ITP 342
Mobile App Development

Model

View

Controller

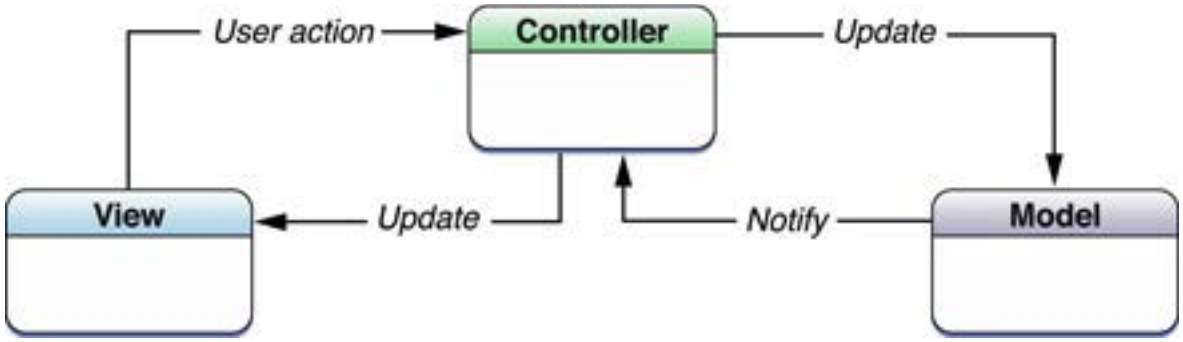
Model-View-Controller

- The Model-View-Controller (MVC) design pattern assigns objects in an application one of three roles: model, view, or controller.
- The pattern defines not only the roles objects play in the application, it defines the way objects communicate with each other.
- Each of the three types of objects is separated from the others by abstract boundaries and communicates with objects of the other types across those boundaries
- The collection of objects of a certain MVC type in an application is sometimes referred to as a layer – for example, model layer.

Model-View-Controller

- MVC is central to a good design for a Cocoa application.
- The benefits of adopting this pattern are numerous.
  - Many objects in these applications tend to be more reusable, and their interfaces tend to be better defined.
  - Applications having an MVC design are also more easily extensible than other applications.
  - Moreover, many Cocoa technologies and architectures are based on MVC and require that your custom objects play one of the MVC roles.

Model-View-Controller



Model Objects

- Model objects encapsulate the data specific to an application and define the logic and computation that manipulate and process that data.
- A model object can have to-one and to-many relationships with other model objects, and so sometimes the model layer of an application effectively is one or more object graphs.
- Much of the data that is part of the persistent state of the application (whether that persistent state is stored in files or databases) should reside in the model objects after the data is loaded into the application.
- Because model objects represent knowledge and expertise related to a specific problem domain, they can be reused in similar problem domains. Ideally, a model object should have no explicit connection to the view objects that present its data and allow users to edit that data—it should not be concerned with user-interface and presentation issues.

Model Objects – Communication

- User actions in the view layer that create or modify data are communicated through a controller object and result in the creation or updating of a model object.
- When a model object changes (for example, new data is received over a network connection), it notifies a controller object, which updates the appropriate view objects.

View Objects

- A view object is an object in an application that users can see.
- A view object knows how to draw itself and can respond to user actions.
- A major purpose of view objects is to display data from the application's model objects and to enable the editing of that data.
- Despite this, view objects are typically decoupled from model objects in an MVC application.
- Because you typically reuse and reconfigure them, view objects provide consistency between applications.
- Both the UIKit and AppKit frameworks provide collections of view classes, and Interface Builder offers dozens of view objects in its Library.

View Objects – Communication

- View objects learn about changes in model data through the application's controller objects and communicate user-initiated changes through controller objects to an application's model objects.
- An example is text entered in a text field.

Controller Objects

- A controller object acts as an intermediary between one or more of an application's view objects and one or more of its model objects.
- Controller objects are thus a conduit through which view objects learn about changes in model objects and vice versa.
- Controller objects can also perform setup and coordinating tasks for an application and manage the life cycles of other objects.

