbnail's  
rectangle

Setting  
thumbnail and its  
data

# Update INIdetailedViewController and INItemsViewController

- After selecting the image for the item set the thumbnail data for that image in detailedViewController
- Once the item has thumbnail data we can use it to construct the image thumbnail in itemsViewController

```
128 - (void) imagePickerController:( UIImagePickerController *) picker didFinishPickingMediaWithInfo:( NSDictionary *) info
129 {
130     // Get picked image from info dictionary
131     UIImage *image = info[UIImagePickerControllerOriginalImage];
132     [self.item setThumbnailFromImage: image];
133     //Store the image in the INItemImageStore for this key
134     [[INItemImageStore sharedStore] setImage: image forKey: self.item.itemKey];
135
136
137
```

```
- (UITableViewCell *) tableView:(UITableView *) tableView cellForRowAtIndexPath:(NSIndexPath *) indexPath
{
    INItemCell *cell = [tableView dequeueReusableCellWithIdentifierWithIdentifier:@"INItemCell" forIndexPath: indexPath];
    NSArray *items = [[INItemStore sharedStore] allItems];
    INItem *item = items[indexPath.row];

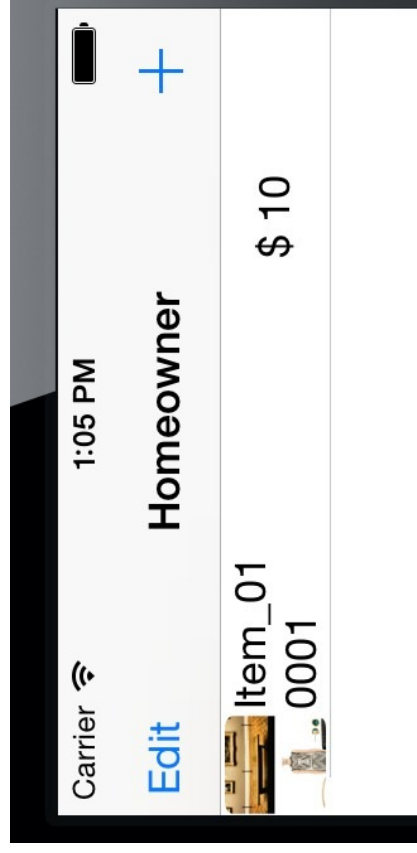
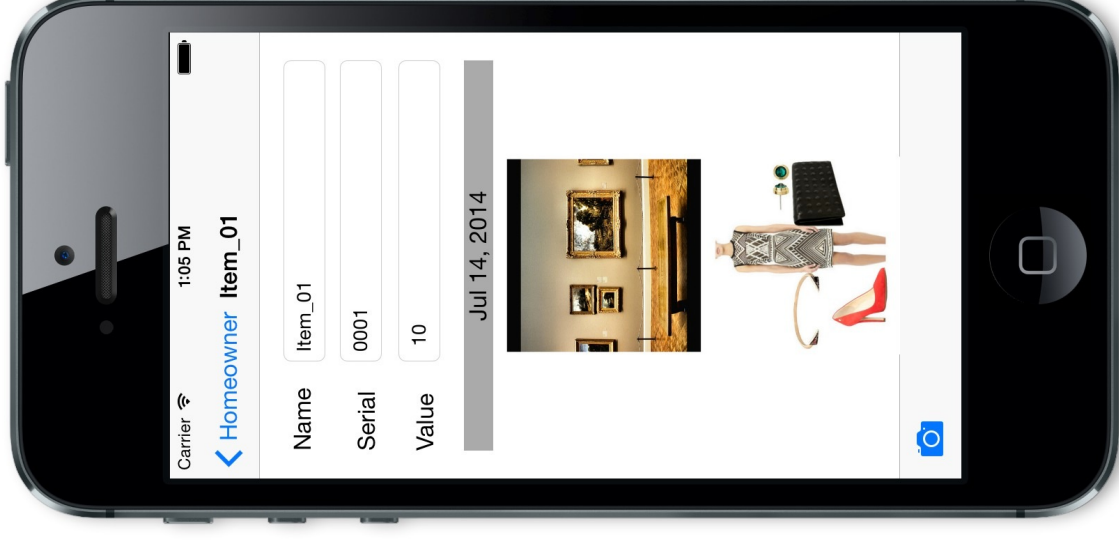
    //Configure the cell with the INItem
    cell.nameLabel.text = item.itemName;
    cell.serialNumberLabel.text = item.serialNumber;
    cell.valueLabel.text = [NSString stringWithFormat:@"%s d", item.valueInDollars];
    cell.thumbnailView.image = item.thumbnail;
    return cell;
}
```

