iOS Development
View Controller

Dong Duong

April 2013

Course Objective

- Be able to create iOS Application using Master-Detail template.
- Be able to show data to table view.
- Understand NavigationController.
- Combine NavigationController & Split ViewController & TableViewController.

Prerequisite

- Joined iOS overview course

Assessment Disciplines

- Class Participation : Required
- Assignment Completion : 100%
- Pass Score : >=70%

Course Timetable

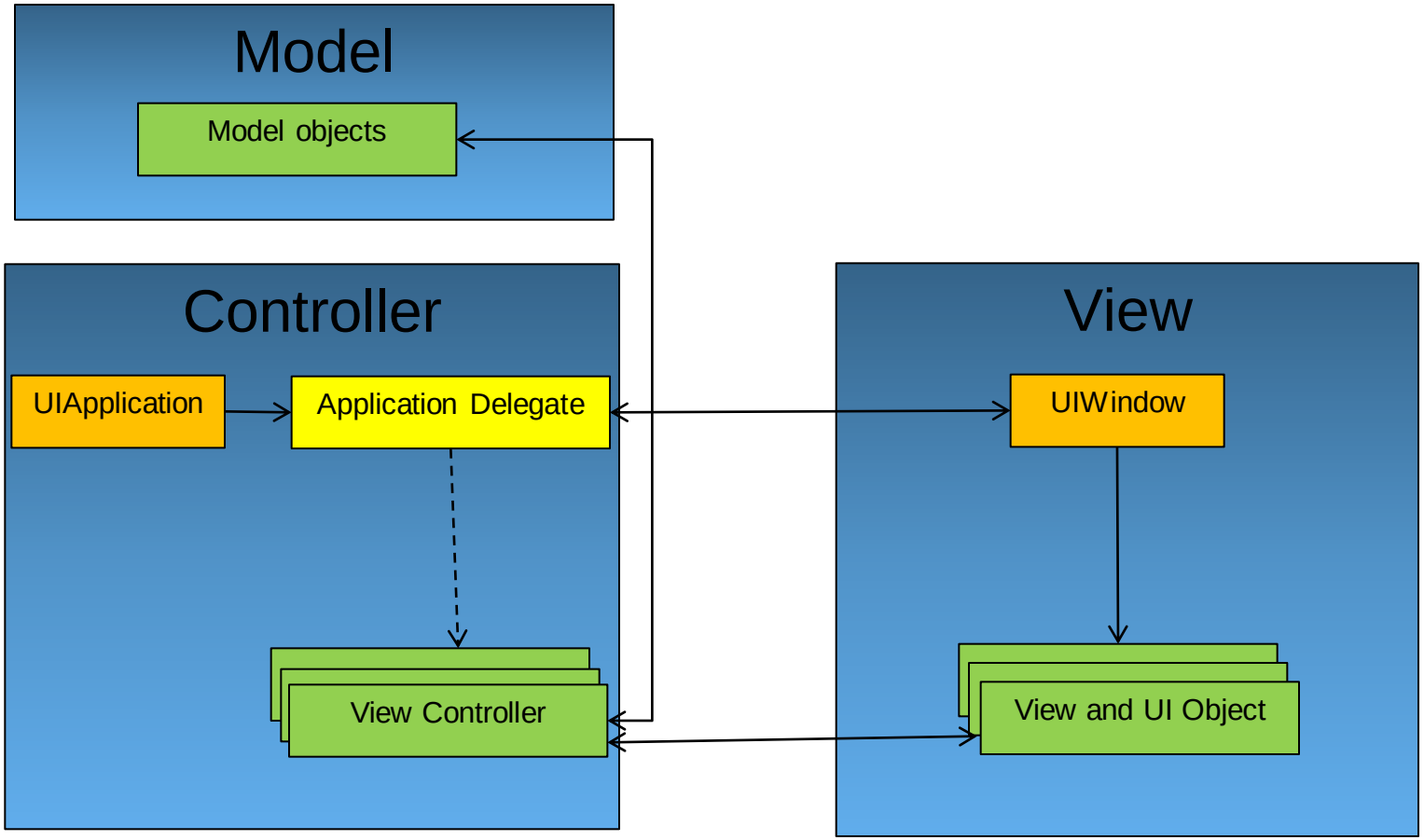
- Lecture Duration + Hands-on Labs: 9 hours

Agenda

- The Core Objects of Your App
- Split ViewController
- Practice 1
- Table View
- Practice 2
- NavigationController
- Practice 3
- Q&A

The Core Objects of Your App

The Core Objects of Your App



View

- Label
- Button
- TextField
- Tableview
- …..

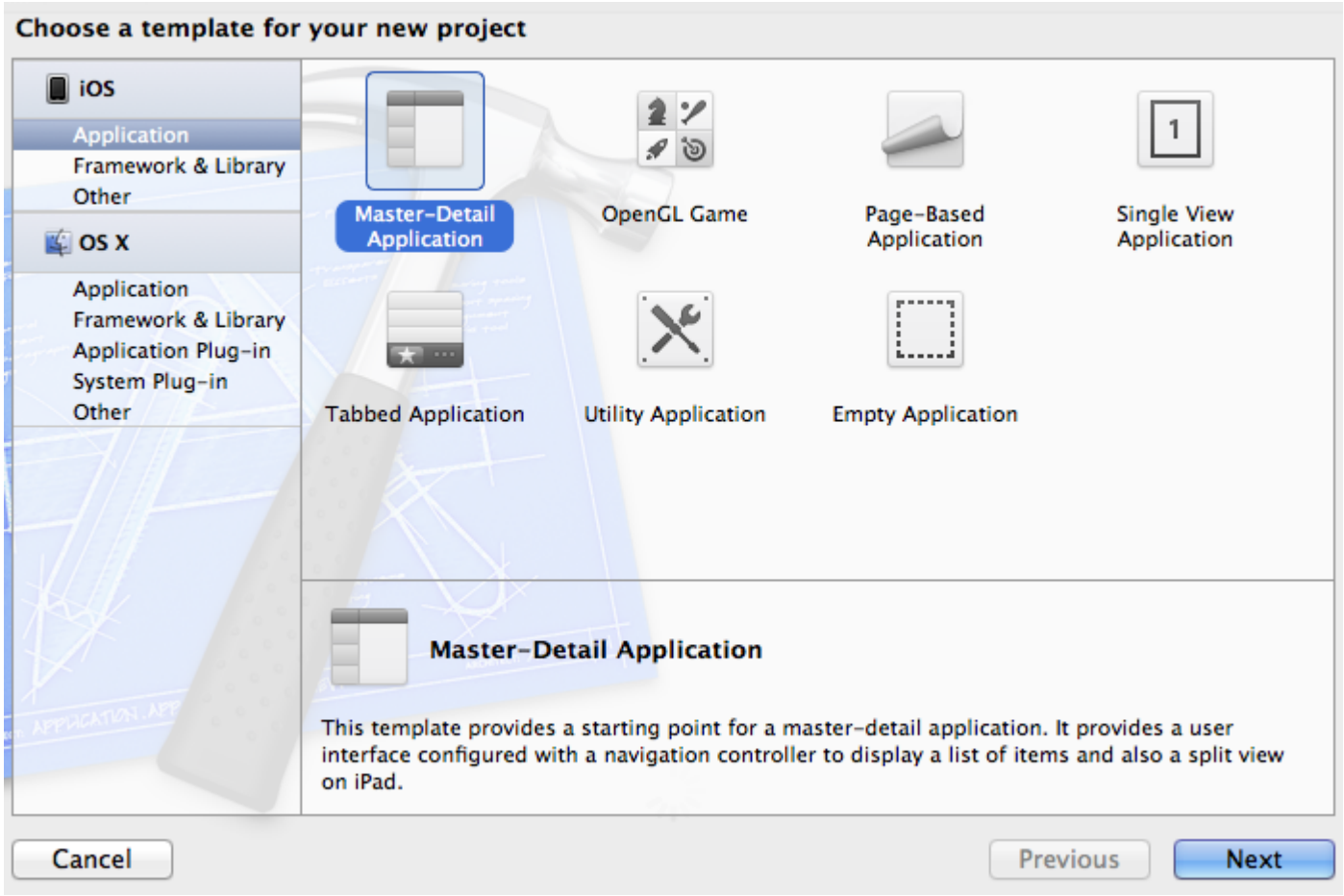
View Controller

- SplitViewController
- TableViewController
- NavigationViewController
- TabBarController
- PageViewController
- PopoverController

