Announcements

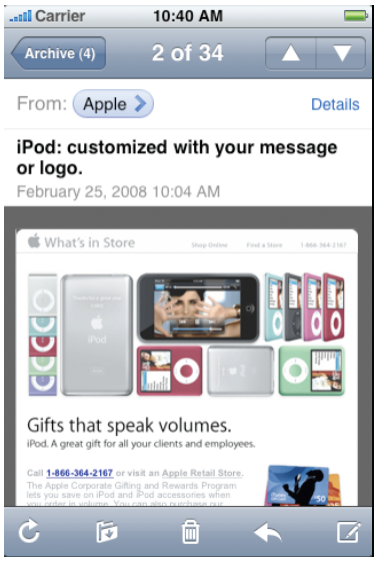
- **Lab 2 is due tonight by 11:59 PM**
  - wustl email SPAM filter prevents sending .zip files
    - If sending from wustl account, remove the .zip extension before adding attachment, the TAs will add it back to fix the issue
- **Lab 3 is posted**
  - Due on Monday Feb 23rd
  - Start early on this lab, I will not provide any extensions

Today's Topics

- **Designing iPhone Applications**
- **Model-View-Controller (Why and How?)**
- **View Controllers**

Designing iPhone Applications

Two Flavors of Mail



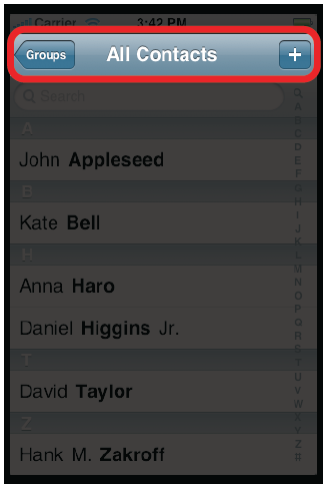
Organizing Content

Organinzing Content

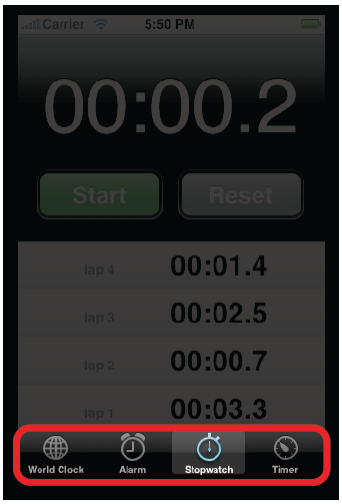
- Focus on your user's data
- One thing at a time
- Screenfuls of content

