iPhone Programming
CMSC 498I – Fall 2010

View Controllers
Adam Porter
Today’s Topics

• Using “View Controllers”

• View Controller overview

• Application flow – Chaining view controllers together
View Controllers

• View Controllers are a specific type of MVC controller

• Provide more than just MVC coordination
  
  • **View management** – manages screenful of content
  
  • **Application flow** – makes navigation easy
  
  • **User experience** – Implement standard behaviors
  
  • **Override points** – For handling system events
  
  • **Performance** – Designed with mobile device performance in mind
Supporting Classes

- NSObject
  - UIResponder
    - UIView
      - UITabBar
      - UINavigationBar
  - UINavigationItem
  - UIBarButtonItem
  - UITabBarItem
  - UIBarButtonItem
ViewController Types

• Base controllers
  ▪ Manage one screen
  ▪ Usually custom subclasses of:
    ▪ UIViewController
    ▪ UITableViewViewController

• Manager controllers
  ▪ Manage child viewcontrollers
    ▪UITabBarController
    ▪UINavigationViewController
Base Controllers

UIViewController & UITableViewController
UIViewController

- Subclassed for application-specific functionality
Calculator Application

- Multiple subviews contained within a top level view
- Top level view managed by a view controller

Calculator View Controller
Calculator Application

- Multiple subviews contained within a top level view
- Top level view managed by a view controller
UIViewController

• Subclass can implement:
  • Rotation control and events
  • Memory management
    ▪ Low memory warnings
  • Editing behavior
  • Display-related events
  • Interaction with parent container (tab, navigation)
UITableViewController

- Subclass to work with your data
Media Library Example

- Contains a `UITableView`
- Top level view managed by a table view controller
UITableViewController

• Subclass can implement:

• Data source and delegate methods

  ▪ See last week’s lectures
Manager Controllers

UITabBarController & UINavigationController
UITabBarController

• Switches between independent views
• Each subview has its own view controller and view
Clock Application

Tab Bar Controller

View Controller

View Controller

view

Monday, October 4, 2010
UITabBarController

• Typically the top level controller in an application
  ▪ E.g., each tab contains a UIViewController or UINavigationController

• Delegate API to control certain behaviors
  ▪ Selection
  ▪ Customization

• Customizable components in tab bar
  ▪ Tab ordering
  ▪ Automatic “More” list creation for overflow
UITabBarItem

• Visual items on tab bar

• Creation

  • Using system items – (UITabBarItemSystemItemFavorites, Contacts, ...)

  ![System Items](image)

  • Using any UIImage

    • UITabBar automatically creates gray, “shiny” blue versions

• Badges

![Badge](image)
UINavigationController

• Hierarchical organization – push / pop screens
Email Application

Navigation Bar

Top Controller’s view

Toolbar
**UI@show_day{Navigation}Controller**

- Manages hierarchal application flow

- Navigation Stack
  - **Root Controller** - bottom of the stack
  - **Top Controller** - currently show view controller

- Customizable components in navigation / toolbar

- Dynamic per-node toolbar
  - TabBar is fixed and does not change with each selection
**UINavigationItem**

- Each UIViewController has a navigationItem
- Specifies what is displayed in navigation bar
  - UINavigationController consults this to configure its display
- Can customize Title, Left / Right buttons
- By default navigation bar contains a centered title and a back button on the left
  - Title automatically configured to match view controller’s title
  - Title also used on next level’s back button
UIBarButtonItem

• Used in UIBarButtonItem and UIBarButtonItem

• Creation
  ▪ With any custom view
  ▪ With buttons
    ▪ System Items – (e.g. UIBarButtonItemSystemItemEdit, Compose, ...)
    ▪ Using a UIImage
    ▪ With an NSString title

• Configurable target, action for buttons
Navigation Bar

- Views provided by UIBarButtonItem's view property
- Title provided as string or custom view
Toolbar

- Views provided using `UIBarButtonItem`'s view property
Mini Summary...

- **UITabBarController** and **UINavigationController**
  - Switch between independent views
  - Hierarchal navigation

- Not meant for subclassing
  - Use delegate to customize

```objective-c
// UITabBarController
@property(nonatomic,assign) id<UITabBarControllerDelegate> delegate;

// UINavigationController
@property(nonatomic, nonatomic, assign) id<UINavigationControllerDelegate> delegate;
```
Combinations

Some Standard Configurations
Tab & Navigation

- **UITabBarController** top level item
  - Enough items for “more” button bar item to automatically be created

- Each tab has a **UINavigationController**

- Each **UINavigationController** manages a chain of **UITableViewControllers**
ToolBar & Navigation

- Top level item is a UINavigationController

- UINavigationController has a toolbar at bottom (UIToolbar)

- UINavigationController manages a chain of UITableViewController

- Leaf view controller is a UIViewController managing a custom view that displays email text
Subclassing

Custom View Controllers
Subclassing

• Choose view controller to subclass

```objectivec
@interface PersonViewController : UIViewController {

@interface PersonViewController : UIViewController {

- (id)initWithPerson:(Person *)person;

@property (retain, readwrite) Person *person;

@end
```

• Implement custom methods for passing data

```objectivec
- (id)initWithPerson:(Person *)person;

@property (retain, readwrite) Person *person;
```

• Hook up IBOutlets and implement required action methods
Lazy Loading

• If a UIViewController’s view property is nil when you access it, -loadView: is called

- (UIView *)view {
  if (!view_) {
    // load the “view_”
    [self loadView];
  }
  return view_;
}
Loading Views

• When -loadView is called

• If -loadView: is overridden, you must create all views and assign a non-nil value to the view controller’s view property
  
  ▪ Otherwise default -loadView: method loads the nib file indicated by the view controller’s nibName and nibBundle properties
  
  ▪ If that nib file is not found, -loadView: looks for a nib file whose name matches the name of the view controller class and loads that file
  
  ▪ If that nib file is not found, it creates an empty UIView object

• Finally, the view controller calls -viewDidLoad:
Subclassing

Standard Overrides
Subclassing - Overrides

• `viewWillAppear:`
  - Perform tasks associated with presenting your view
    - E.g. update a display value, coordinate status bar look, etc.

• `viewDidAppear:`
  - Called when the view has been fully transitioned onto the screen
    - E.g. `UITableView` flashes its scroll bars
Subclassing - Overrides

• -viewDidDisappear:
  ▪ Your view is dismissed, covered or hidden
  ▪ Should stop timers, display updates, etc...

• -viewWillDisappear:
  ▪ May want to stop timers and updates at this point
  ▪ May want to save state to disk
Subclassing Overrides

- **viewDidLoad**
  - Called after a view is loaded from XIB, or by `-loadView`

- **viewDidUnload**
  - Called whenever UIKit deems it necessary to release the view without deallocating the controller
  - You should release any data that is easily recreated
  - If your data is expensive to recreate (e.g., cache contents), maybe wait until `-didReceiveMemoryWarning` to release
**Reading**

- “View Controller Programming Guide For iPhone OS”
  - ViewControllerPGforiPhoneOS.pdf
  - Read “About View Controllers” – p.11 - 20
  - Read “Custom View Controllers” – p.21 - 52

- iPhone OS Reference Library
  - UIViewController Class Reference