# Adding thumbnailData to the archive

```

48 -(instancetype) initWithCoder:(NSCoder *) aDecoder
49 {
50     self = [super init];
51     if (self) {
52         _itemName = [aDecoder decodeObjectForKey:@"itemName"];
53         _serialNumber = [aDecoder decodeObjectForKey:@"serialNumber"];
54         _dateCreated = [aDecoder decodeObjectForKey:@"dateCreated"];
55         _itemKey = [aDecoder decodeObjectForKey:@"itemKey"];
56         _valueInDollars = [aDecoder decodeIntForKey:@"valueInDollars"];
57         _thumbnail = [aDecoder decodeObjectForKey:@"thumbnail"];
58     }
59     return self;
60 }
61
62 -(void) encodeWithCoder:(NSCoder *) aCoder
63 {
64     [aCoder encodeObject: self.itemName forKey:@"itemName"];
65     [aCoder encodeObject: self.serialNumber forKey:@"serialNumber"];
66     [aCoder encodeObject: self.dateCreated forKey:@"dateCreated"];
67     [aCoder encodeObject: self.itemKey forKey:@"itemKey"];
68     [aCoder encodeInt: self.valueInDollars forKey:@"valueInDollars"];
69     [aCoder encodeObject: self.thumbnail forKey:@"thumbnail"];
70 }
71
72

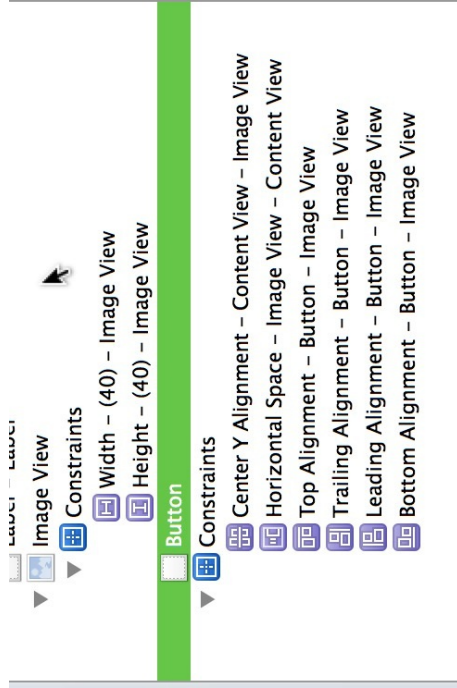

```





# Relaying Actions from UITableViewCell

- Create a transparent button on top of the thumbnail
- Set button type to custom to make sure it is transparent
- connect it to an action in INIItemCell.h called showImage:
- Add two properties, controller and tableView to keep the cell aware of its controller and the tableView it belongs to.



```
0 1 2 3 4 5 6 7
@implementation INIItemCell
- (IBAction) showImage:(id) sender
{
    NSLog(@"I am in showImage");
}

```

07-14 15:13:43.758 Homepwner[2823:60b] I am in showImage

# Design Issues

- So now you have a button that will send `showImage:` to `BNRItemCell` when it is tapped.
- Obviously, you will have to implement that method, but here you run into a problem:
  - this message is being sent to the `INItemCell`, but `INItemCell` is not a controller and does not have access to any of the image data necessary to get the full- size image.
  - In fact, it does not even have access to the `INItem` whose thumbnail it is displaying.
- We might consider letting `INItemCell` keep a pointer to the `INItem` it displays. But table view cells are view objects, and they should not manage model objects or be able to present additional interfaces ( like the `UIPopoverController`).

# Adding a block to the cell subclass

Return type of block

A comma-delimited list of arguments

```
void (^)(NSString *someArg, int anotherArg)
```

Notation to specify this is a block

```
void (^blockReturningVoidWithVoidArgument)(void);  
int (^blockReturningIntWithIntAndCharArguments)(int, char);  
void (^arrayOfTenBlocksReturningVoidWithIntArgument[10])(int);
```

```
15 @property (weak, nonatomic) IBOutlet UILabel *valueLabel;  
16  
17 @property (copy, nonatomic) void (^actionBlock)(void);  
18
```