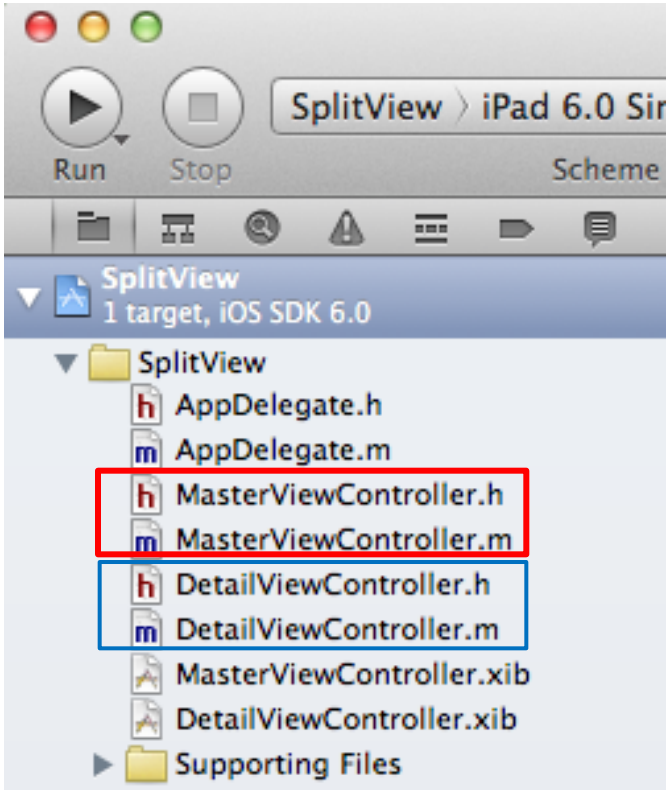
Split ViewController

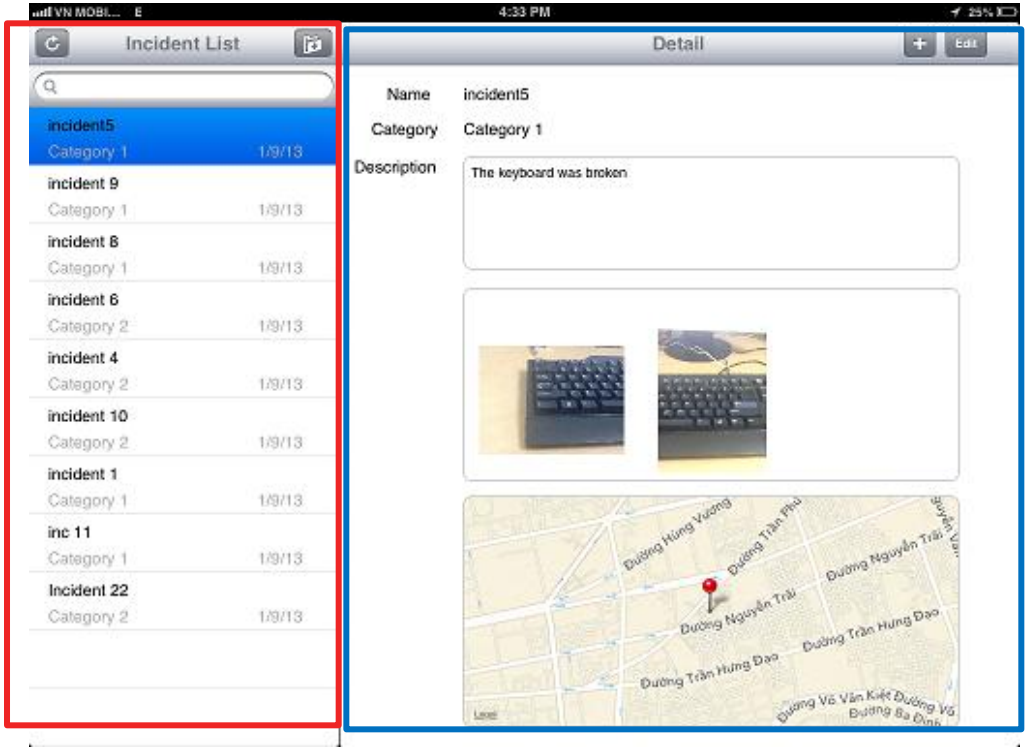
- The UISplitViewController class is a container view controller that manages two panes of information.

- Creating a SplitView Project
  - Go to Xcode and select File -> New -> New Project….
  - From the iOS *Application group,* select *Master-Detail Application and click Next.*



- Creating a SplitView Project





- AppDelegate

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

    MasterViewController *masterViewController = [[MasterViewController alloc]
        initWithNibName:@"MasterViewController" bundle:nil];
    UINavigationController *masterNavigationController = [[UINavigationController alloc]
        initWithRootViewController:masterViewController];

    DetailViewController *detailViewController = [[DetailViewController alloc]
        initWithNibName:@"DetailViewController" bundle:nil];
    UINavigationController *detailNavigationController = [[UINavigationController alloc]
        initWithRootViewController:detailViewController];

    masterViewController.detailViewController = detailViewController;

    self.splitViewController = [[UISplitViewController alloc] init];
    self.splitViewController.delegate = detailViewController;
    self.splitViewController.viewControllers = @[masterNavigationController,
        detailNavigationController];
    self.window.rootViewController = self.splitViewController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

- MasterViewController is subclass of TableViewController

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

@class DetailViewController;

@interface MasterViewController : UITableViewController

@property (strong, nonatomic) DetailViewController *detailViewController;

@end
```

- DetailViewController is subclass of ViewController

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

@interface DetailViewController : UIViewController <UISplitViewControllerDelegate>

@property (strong, nonatomic) id detailItem;

@property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabel;
@end
```

- This is a delegate method for UISplitViewController

```
- (void)splitViewController:(UISplitViewController *)splitController willHideViewController:
    (UIViewController *)viewController withBarButtonItem:(UIBarButtonItem *)barButtonItem
    forPopoverController:(UIPopoverController *)popoverController
{
    barButtonItem.title = NSLocalizedString(@"Master", @"Master");
    [self.navigationItem setLeftBarButtonItem:barButtonItem animated:YES];
    self.masterPopoverController = popoverController;
}

- (void)splitViewController:(UISplitViewController *)splitController willShowViewController:
    (UIViewController *)viewController invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    // Called when the view is shown again in the split view, invalidating the button and popover
        controller.
    [self.navigationItem setLeftBarButtonItem:nil animated:YES];
    self.masterPopoverController = nil;
}
```

Practice: Create iOS Project by using Master-Detail Template

Table View

Table View – View in MVC

- Tables display lists of data.
- Each item in a table's list is a row.

