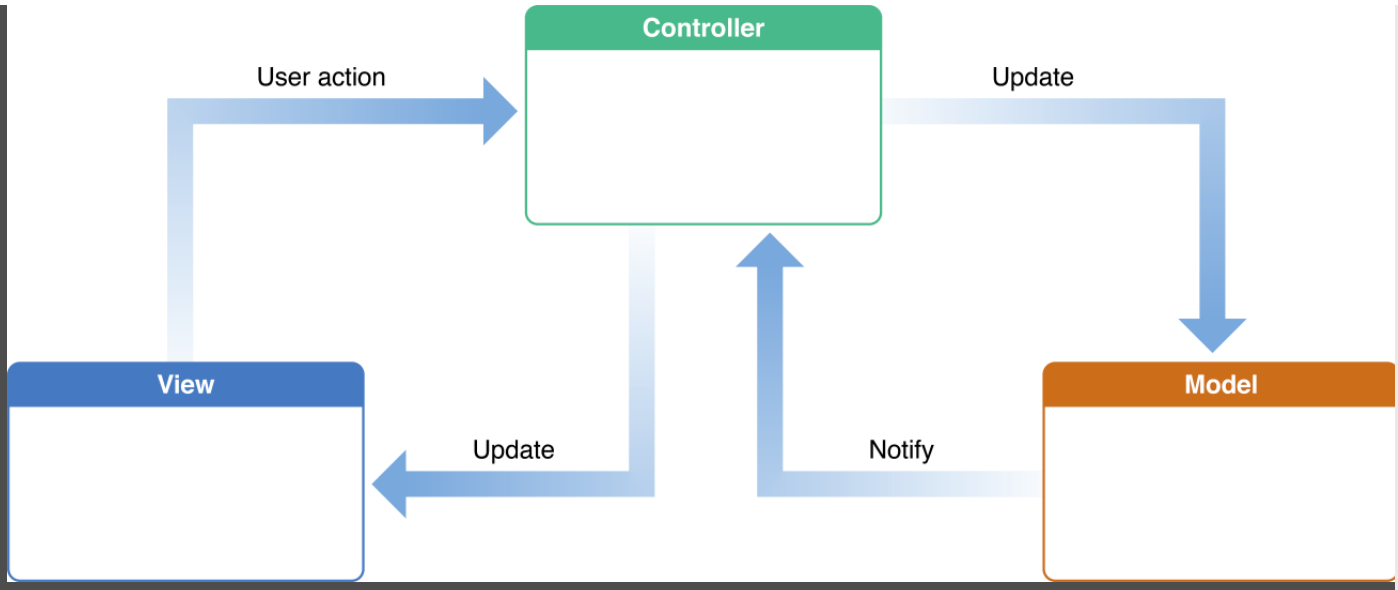
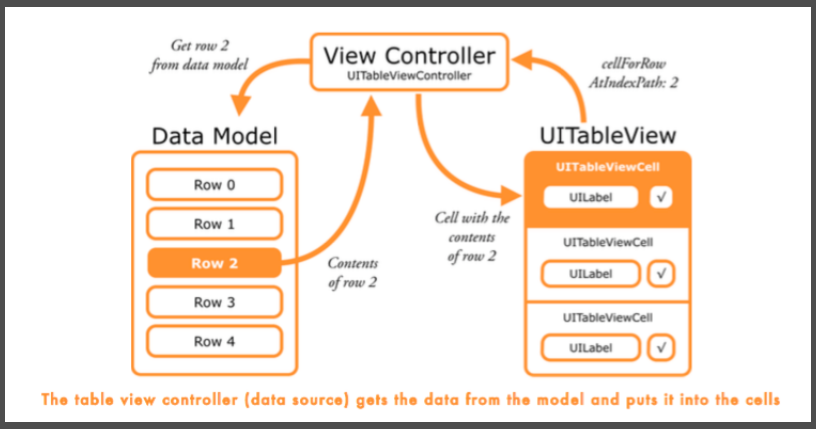
Seneca College of Applied Arts & Technology

PLAIN HTML VERSION: lecture6b.161.txt.html
Xcode Project Archive: lecture6b.161.zip

```
/*
file:    lecture6a.161.swift
author:  Danny Abesdris
date:    February 24, 2016
purpose: MAP523AA/DPS923AA lecture #6b

         Swift programming language topics:
         Model-View-Controller (MVC) design pattern.
*/

/*
Model-View-Controller (MVC):
The Model-View-Controller paradigm is one of the three fundamental design patterns of
The others are, Delegation-> making one object do something on behalf of another;
and target-action (connecting events such as button taps to action methods in the View

Model-View-Controller allows the objects in your app to be split up into three groups:

1. Model objects -> Contain your data and any operations on the data. For example, in
   Lab #3, the AuthorsViewController.swift file contains the class that handles
   the AuthorsView Scene.
   The operations that the data model objects perform are sometimes called the busines
   rules or the domain logic.

2. View objects -> These objects make up the visual part of the app: images, buttons,
   labels, text fields, table view cells, and so on.
   A view can draw itself and responds to user input, but it typically does not handle
   application logic. Many views, such as UITableView, can be re-used in many differen
   apps because they are not tied to a specific data model.

3. Controller objects. -> The controller is the object that connects your data model
   objects to the views. It listens to taps on the views, makes the data model objects
   calculations/perform other logic in response, and updates the views to reflect the
   state of your model. The controller is what handles all of these events.
   In iOS, the controller is called the “view controller”.

Conceptually, this is how these three building blocks fit together:
```




```
References:
iOS Model Object: ModelObject.html
iOS Model View Controller: MVC.html

How the Model-View-Controller works:
The view controller has one main view, accessible through its view property, and
contains many subviews. A screen may have dozens of views all at once.
The top-level view usually fills the whole screen and requires you (the programmer)
to design the layout of the view controller's screen on the Main.storyboard.
In Lab #3 the main view is the UITableView and its subviews are the table view cells.
```




```
A view controller handles one screen of the app. If your app has more than one screen,
the each of these is handled by its own view controller and has its own views. An iOS a
flows from one view controller to the other.

Views vs. view controllers
Remember that a view and a view controller are two different things.
A view is an object that draws something on the screen, such as a button or a label
(i.e. the view is what you see).
The view controller on the other hand is what does the work behind the scenes. It is
the bridge that sits between your data model and the views.
Accordingly, view controller names should contain the name of the object along with
the word "Controller", such as: ViewController, AuthorsViewController, BooksViewContro

*/
```


Made on a
Mac

BEST VIEWED WITH
AnyBrowser