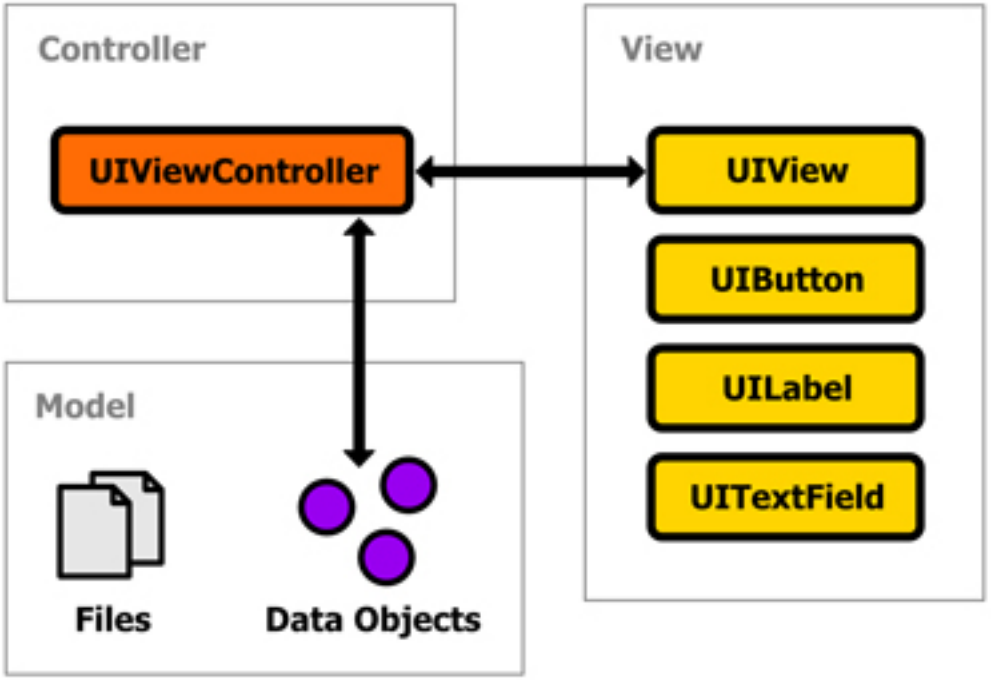
Controller Objects – Communication

- A controller object interprets user actions made in view objects and communicates new or changed data to the model layer.
- When model objects change, a controller object communicates that new model data to the view objects so that they can display it.

MVC

- Model – Hold data
- View – GUI
- Controller – Views & View Controllers
  - Every app has a single *window*
  - Every screenful of content ("scene") sits in a *view*
  - Navigation changes the window's current *view*
  - Each screen's *view* is managed by a *view controller*

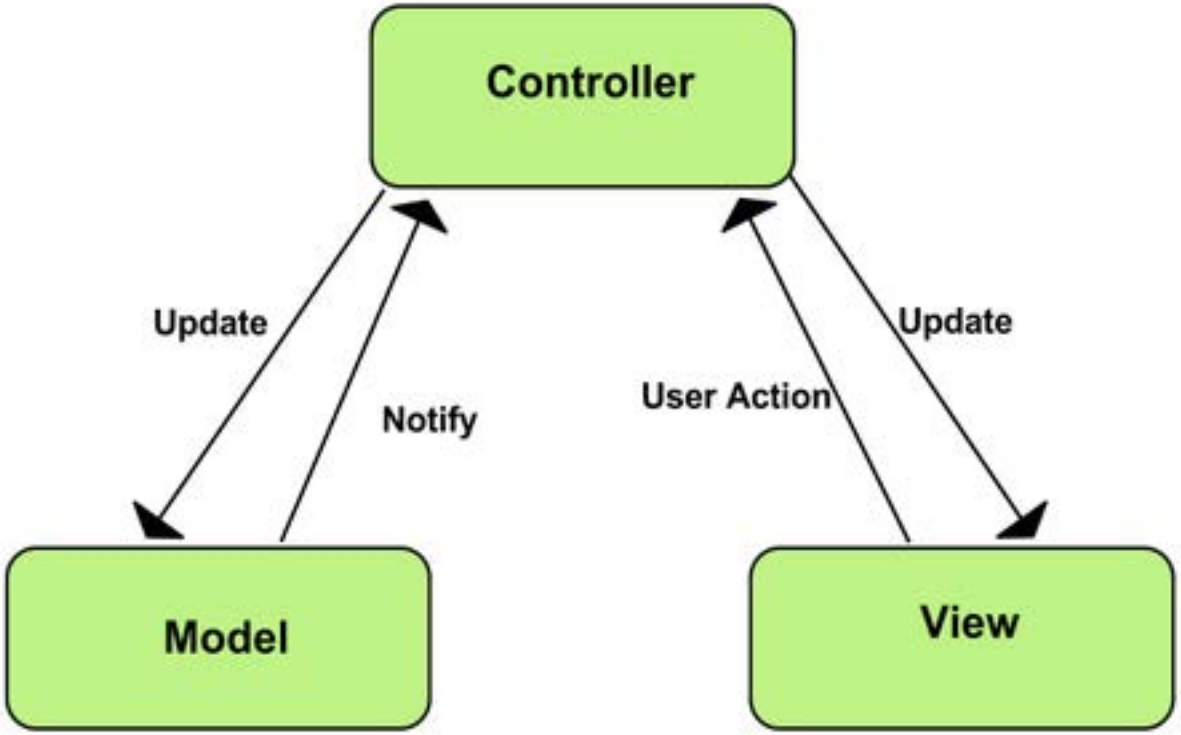
MVC


Controller
UIViewController
View
UIView
UIButton
UILabel
UITextField
Model
Files
Data Objects

Model-View-Controller

- Resources
- https://developer.apple.com/library/ios/documentation/general/conceptual/devpedia-cocoacore/MVC.html
- https://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html

MVC



Model

- Create classes
- Each class contains an interface (.h) file and an implementation (.m) file
- NSObject may be the parent class

View

- Interface Builder
  - Storyboard (.storyboard) file
  - Nib (.nib) or Xib (.xib) file

Controller

- Each scene/view should have a View Controller class (.h and .m)
- Put code in here to change the GUI