MVC in Table View

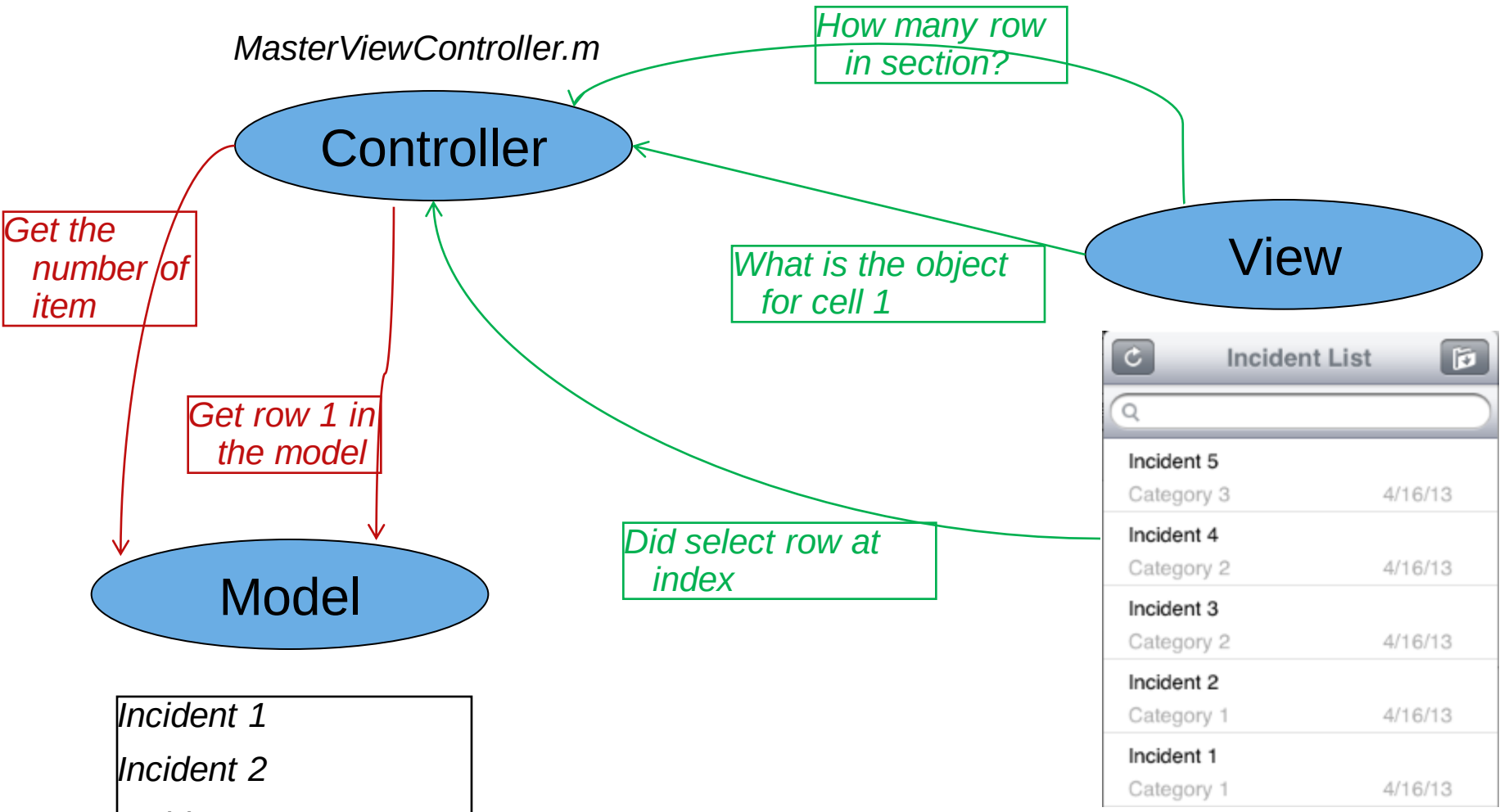
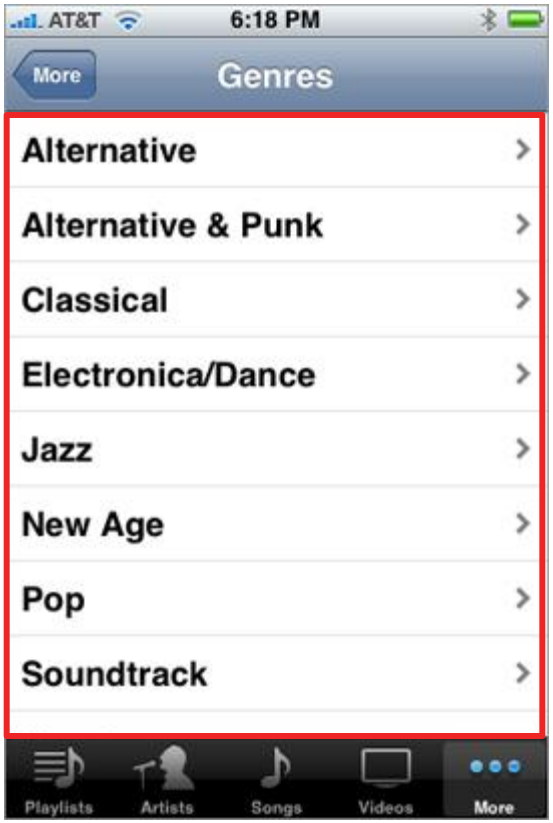


Table View Styles

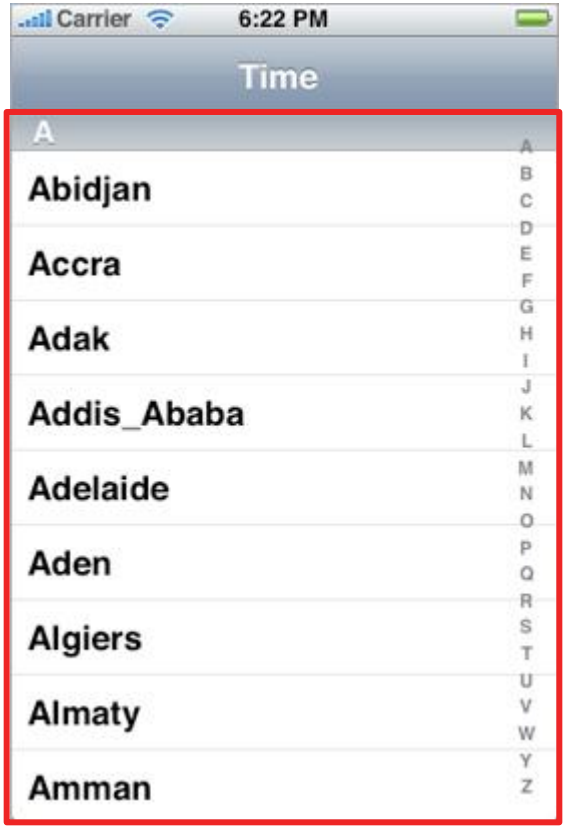
- There are two major styles of table views: plain and grouped

Plain Table Views

A table view in the plain style



A table view configured as an indexed list



A table view configured as a selection list

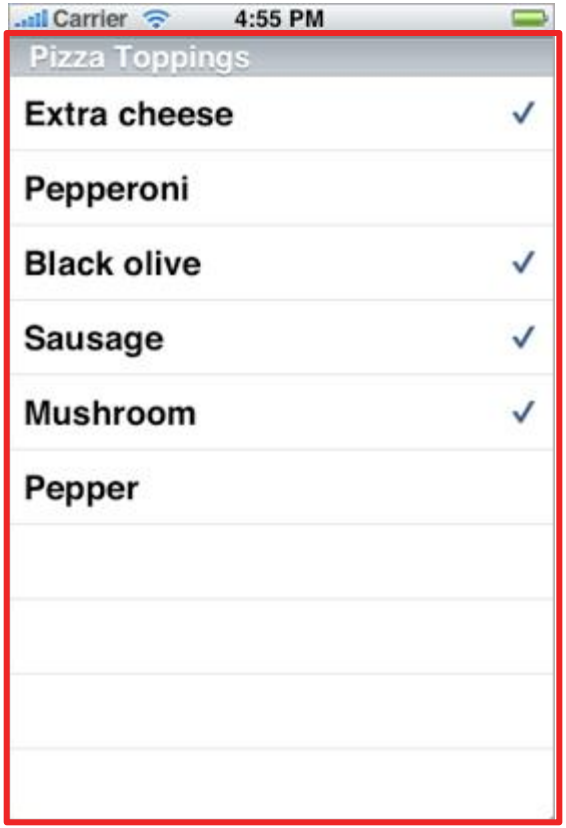
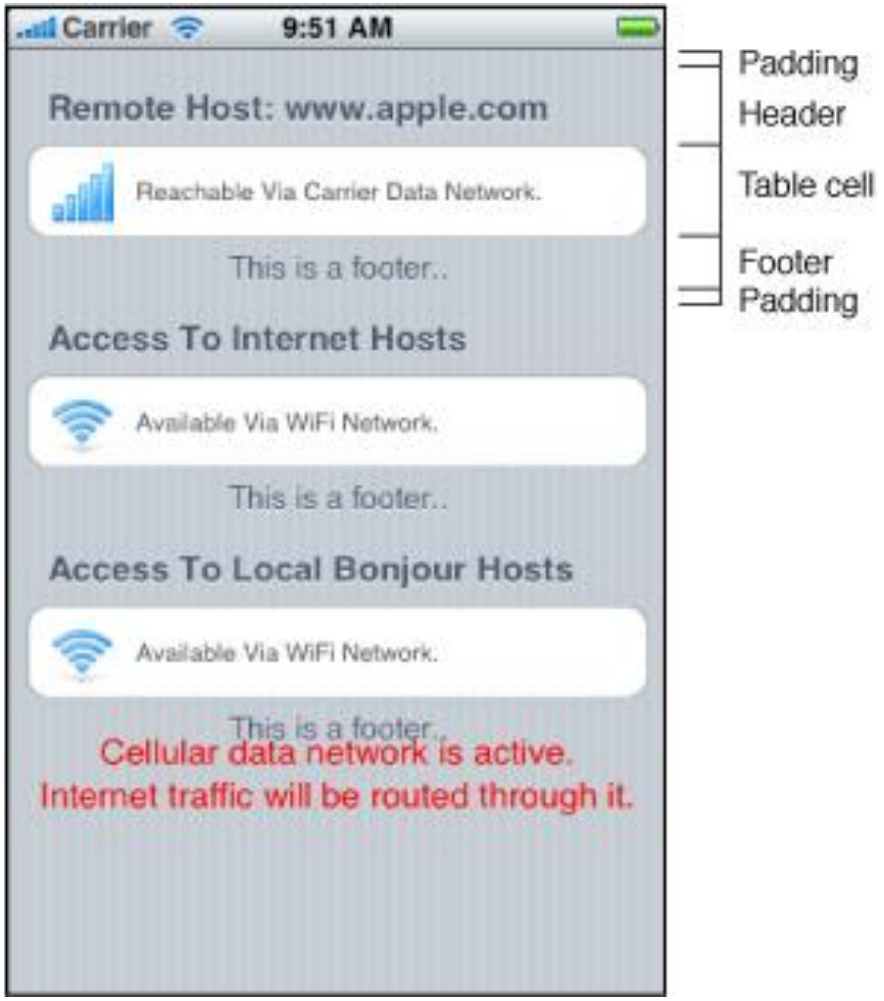
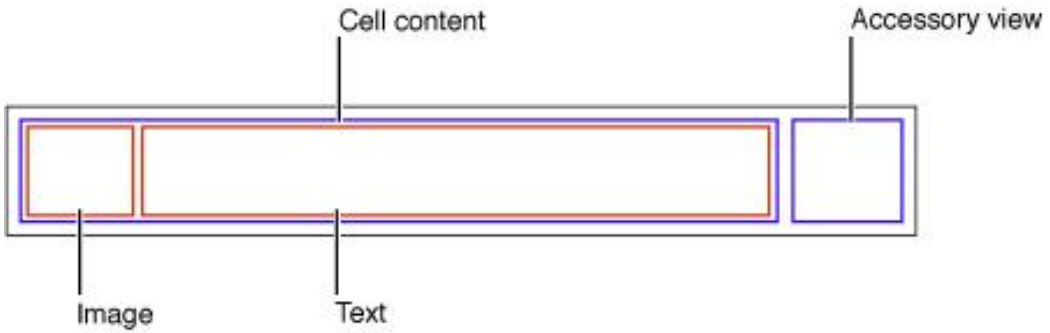