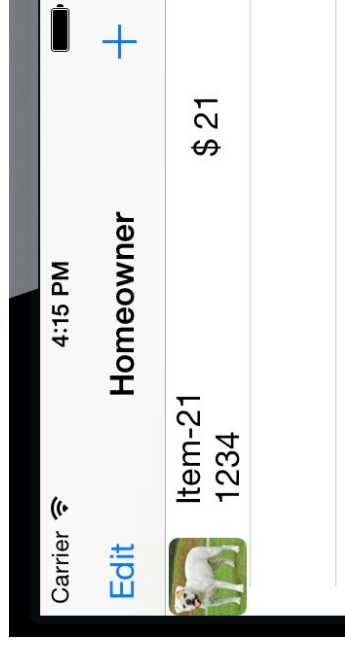
```
11 @implementation INIItemCell  
12  
13 - (IBAction) showImage:(id) sender  
14 {  
15     NSLog(@"I am in showImage");  
16     if (self.actionBlock) {  
17         self.actionBlock();  
18     }  
19 }
```

```
- (UITableViewCell *) tableView:(UITableView *) tableView cellForRowAtIndexPath:(NSIndexPath *) indexPath  
{
```

```
    INIItemCell *cell = [tableView dequeueReusableCellWithIdentifierWithIdentifier:@"INIItemCell" forIndexPath: indexPath];  
    NSArray *items = [[INIItemStore sharedInstance] allItems];  
    INIItem *item = items[indexPath.row];
```

```
    //Configure the cell with the INIItem  
    cell.textLabel.text = item.itemName;  
    cell.serialNumberLabel.text = item.serialNumber;  
    cell.valueLabel.text = [NSString stringWithFormat:@"%d", item.valueInDollars];  
    cell.thumbnailView.image = item.thumbnail;  
    cell.actionBlock = ^{  
        NSLog(@"Going to show image for %@", item);  
    };  
    return cell;  
}
```

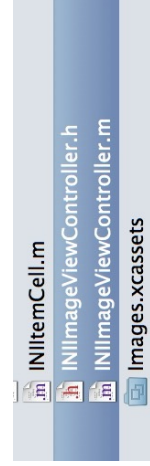
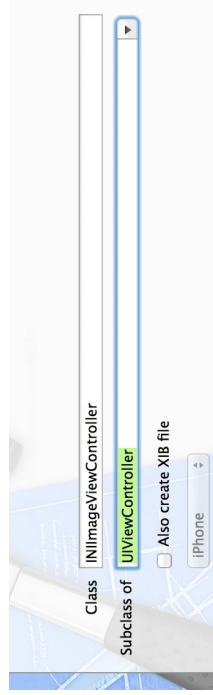
```
2014-07-14 16:14:25.687 Homepwner[3018:60b] I am in showImage  
2014-07-14 16:14:25.689 Homepwner[3018:60b] Going to show image for Item-21 (1234): Worth $21, recorded on  
2014-07-14 20:12:53 +0000
```





# Presenting the image in a popover controller

- UINavigationController we need to change the action block to grab the INItem associated with the cell whose button was tapped and display its image in a UIPopoverController.
- To display an image in a popover, We need a new View Controller whose view shows an image.
- We will create a new Objective-C subclass named UINavigationController with UINavigationController as its superclass



Expose the image

```
@interface UINavigationController : UINavigationController
@property (strong, nonatomic) UIImage *image;
@end
```

```
26 -(void) loadView
27 {
28     UIImageView *imageView = [[UIImageView alloc] init];
29     imageView.contentMode = UIViewContentModeScaleAspectFit;
30     self.view = imageView;
31 }
```

```
UIImageView *imageView = [[UIImageView alloc] init];
imageView.contentMode = UIViewContentModeScaleAspectFit;
self.view = imageView;
```

This view controller will only have one view that we will create programmatically.

- When an instance of `UIImageViewController` is created, it will be given an image.
- In `UIImageViewController.m`, implement `viewWillAppear:` to set the view's image from the passed image.

```
-(void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    //We must cast the view to UIImageView so the compiler knows it
    //is okay to send it setImage:
    UIImageView *imageView = (UIImageView *)self.view;
    imageView.image = self.image;
}
```

