iOS Foundations II
Day 2

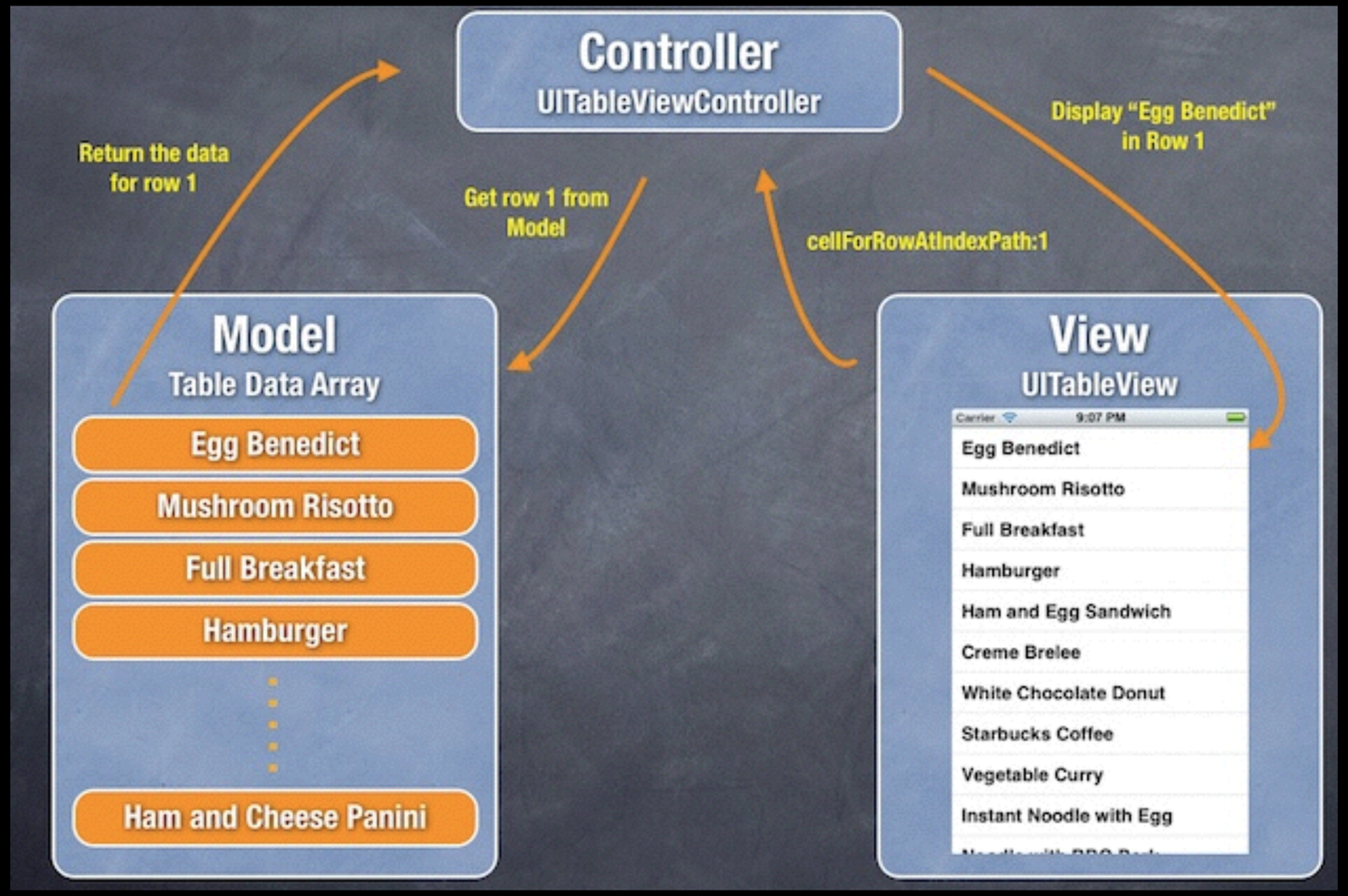
- Homework Review
- Design Patterns/MVC
- View Controllers
- View Hierarchy
- More Git!

iOS Design Patterns

- “A software design pattern is a general reusable solution to commonly occurring problems within a given context in software design”
- No matter what kind of app it is, there are a few fundamental design pattern techniques that all apps use.
- The most important design patterns to know for iOS development:
  - **Model View Controller - MVC - Governs the overall structure of your app.**
  - Delegation - Facilitates the transfer of data and responsibilities from one object to another.
  - Target-Action - Translates user interactions into code your app can execute.
  - Closures/Blocks - Use these for callbacks and asynchronous code.