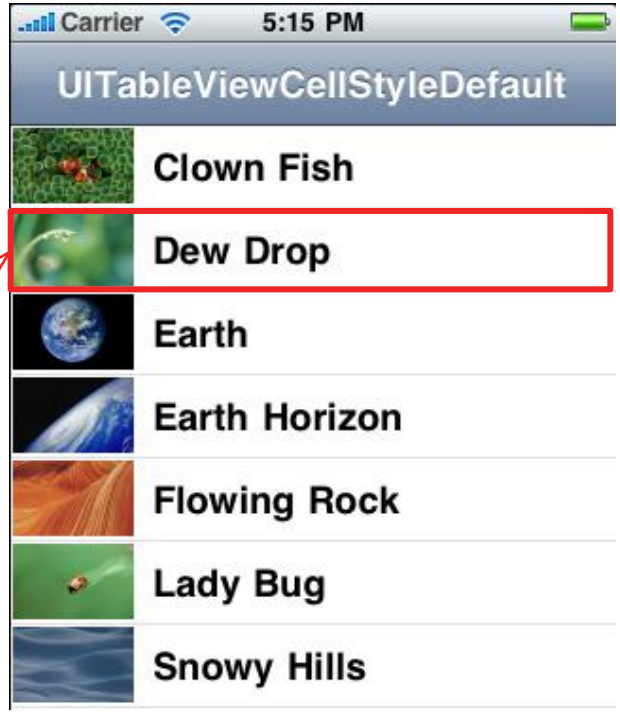
Table View Styles

Grouped Table Views



Standard Styles for Table View Cells





Default table row style

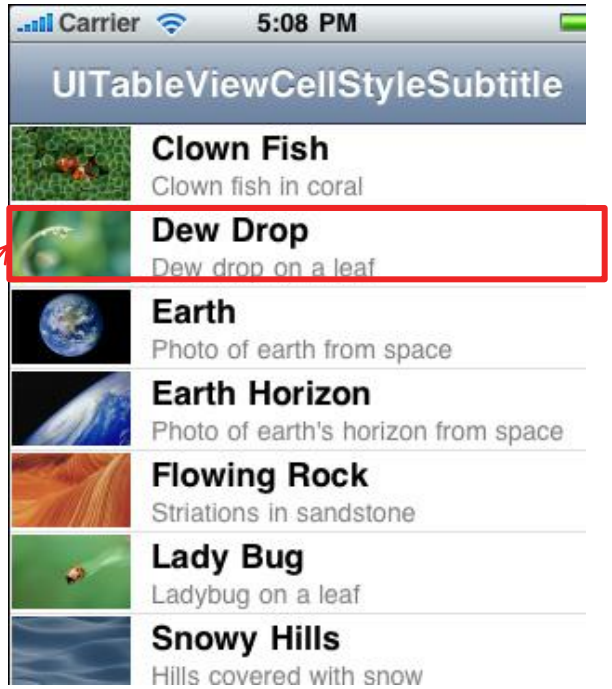


Table row style with a subtitle under the title

Standard Styles for Table View Cells

Table row style with a right-aligned subtitle

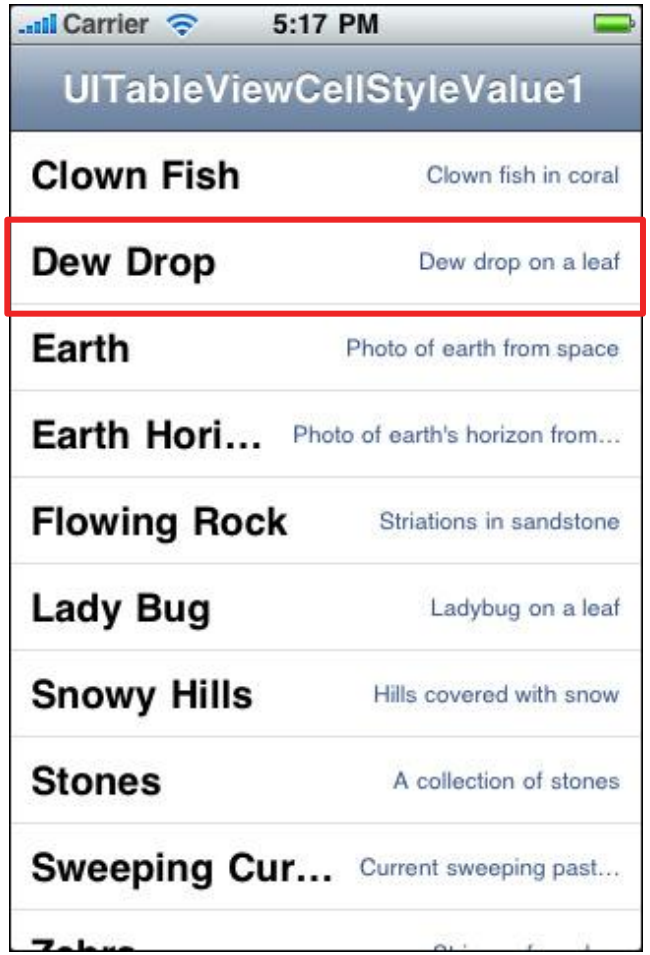
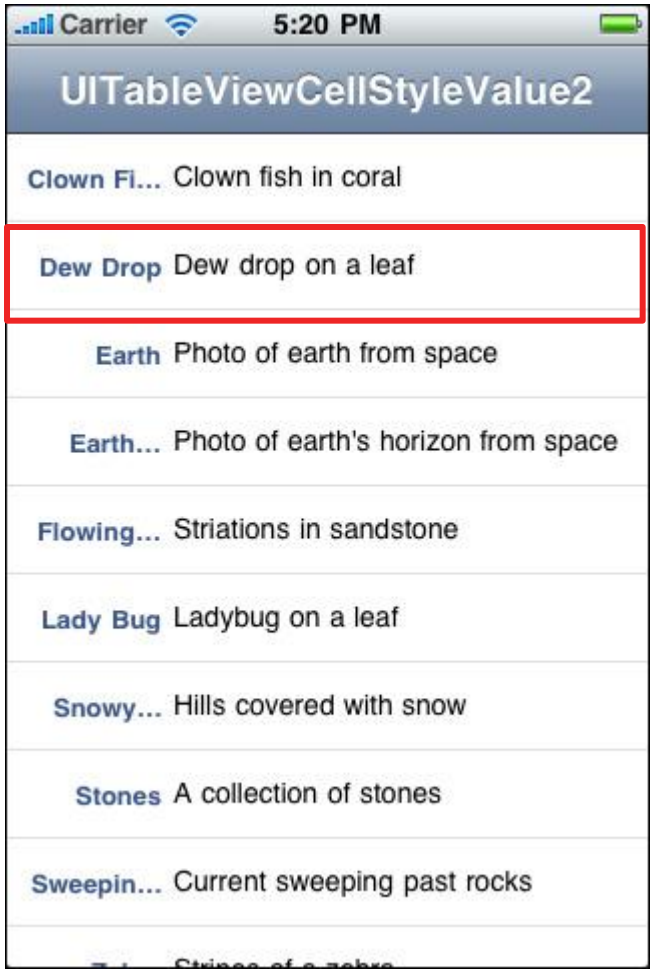
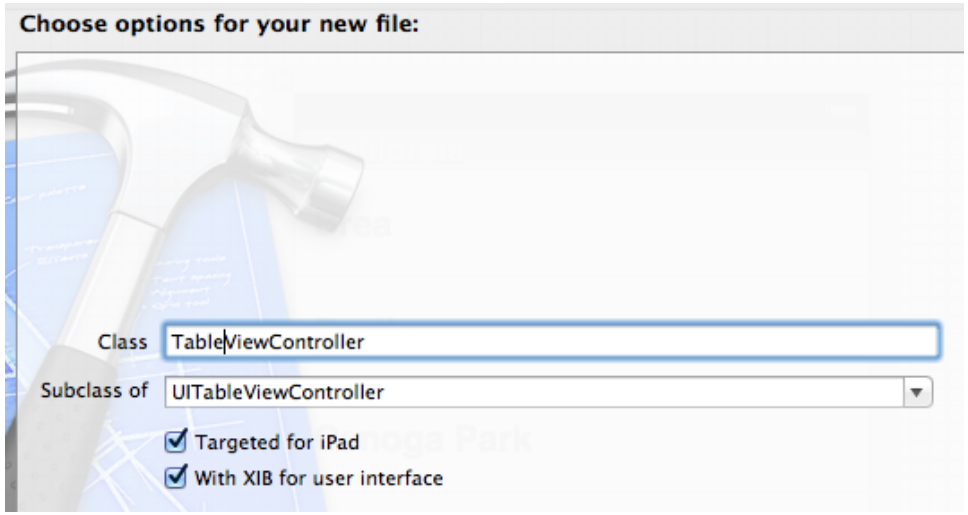