Patterns for Organizing Content

Navigation Bar

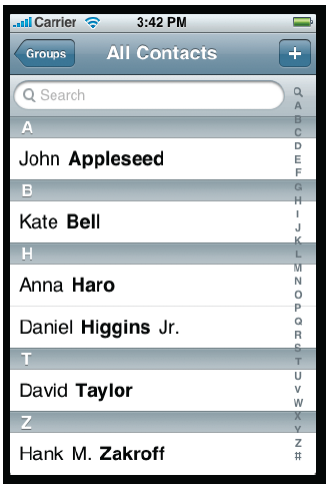


Tab Bar



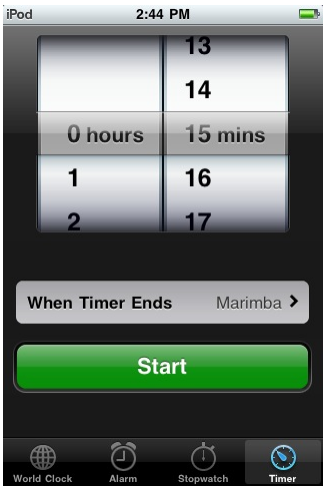
Navigation Bar

- Hierarchy of content
- Drill down into greater detail



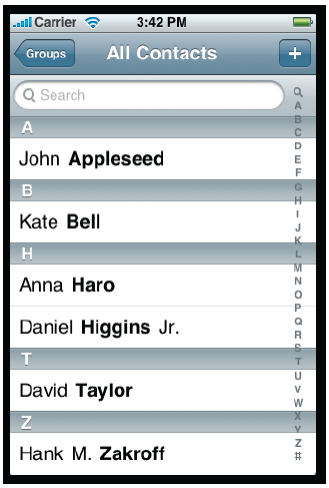
Tab Bar

- Self-contained modes

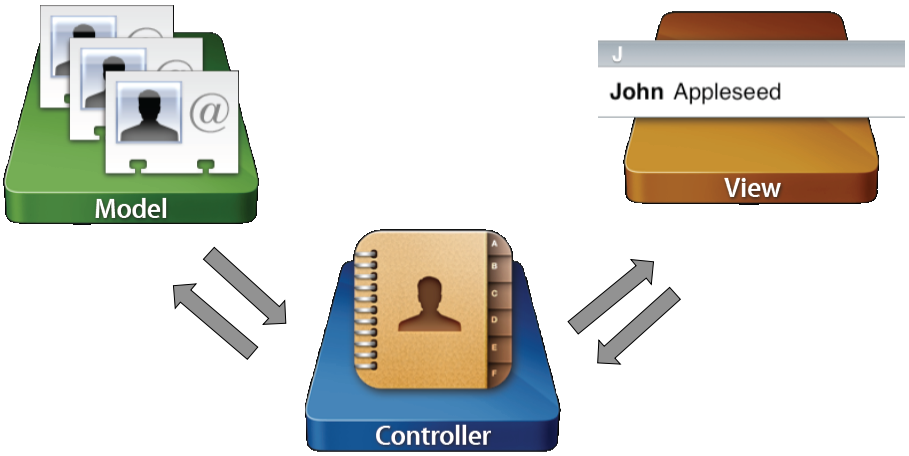


A Screenful of Content

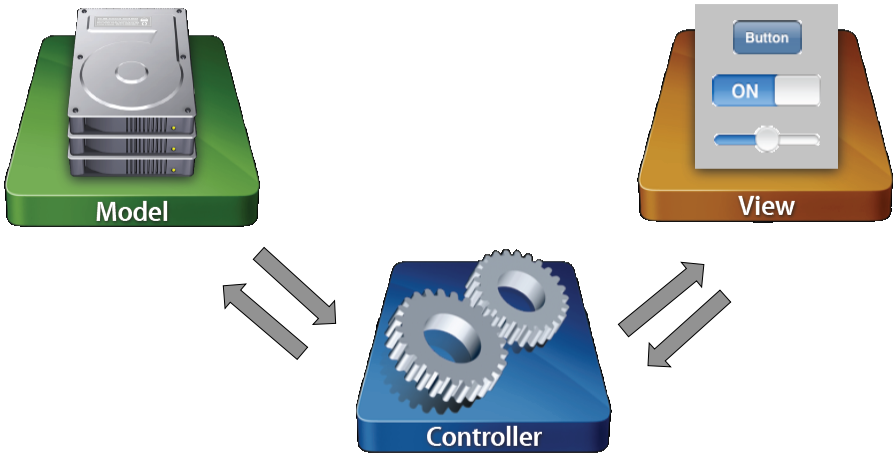
- Slice of your application
- Views, data, logic



Parts of a Screenful



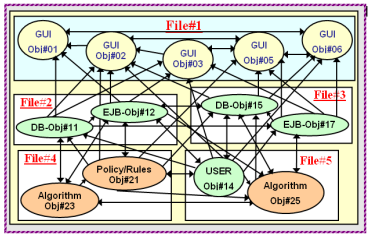
Parts of a Screenful



Model-View-Controller (Why and How?)

Why Model-View-Controller?