Design Patterns Are Cool

- A benefit of the universal usage of design patterns is commonality between all apps besides just the language they are programmed in.
- If someone is showing you the source code of their app, you can ask things like “What kind of model classes are you using?” or “How many View Controllers do you have?” and sound like you know what you are talking about.



MVC
(Model-View-Controller)

MVC Facts

- Introduced in the 70's with the Smalltalk programming language.
- Didn't become a popular concept until the late 80's
- The MVC pattern has spawned many evolutions of itself, like MVVM (Model-View-ViewModel)
- MVC is very popular with web design and applications. It's not just for mobile or desktop.

So what is MVC?

- MVC is simply the separation of **M**odel, **V**iew, and **C**ontroller.
- It is a separation of concerns for your code. Being able to separate out these components makes your code easier to read, re-use, test, think about, and discuss.
- The **Model layer** is the data of your app, the **View layer** is anything the user sees and interacts with, and the **Controller layer** mediates between the two.
- Examples of bad MVC practices: Your model classes directly communicating with your views, your view classes making network calls, etc

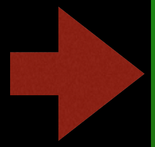
The MVC layout of our App

Model Layer

Person

Controller Layer

PersonTable ViewController



PersonDetail ViewController

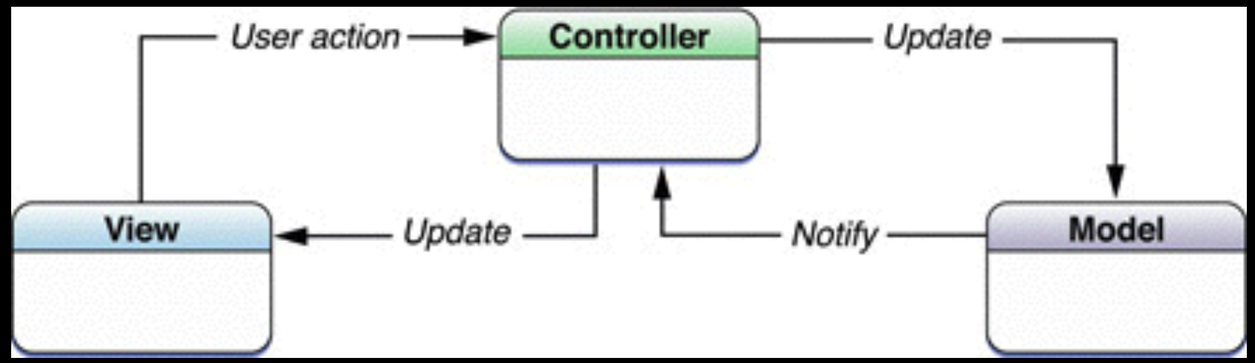
View Layer

Storyboard

TableView

PersonCell

MVC or MCV LOL?



Some people joke its more like MCV, because the controller is the middle man so the C should go in the middle*

*Classic programming joke

Model Layer

- Model objects encapsulate data and logic that are going to be used by your application.
- The Twitter App has a Tweet model class, a User model class, a Favorite model class, etc.

View Layer

- A View object is an object the user can see and possibly interact with.
- Enables displaying and editing of the app's data.
- Communication between the View and Model layers is made possible by…..

Controller Layer

- Act as the intermediary between the model layer and view layer.
- The most common form of a controller in iOS is a view controller.
- Another common controller is a network controller.
- At first your view controllers will have a lot of code. Eventually you should strive to make them lighter so its easier to understand what they are doing at a glance.