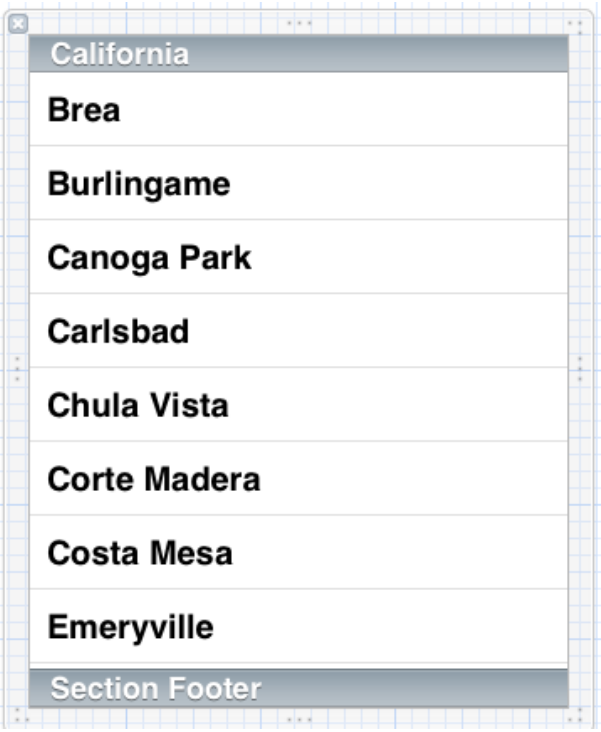
Table row style in Contacts format



Create Table View

Creating a Table View by Interface Builder





Demo: Create table view using Interface Builder

Create Table View

Creating a Table View Programmatically

Adopt the Data Source and Delegate Protocols

```
@interface RootViewController : UIViewController <UITableViewDelegate,
UITableViewDataSource>
```

Create and Configure a Table View

```
- (void)loadView
{
UITableView *tableView = [[UITableView alloc] initWithFrame:[[UIScreen mainScreen]
applicationFrame] style:UITableViewStylePlain];
tableView.autoresizingMask =
UIViewAutoresizingFlexibleHeight|UIViewAutoresizingFlexibleWidth;
tableView.delegate = self;
tableView.dataSource = self;
[tableView reloadData];

self.view = tableView;
}
```

Demo: Create table view programatically

Populating Table View with Data (Controller and Model in MVC)- *MasterViewController.m*

How many section in table view?

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}
```

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return [[DataManager sharedInstance].incidentList count];
}
```

Model

How many row in section?

Populating Table View with Data (Controller and Model in MVC)

- *MasterViewController.m*

What will be show at row n?

```
// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)
    indexPath
{
    static NSString *CellIdentifier = @"Cell";

    IncidentViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        //cell = [[IncidentCell alloc] initWithStyle:UITableViewCellStyleDefault
            reuseIdentifier:CellIdentifier];
        NSArray* views = [[NSBundle mainBundle] loadNibNamed:@"IncidentViewCell" owner:nil
            options:nil];

        for (UIView *view in views) {
            if([view isKindOfClass:[UITableViewCell class]])
            {
                cell = (IncidentViewCell*)view;
            }
        }
    }

    Incident *inc = [[DataManager sharedInstance].incidentList objectAtIndex:indexPath.row];
    cell.nameLbl.text = inc.name;
    cell.categoryLbl.text = inc.category;

    NSString *dateString = [NSDateFormatter localizedStringFromDate:inc.createDate
                                                          dateStyle:NSDateFormatterShortStyle
                                                          timeStyle:NSDateFormatterNoStyle];

    cell.createDateLbl.text = dateString;
    return cell;
}
```

Model

Managing Selections (Controller in MVC)

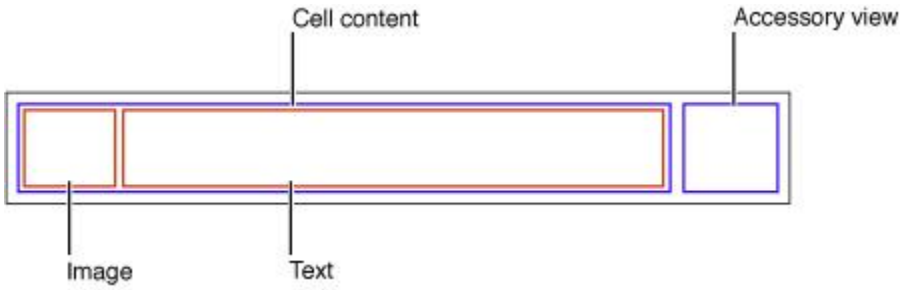
In response to the user tapping a row, an application could do any of the following:

- Show the next level in a data-model hierarchy.
- Show a detail view of an item.
- Show a checkmark in the row to indicate that the represented item is selected.

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{

}
```

Customizing Cells

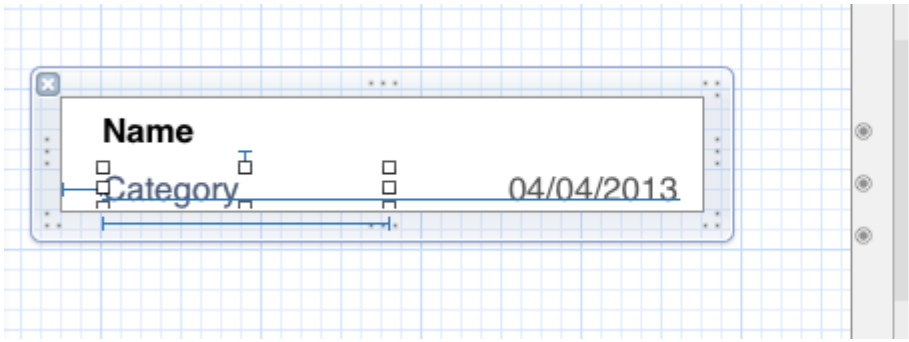
Default cell content in a UITableViewCell object



- Add subviews to a cell's content view.
- Create a custom subclass of UITableViewCell.

Customizing Cells

- Create a custom subclass of UITableViewCell.