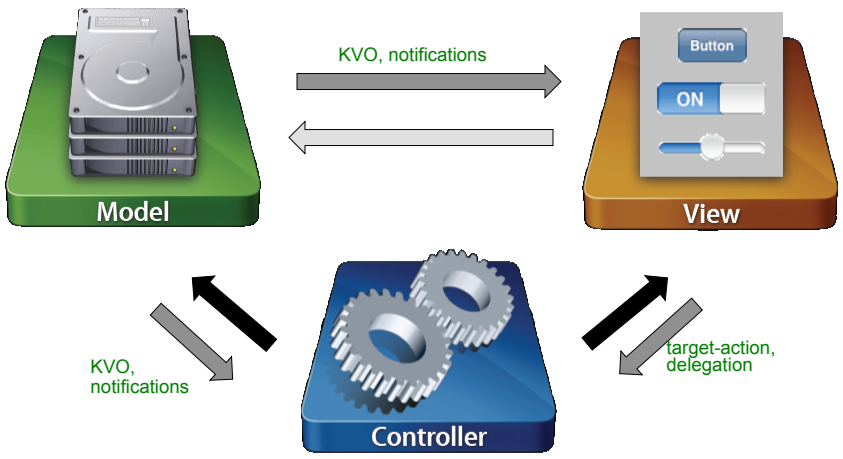
- Ever used the word "spaghetti" to describe code?
- Clear responsibilities make things easier to maintain
- Avoid having one monster class that does everything



Why Model-View-Controller?

- Separating responsibilities also leads to reusability
- By minimizing dependencies, you can take a model or view class you've already written and use it elsewhere
- Think of ways to write fewer lines of code

Communication and MVC



View Controllers

Problem: Managing a Screenful

- Controller manages views, data and application logic
- Apps are made up of many of these
- Would be nice to have a well-defined starting point
  - A la UIView for views
  - Common language for talking about controllers

Problem: Building Typical Apps

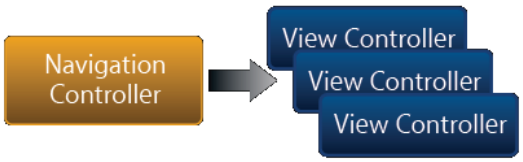
- **Some application flows are very common**
  - Navigation-based
  - Tab bar-based
  - Combine the two
- **Don't reinvent the wheel**
- **Plug individual screens together to build an app**

UIViewController

- **Basic building block**
- **Manages a screenful of content**
- **Subclass to add your application logic**

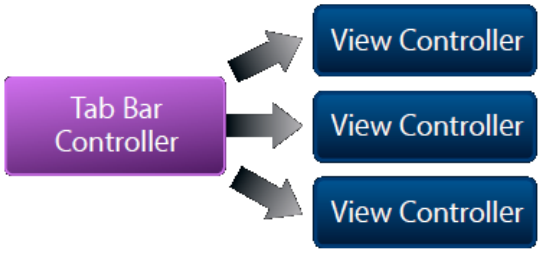
"Your" and Apple View Controllers

- Create your own UIViewController subclass for each screenful
- Plug them together using existing composite view controllers



"Your" and "Our" View Controllers

- Create your own UIViewController subclass for each screenful
- Plug them together using existing composite view controllers



Your View Controller Subclass

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

@interface MyViewController : UIViewController {
   // A view controller will usually
   // manage views and data
  NSMutableArray *myData;
  UILabel *myLabel;
}

//Expose some of its contents to clients
@property NSArray *myData;

// And respond to actions
 - (IBAction) doSomeAction:(id)sender;
```


The "View" in "View Controller"

- **UIViewController superclass has a view property**
  - @property (retain) UIView *view;
- **Loads lazily**
  - On demand when requested
    - OS figures this out
  - Can be purged on demand as well (low memory)
- **Sizing and positioning the view?**
  - Depends on where it's being used
  - Don't make assumptions, be flexible

When to call -loadView?

- **Don't do it!**
- **Cocoa tends to embrace a lazy philosophy**
  - Call -release instead of -dealloc
  - Call -setNeedsDisplay instead of -drawRect:
- **Allows work to be deferred**
  - Performance!
    - Consider time to launching an application

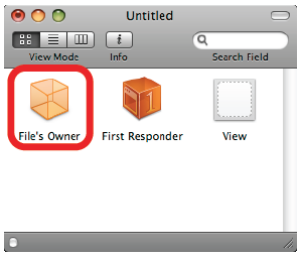
Creating Your View in Code

- **Override -loadView**
  - Never call this directly
- **Create your views**
- **Set the view property**
- **Create view controller with -init**

```
// Subclass of UIViewController
-(void)loadView
{
    MyView *myView = [[MyView alloc] initWithFrame:frame];
    self.view = myView; // The view controller now owns the view
    [myView release]; // With ARC we no longer need this…
}
```

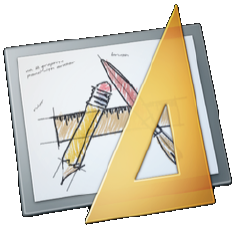
Creating Your View with Storyboard

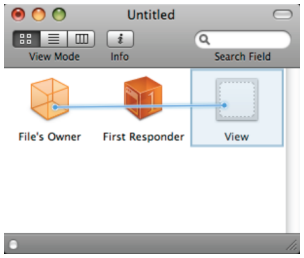
- Lay out a view in Storyboard
- File's owner is view controller class



Creating Your View with Storyboard

- Lay out a view Storyboard
- File's owner is view controller class
- Hook up view outlet
- Create view controller with -initWithNibName:bundle:





View Controller Lifecycle

```
- (id)initWithNibName:(NSString *)nibName
bundle:(NSBundle *)bundle
{
    if (self == [super init...]) {
      // Perform initial setup, nothing view-related
      myData = [[NSMutableArray alloc] init];
      self.title = @“Foo”;
    }
    return self;
}
```


View Controller Lifecycle

```
- (void)viewDidLoad
{
    // Your view has been loaded
  // Customize it here if needed

view.someWeirdProperty = YES;
}
```

View Controller Lifecycle

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    // Your view is about to show on the screen
    [self beginLoadingDataFromTheWeb];
    [self startShowingLoadingProgress];
}
```


View Controller Lifecycle

```
- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
    // Your view is about to leave the screen
    [self rememberScrollPosition];
    [self saveDataToDisk];
}
```

Navigation View Controller Demo

Loading and Saving Data

- **Lots of options out there, depends on what you need**
  - NSUserDefaults
  - Property lists
  - SQLite
  - Web services
- **Covering in greater depth in a few weeks**

Saving State Across App Launches

- **NSUserDefaults to read and write prefs & state**
- **Singleton object:**
  ```
  + (NSUserDefaults *)standardUserDefaults;
  ```
- **Methods for storing & fetching common types:**
  ```
  -(NSInteger)integerForKey:(NSString *)key;
  -(void)setInteger:(int)value forKey:(NSString *)key;

  -(id)objectForKey:(NSString *)key;
  -(void)setObject:(int)value forKey:(NSString *)key;•
  ```
- **Find an appropriate time to store and restore your state**

More View Controller Hooks

- Automatically rotating your user interface
- Low memory warnings

Supporting Interface Rotation

```
-(BOOL)shouldAutorotateToInterfaceOrientation:
            (UIInterfaceOrientation)interfaceOrientation
{

  // This view controller only supports portraits
    return (interfaceOrientation == UIInterfaceOrientationPortrait);

}
```

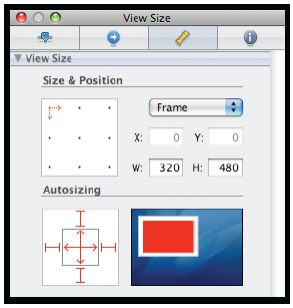
Supporting Interface Rotation

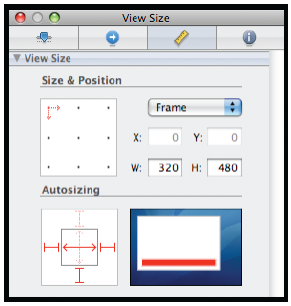
```
-(BOOL)shouldAutorotateToInterfaceOrientation:
            (UIInterfaceOrientation)interfaceOrientation
{

  // This view controller supports all orientations
  // except for upside-down.
    return (interfaceOrientation !=UIInterfaceOrientationPortraitUpsideDown);

}
```


Autoresizing Your Views

```
view.autoresizingMask = UIViewAutoresizingFlexibleWidth |
                        UIViewAutoresizingFlexibleHeight;
```

```
view.autoresizingMask = UIViewAutoresizingFlexibleWidth |
                        UIViewAutoresizingFlexibleTopMargin;
```





NSUserDefaults and Screen Orientation Demo

Lab 3 Demo