Demo

ViewControllers

- Basic Visual Component of iOS.
- For the most part, every ‘screen’ in an app is a separate view controller, although it is possible to have more than one ViewController on the screen at once.
- They are the link between your model layer and view layer.
- iOS provides many built-in ViewControllers to support standard interfaces.
- The most important property of a ViewController is it’s view property. Every ViewController has a root view so it can display your interface objects. You can access this view inside of a view controller with self.view

ViewControllers

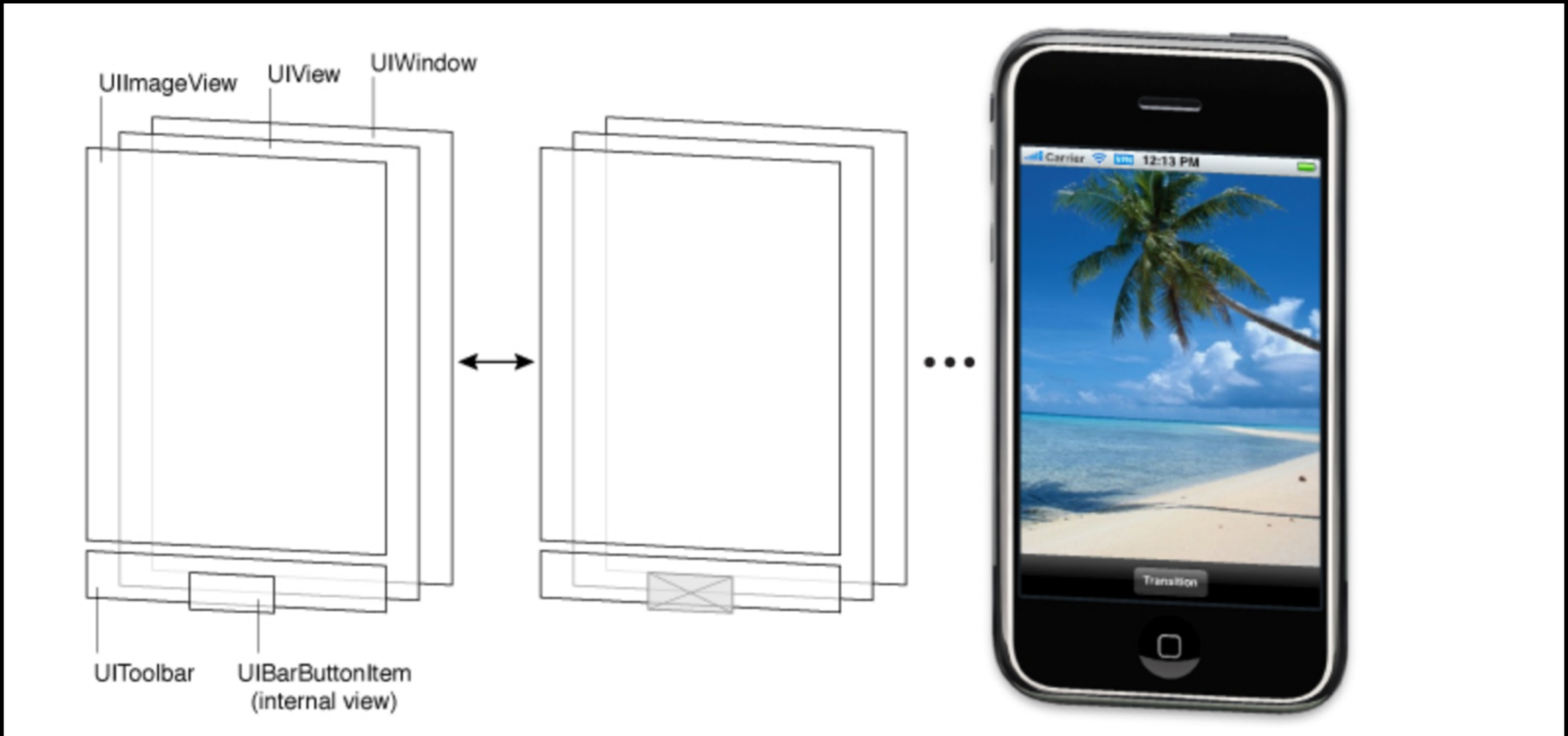
- Really only 4 ways to get a ViewController's view on screen:
    1. Make the ViewController a window's rootViewController (Use that arrow in storyboard to denote the rootViewController!)
    2. Make the ViewController a child of a container View Controller
    3. Show the ViewController in a popover
    4. Present it from another View Controller (The most common scenario)

ViewController Life Cycle

- **The concept of ViewController's lifecycle is important to understand. So important I bolded it.**
- There are 5 methods to know:
    1. ViewDidLoad - Called when the ViewController's view is loaded into memory
    2. ViewWillAppear - Called before the ViewController's view appears onscreen
    3. ViewDidAppear - Called immediately after the ViewController's view has appeared onscreen
    4. ViewWillDisappear - Called when the ViewController's view is about to be removed from the view hierarchy
    5. loadView - Implement this method when you aren't using a storyboard or nib file. This gives you the opportunity to create your own view, configure and add subviews to it, then you can assign to self.view.