```
@interface IncidentViewCell : UITableViewCell

@property (weak, nonatomic) IBOutlet UILabel *nameLbl;
@property (weak, nonatomic) IBOutlet UILabel *categoryLbl;
@property (weak, nonatomic) IBOutlet UILabel *createDateLbl;

@end
```

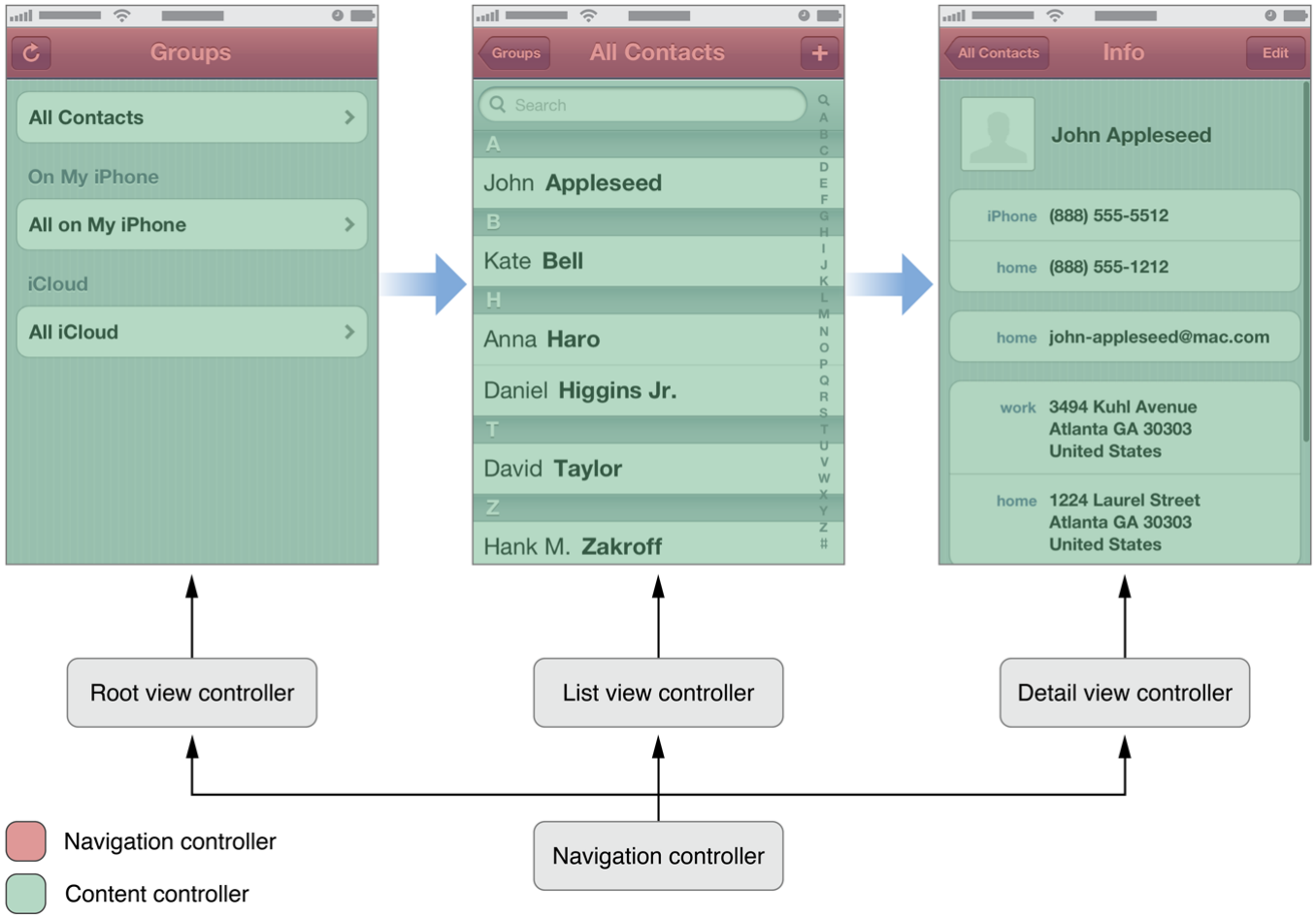
```
// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)
    indexPath
{
    static NSString *CellIdentifier = @"Cell";

    IncidentViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        //cell = [[IncidentCell alloc] initWithStyle:UITableViewCellStyleDefault
            reuseIdentifier:CellIdentifier];
        NSArray* views = [[NSBundle mainBundle] loadNibNamed:@"IncidentViewCell" owner:nil
            options:nil];