# Now you can finish implementing the action block

```

13 #import "INIItemsStore.h"
14 #import "INIItemCell.h"
15 #import "INIImageStore.h"
16 #import "INIImageViewController.h"
17
18 @interface INIItemsViewController () <UIPopoverControllerDelegate>
19
20 @property (strong, nonatomic) UIPopoverController *imagePopover;
21
22 @end
23
INIItem *item = items[indexPath.row];
//Configure the cell with the INIItem
cell.nameLabel.text = item.itemName;
cell.serialNumberLabel.text = item.serialNumber;
cell.valueLabel.text = [NSString stringWithFormat:@"%d", item.valueInDollars];
cell.thumbnailView.image = item.thumbnail;
cell.actionBlock = ^{
    NSLog(@"Going to show image for %@", item);
    if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad){
        NSString *itemKey = item.itemKey;
        //If there is no image, we don't need to display anything
        UIImage *img = [[INIImageStore sharedInstance] imageForKey: itemKey];
        if (!img){
            return;
        }
        //Make a rectangle for the frame of the thumbnail relative to our table view
        //Note: there will be a warning on this line that we'll soon discuss
        CGRect rect = [self.view convertRect: cell.thumbnailView.bounds
                                fromView: cell.thumbnailView];
        //Create a new INIImageViewController and set its image
        INIImageViewController *ivc = [[INIImageViewController alloc] init];
        ivc.image = img;
        //Present a 600x600 popover from the rect
        self.imagePopover = [UIPopoverController alloc] initWithContentViewController: ivc];
        self.imagePopover.delegate = self;
        self.imagePopover.popoverContentSize = CGSizeMake(600, 600);
        [self.imagePopover presentPopoverFromRect: rect
                                inView: self.view
                                permittedArrowDirections: UIPopoverArrowDirectionAny
                                animated: YES];
    }
};
return cell;
}

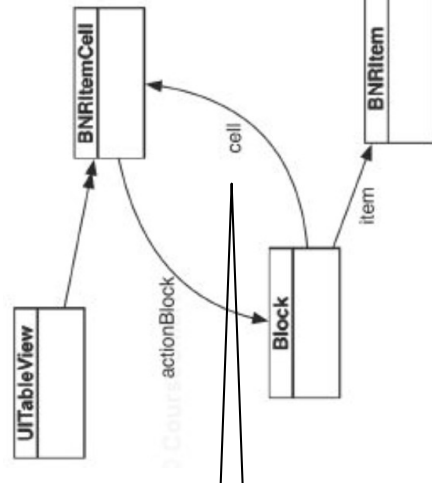
```

⚠️ ARC Retain Cycle  
Capturing 'cell' strongly in this block is likely to lead to a retain cycle

add a property to  
ang on to a  
opover controller  
nd have this  
lass conform to  
ie Popover  
;ontroller  
elegate protocol.

# Variable Capturing

- A block can use any variables that are visible within its enclosing scope.
- The enclosing scope of a block is the scope of the method in which it is defined.
- A block has access to all of the local variables, arguments passed to, and instance variables that belong to the object running the enclosing method.
- In our `actionBlock` the `BNRItem` ( `item`) and the `BNRItemCell` ( `cell`) have been captured from the enclosing scope.
- Blocks own the objects that they capture, and this can easily result in a strong reference cycle.
- The `actionBlock`: “Capturing ‘cell’ strongly in this block which is likely to lead to a strong reference cycle”.
- The cell has ownership of `actionBlock`, and `actionBlock` has strong ownership of `cell`





# The Final Block

```
- (UITableViewCell *) tableView:(UITableView *) tableView cellForRowAtIndexPath:(NSIndexPath *) indexPath
{
    INIItemCell *cell = [tableView dequeueReusableCellWithIdentifier:@"INIItemCell" forIndexPath: indexPath];
    NSArray *items = [[INIItemStore sharedInstance] allItems];
    INIItem *item = items[indexPath.row];

    //Configure the cell with the INIItem
    cell.nameLabel.text = item.itemName;
    cell.serialNumberLabel.text = item.serialNumber;
    cell.valueLabel.text = [NSString stringWithFormat:@"%d", item.valueInDollars];
    cell.thumbnailView.image = item.thumbnail;

    __weak INIItemCell *weakCell = cell;
    cell.actionBlock = ^{
        NSLog(@"Going to show image for %@", item);

        INIItemCell *strongCell = weakCell;
        if ([UIDevice currentDevice].userInterfaceIdiom == UIUserInterfaceIdiomPad){
            NSString *itemKey = item.itemKey;
            //If there is no image, we don't need to display anything
            UIImage *img = [[INIItemStore sharedInstance] imageForKey: itemKey];
            if (!img){
                return;
            }
            //Make a rectangle for the frame of the thumbnail relative to our table view
            CGRect rect = [ self.view convertRect: strongCell.thumbnailView.bounds
                                fromView: strongCell.thumbnailView];

            //Create a new INIImageViewController and set its image
            INIImageViewController *ivc = [[INIImageViewController alloc] init];
            ivc.image = img;
            //Present a 600x600 popover from the rect
            self.imagePopover = [[UIPopoverController alloc] initWithContentViewController: ivc];
            self.imagePopover.delegate = self;
            self.imagePopover.popoverContentSize = CGSizeMake(600, 600);
            [self.imagePopover presentPopoverFromRect: rect
                                     inView: self.view
                                     permittedArrowDirections: UIPopoverArrowDirectionAny
                                     animated: YES];
        }
    };
    return cell;
}
```