Demo

Views



- “A view is an instance of the `UIView` class (or one of its subclasses) and manages a rectangular area in your application window”
- “Views are responsible for drawing content, handling multitouch events, and managing the layout of any subviews. “

Views

- To get onscreen, a UIView must be added as a subview of a parent view. All views can be a child view of a parent view, and all views can have children views of their own. This is called the view hierarchy.
- A UIView has many properties that dictate its appearance: backgroundColor, alpha, hidden, opaque, tintColor, layer, clipsToBounds, etc.
- But the most important property of a UIView is the frame.
- The frame of a view describes the view's location and size relative to its super view.

Frame and CGRect

- The type of the frame property is CGRect.
- A CGRect type has 4 values: X, Y, Width, Height.
- The X and Y values dictate where the view is going to be relative to the view's superview.
- The Width and Height dictate the size of the view.

Demo

Segues

- Segues are provided by the storyboard to help you easily transition from one view controller to another.
- There are 2 primary segues that are used : Show and Present
- Show refers to pushing a view controller onto the navigation stack, it usually will slide in from the right or left.
- Present refers to modally presenting a view controller, it usually slides up from the bottom.
- You can create your own custom segue to customize the behavior to match your needs.

Segues

- 2 primary methods for dealing with segues in code:
    1. performSegueWithIdentifier:
    2. prepareForSegue:sender:

performSegueWithIdentifier

- Call this method in your View Controller to trigger a segue in code.
- If you your segue isn't hooked up to be triggered by an action in your interface, you will need to call this method to trigger the segue.

prepareForSegue:sender:

- This method is called on the source view controller of the segue, right before the segue is actually performed.
- The first paramater is the segue itself, which is an instance of the UISegue class.
- The most important property of a UISegue instance is the destinationViewController property, which gives you a reference to the view controller you are about to segue to.
- This is a great spot to pass information to the next screen.

Demo

More Git

Git Commits

- A git commit is synonymous with saving the state of your project in git.
- Every time you commit, git essentially takes a picture of what all of your files look like at that moment in time, and then stores a reference to that snapshot.
- For the sake of efficiency, if any files have not had any changes made to them since the last commit, git does not store those files again. It just stores a reference to the previous identical file it already has stored in a previous snapshot.

Commits are made locally

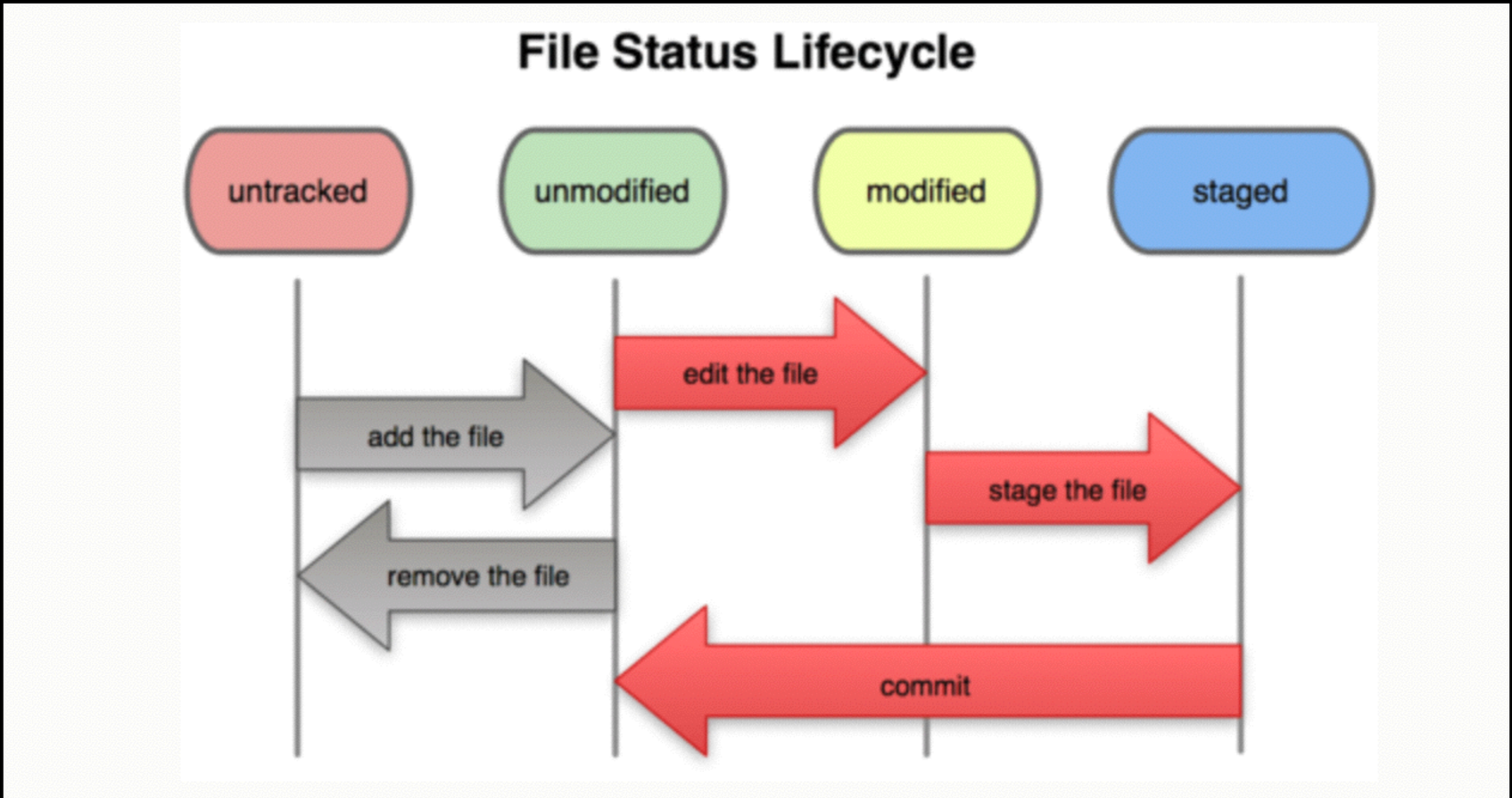
- A git commit is strictly a local operation.
- In fact, almost all operations in git are local, thats why its so much faster than previous version control systems.
- And since repositories have the entire history of the project, you can browse through the entire set of versions of all files while offline, without making any network calls. Technology rocks.

The 3 states

- Git has 3 main states your files can be in:

    1. Unmodified - data is safely stored in your local database. A tracked file without changes is in this state.
    2. Modified - you have changed the file but have not committed it to your database yet
    3. Staged - you have marked a file with changes to go into the next commit

Git File Status Lifecycle



The 3 sections

- Git uses 3 sections for a git project:

    1. **git directory** - where git stores the metadata and object database for your project. This is the most important part of git, and it's what is copied when you clone a repository from another computer/server. It is that hidden .git file.
    2. **working directory** - a single checkout of one version of the project. These files are pulled from the compressed database in the git directory and placed on disk for you to use. Any coding you do will be in the working directory.
    3. **staging area** - a simple file, generally contained in your git directory, that stores information about what will go into your next commit. Sometimes referred to as the index.

Working Directory

- Every file in your working directory has only 2 states - tracked and untracked.
- tracked means it was in the last snapshot, and untracked means it wasn’t.
- Once a file is tracked, it can then be one of 3 states: unmodified, modified, staged
- So when you first initialize a repository, nothing is being tracked. You will need to git add the files so they can be tracked by git.

Basic Git Workflow

1. You modify files in your working directory.
2. You stage the files, which adds snapshots of them to your staging area.
3. You do a commit, which takes files as they are in the staging area and stores snapshot permanently to your git directory.

git commit command

- Use the git commit once your staging area is set up the way you want it.
- Running just plane git commit will launch your text editor configured in your global git config.
- It does this because git wants a git commit message describing what this commit has changed.
- Alternatively, you can use the git commit -m “commit message” to enter your commit message inline.

Demo