        for (UIView *view in views) {
            if([view isKindOfClass:[UITableViewCell class]])
            {
                cell = (IncidentViewCell*)view;
            }
        }
    }
```

Customizing Cells

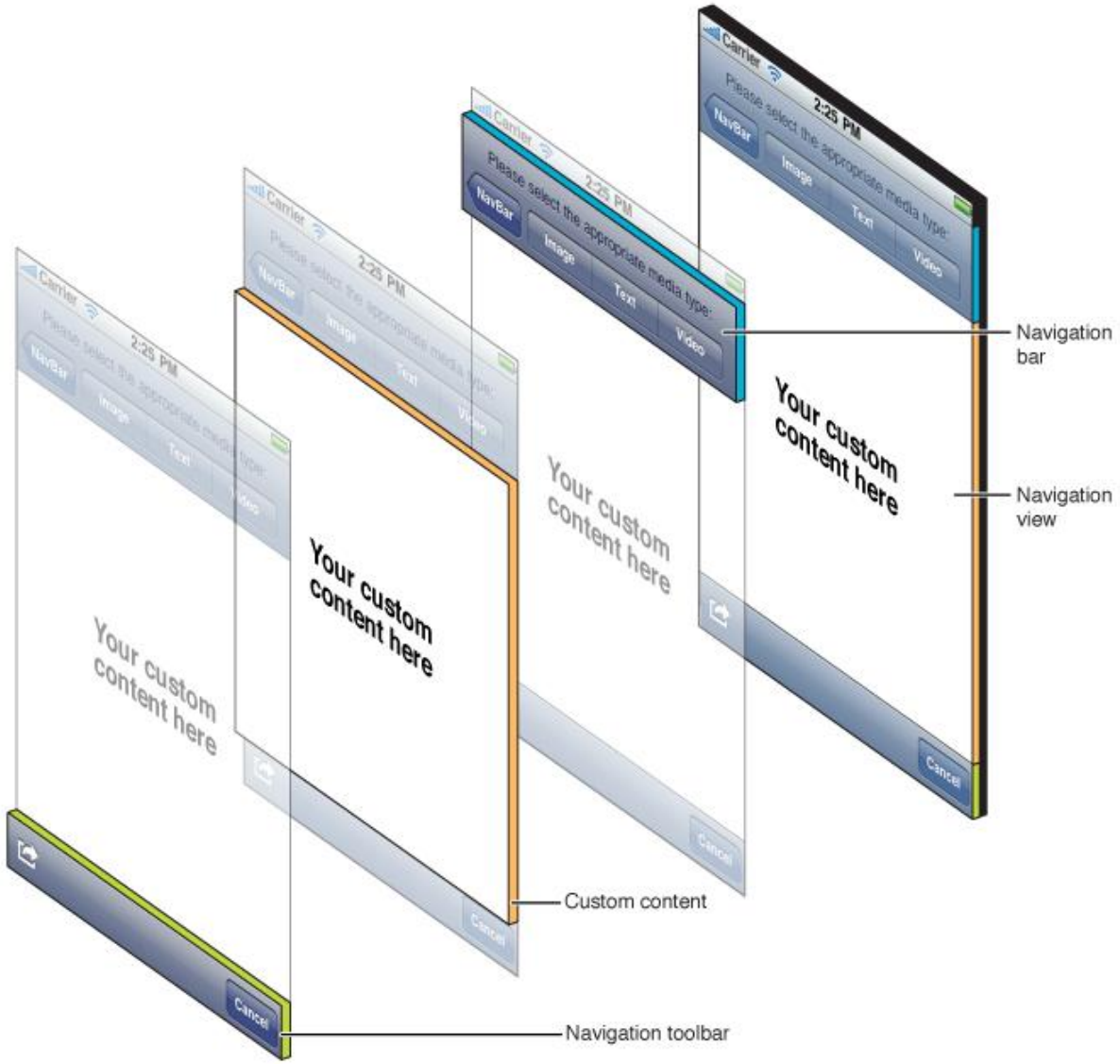
Practice: Create Custom cell and Populating Table View with Data

Navigation Controller

A navigation controller manages a stack of view controllers to provide a drill-down interface for hierarchical content

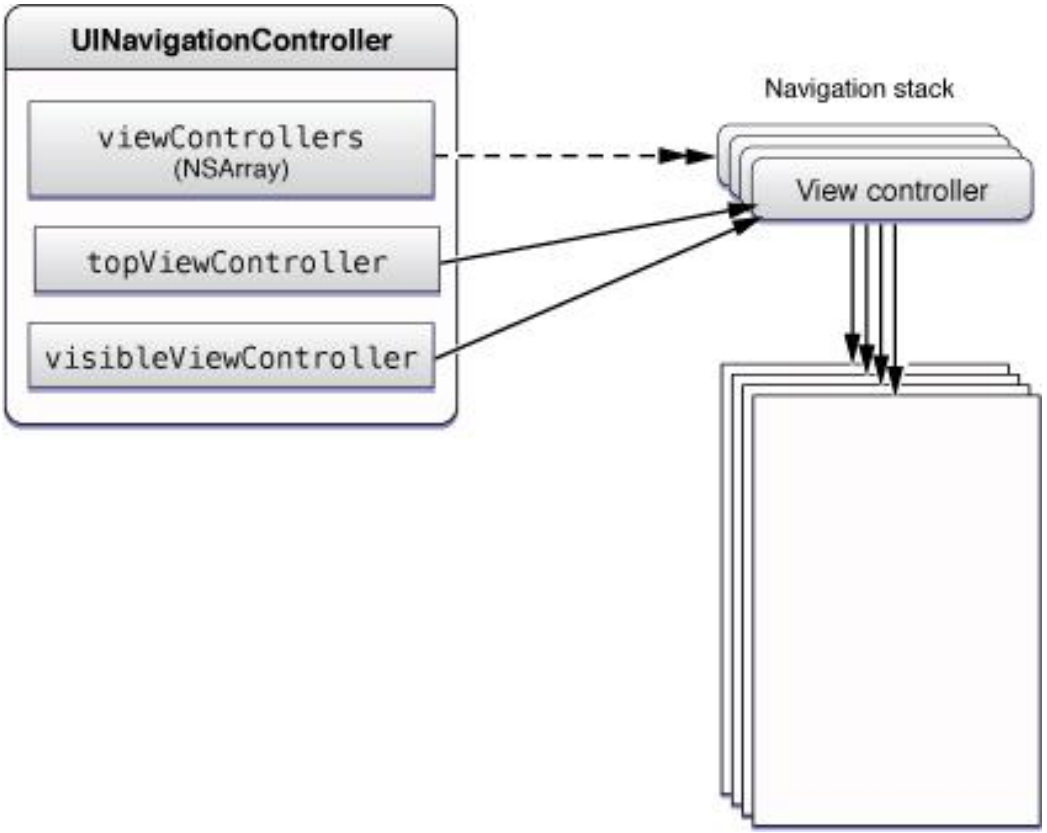


Anatomy of a Navigation Interface



Navigation stack

- The navigation stack is a last-in, first-out collection of custom view controller objects that is managed by the navigation controller.
- The first item added to the stack becomes the **root view controller** and is never popped off the stack



Creating a Navigation Interface Programmatically

- Create the root view controller for the navigation interface.
- Create the navigation controller, initializing it using the initWithRootViewController: method.
- Set the navigation controller as the root view controller of your window (or otherwise present it in your interface).

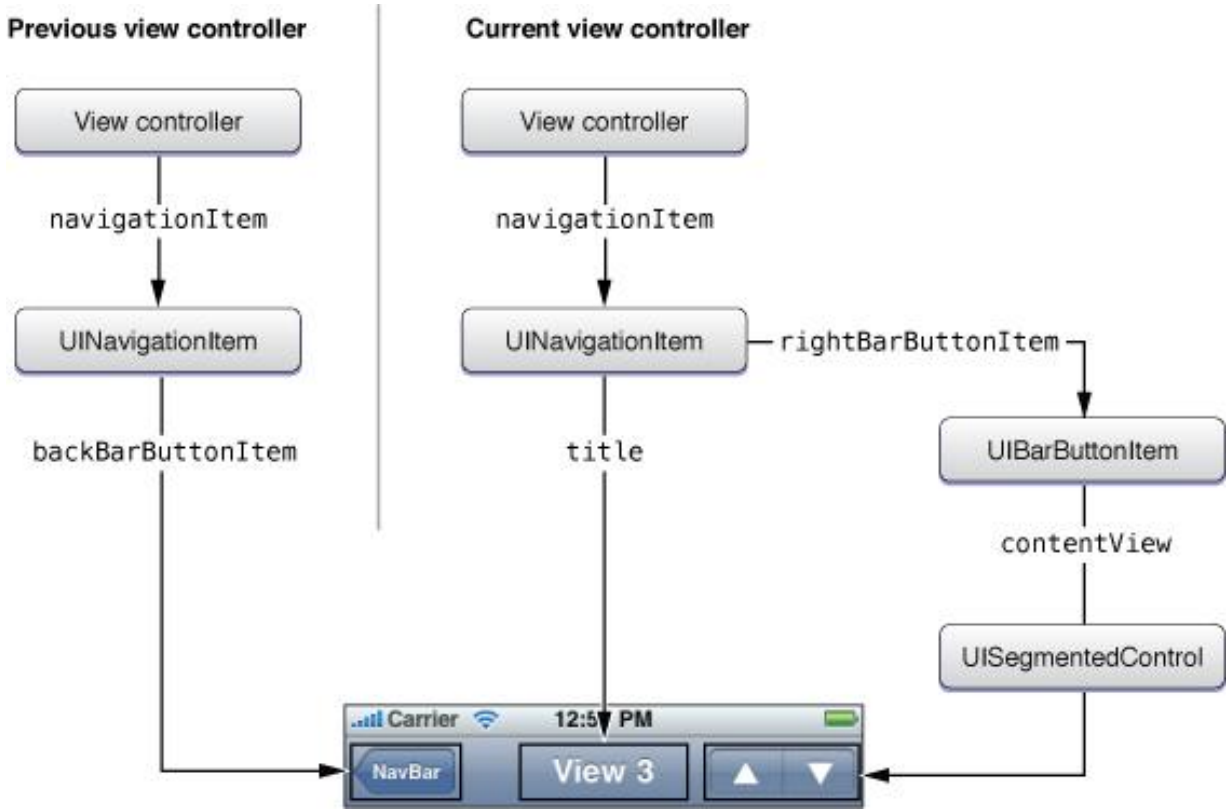
```
//Root View
DetailViewController *detailViewController = [[DetailViewController alloc]
    initWithNibName:@"DetailViewController" bundle:nil];
//Navigation Controller
UINavigationController *detailNavigationController = [[UINavigationController alloc]
    initWithRootViewController:detailViewController];

masterViewController.detailViewController = detailViewController;

self.splitViewController = [[UISplitViewController alloc] init];
self.splitViewController.delegate = detailViewController;
self.splitViewController.viewControllers = @[masterNavigationController,
    detailNavigationController];
```

Managing the Navigation Stack

- Display the next level of hierarchical data.
- Back up one level in the hierarchy.
- Restore the navigation stack to a previous state.
- Back up an arbitrary number of levels in the hierarchy.
- Return the user to the root view controller.

Navigation Bar

- Item positions on a navigation bar
  - Left (*leftBarButtonItem*)
  - Center (*titleView*)
  - Right (*rightBarButtonItem*)



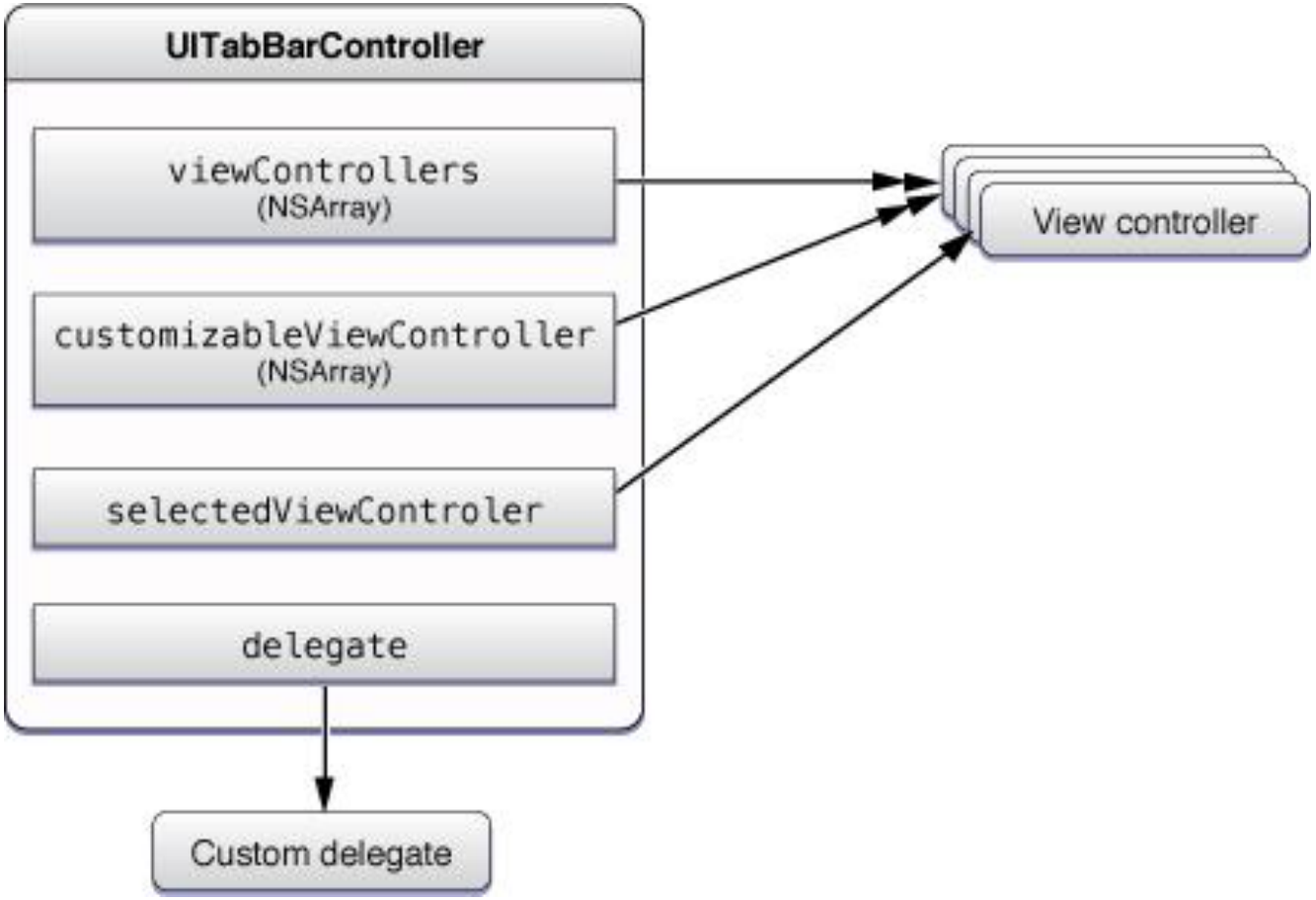
Pratice: Create Adding Incident Screen and Using NavigationController to show the screen.

Tab Bar Controllers

Tab Bar Controllers

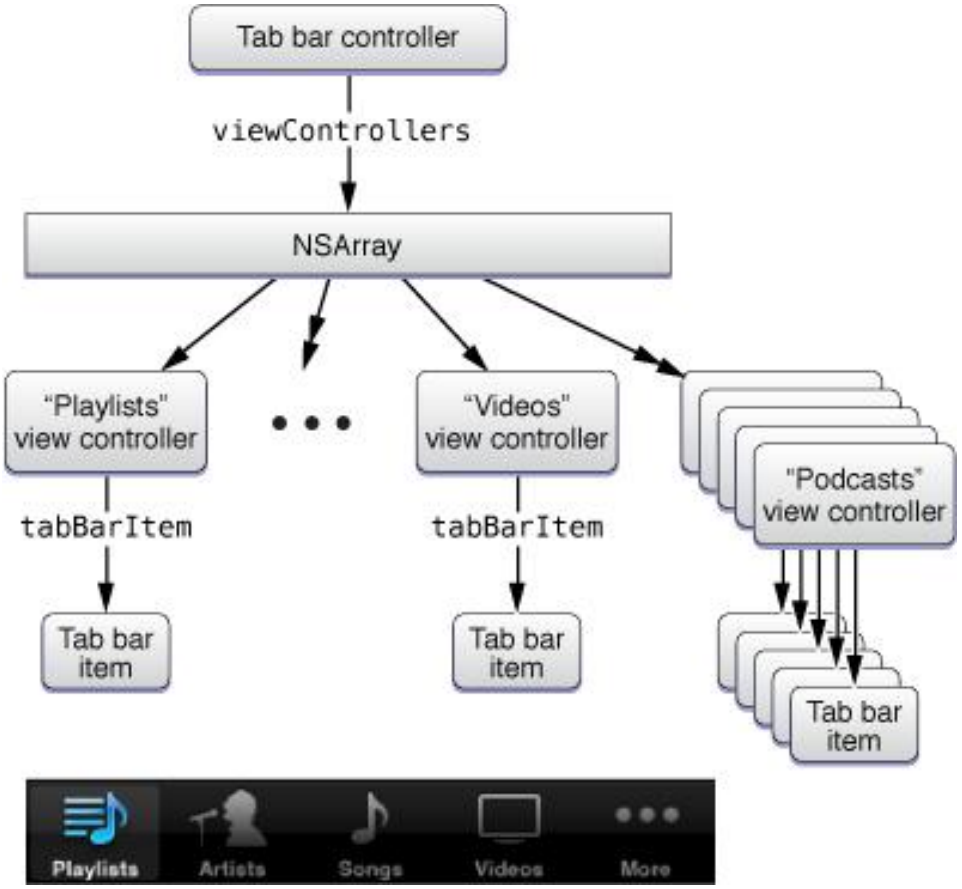
- Anatomy of a Tab Bar Interface

The Objects of a Tab Bar Interface



Tab Bar Controllers

- If you add more than five items to the viewControllers property, the tab bar controller automatically inserts a special view controller (called the More view controller)



Creating a Tab Bar Interface Programmatically

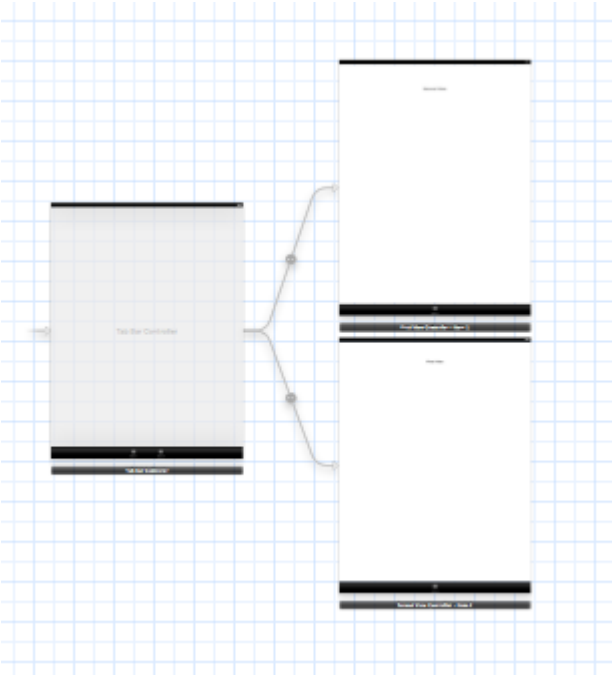
- Create a new UITabBarController object.
- Create a content view controller for each tab.
- Add the view controllers to an array and assign that array to your tab bar controller's viewControllers property.
- Set the tab bar controller as the root view controller of your window (or otherwise present it in your interface).

```
_tabBarController = [[UITabBarController alloc] init];

MyViewController* vc1 = [[MyViewController alloc] init];

MyOtherViewController* vc2 = [[MyOtherViewController alloc] init];

NSArray* controllers = [NSArray arrayWithObjects:vc1, vc2, nil];

_tabBarController.viewControllers = controllers;


window.rootViewController = _tabBarController;
```

Demo and Practice: Create Tab Bar Interface Programmatically

Creating a Tab Bar Interface Using a Storyboard

- Drag a tab bar controller from the library.
- Interface Builder creates a tab bar controller and two view controllers, and it creates relationships between them. These relationships identify each of the newly created view controllers as the view controller for one tab of the tab bar controller.



Demo and Practice: Create Tab Bar Interface Using a Storyboard

Managing Tabs at Runtime

- Adding and Removing Tabs
- Preventing the Selection of Tabs
- Monitoring User-Initiated Tab Changes
- Changing a Tab’s Badge

THANK YOU

- http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/tableview_iphone/AboutTableViewsiPhone/AboutTableViewsiPhone.html#//apple_ref/doc/uid/TP40007451-CH1-SW1
- http://developer.apple.com/library/ios/#documentation/WindowsViews/Conceptual/ViewControllerCatalog/Introduction.html#//apple_ref/doc/uid/TP40011313-CH1-SW1
- http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007072-CH1-SW1


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING