iOS DeCal : Lecture 4

UITableViews and UICollectionViews

February 28, 2017

Announcements

Project 1 (Hangman) due Tonight at 11:59pm

Submit on Gradescope using GitHub option

See Lecture 3 Slides for instructions

Custom App Proposal due March 23

Start brainstorming ideas + forming groups

We will have formalized proposal instructions posted soon

Overview : Today's Lecture

Delegation

UIScrollViews

UITableViews

UICollectionViews

Delegation

What is Delegation?

Design Pattern

Allows objects to interact with each other without creating dependencies

via **Protocols**, **Delegates**, and **Data Sources**

Protocols

Protocol: a generic outline or skeleton
Set of rules that delegates must follow
Can be above a class declaration, or in own .swift file

Classes that follow such rules are said to *"conform to the protocol"*

Classes can conform to any number of protocols

Protocols

```
protocol SomeDelegate {
    func sendBack(str: String)
    func updateModel(strs: [String])
}
```

For a class to conform to this protocol `SomeDelegate`, it must implement the `sendBack` and `updateModel` methods

Protocols

```
class ViewController: UIViewController, SomeDelegate {
    func sendBack(str: String) {
        print(str)
    }
    func updateModel(strs: [String]) {
      //assume model exists
        model.appendContentsOf(strs)
    }
}
```

`ViewController` is a subclass of `UIViewController` and conforms to the protocol `SomeDelegate`

Delegates/Data Sources

Delegates

Typically responsible for the UI

example: UITableViewDelegate

Data Source

Typically responsible for the Data (Model)

example: UITableViewDataSource

UIScrollViews

UIScrollViews : What are they?

Right now the user is limited to the screen space they have on their phone

UIScrollView allows a view to extend beyond it's space, letting the user "scroll" around

You will rarely use ScrollViews directly

The important part to note is that `UIScrollView` is a **Superclass** of `UITableView` and `UICollectionView`

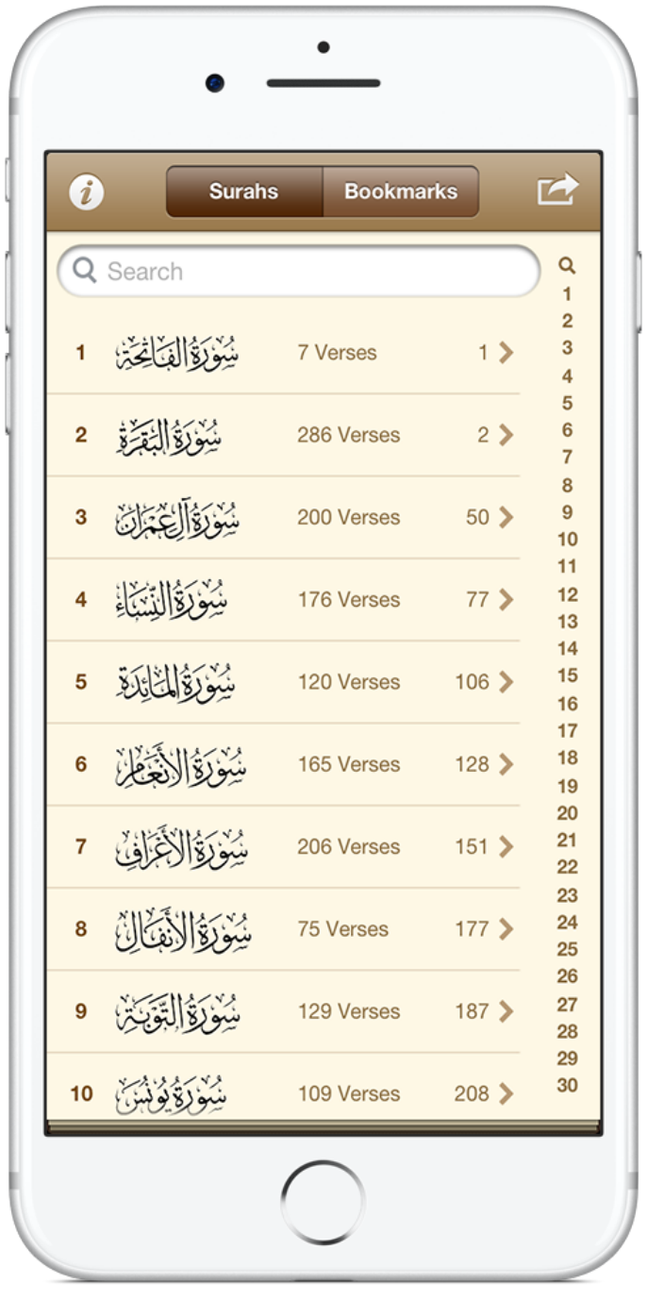
UITableViews

UITableViews : What are they?

UIKit class that presents data in a scrollable list.

UIScrollView is superclass

Each row is a UITableViewCell, very configurable



UITableViewCells

UITableViewCells are the components that make up a tableView.

Cells are “reusable”, old cells that aren’t in view anymore are used to create new ones.

Built out of “prototype cells” which you define.

UITableViewCells : Attributes

`indexPath`: How you index into a TableView,

Consists of .section and .row attributes (more on this soon).

`identifier`: Which prototype cell is being used to construct it.

`.textLabel`: textLabel that is contained in the cell. There are also other accessories depending on the style you use, for example, RightDetail, in addition to .text, has **.detailTextLabel**, which contains smaller text that shows up in the right part of the cell

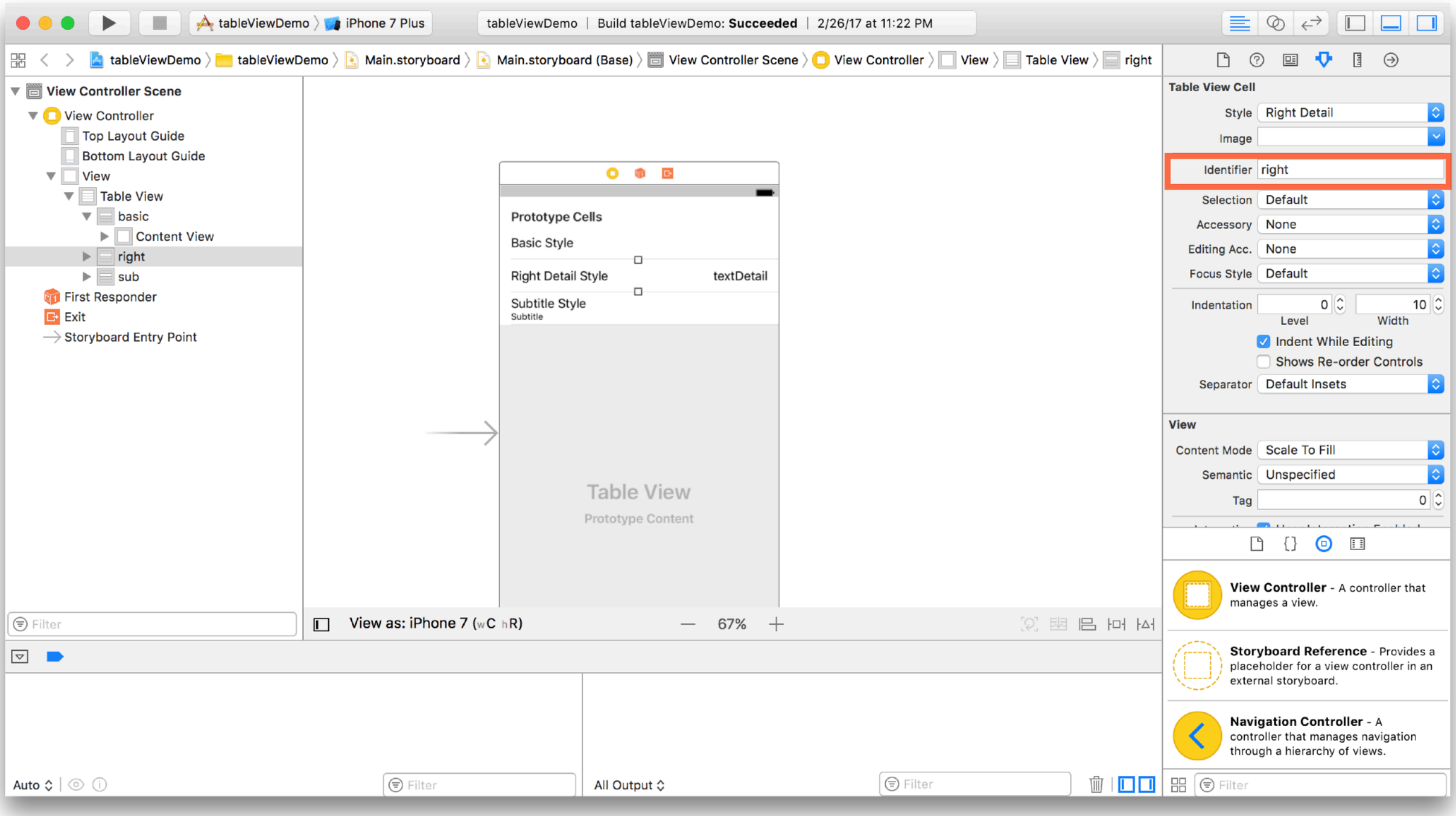
UITableViewCells : Prototype Cells

You can make your own cells in storyboard.

When you make a new TableView, you will see a cell in it to start with. You can edit this cell, for example adding views for more text, or a place for an image.

Make sure that any prototype cell you make has it's identifier attribute set! You'll need this shortly. (will definitely have an image here for this)

UITableViewCells : Prototype Cells



UITableViews : Delegation

In order to create the aforementioned cells and use the data that will fill those cells, delegation is used.

Delegates manage row configuration, ordering, highlighting and editing

UITableViewDelegate : Required Methods

numberOfRowsInSection

cellForRowAtIndexPath

UITableViewDataSource : Methods

numberOfRowsInSection

cellForRowAtIndexPath

```
func tableView(_ tableView: UITableView,
               numberOfRowsInSection section: Int) -> Int {

    if section < 3 {
          return numRows1 \\ an integer
     } else {
          return numRows3 \\ a different integer
     }
}
```

UITableViewDataSource : Methods

numberOfRowsInSection

cellForRowAtIndexPath

```
func tableView(tableView: UITableView,
               cellForRowAt indexPath: IndexPath)
              -> UITableViewCell {

  var cell = tableView.dequeueReusableCell(withIdentifier: ”cell”
                                           for: indexPath)
                                           as! UITableViewCell

  cell.textLabel?.text = self.items[indexPath.row]
  return cell
}
```

UITableView Delegation : Other Methods

Some other methods of interest include:

numberOfSections

heightForRowAtIndexPath

didSelectRowAtIndexPath

etc.

UITableViews : UITableViewController

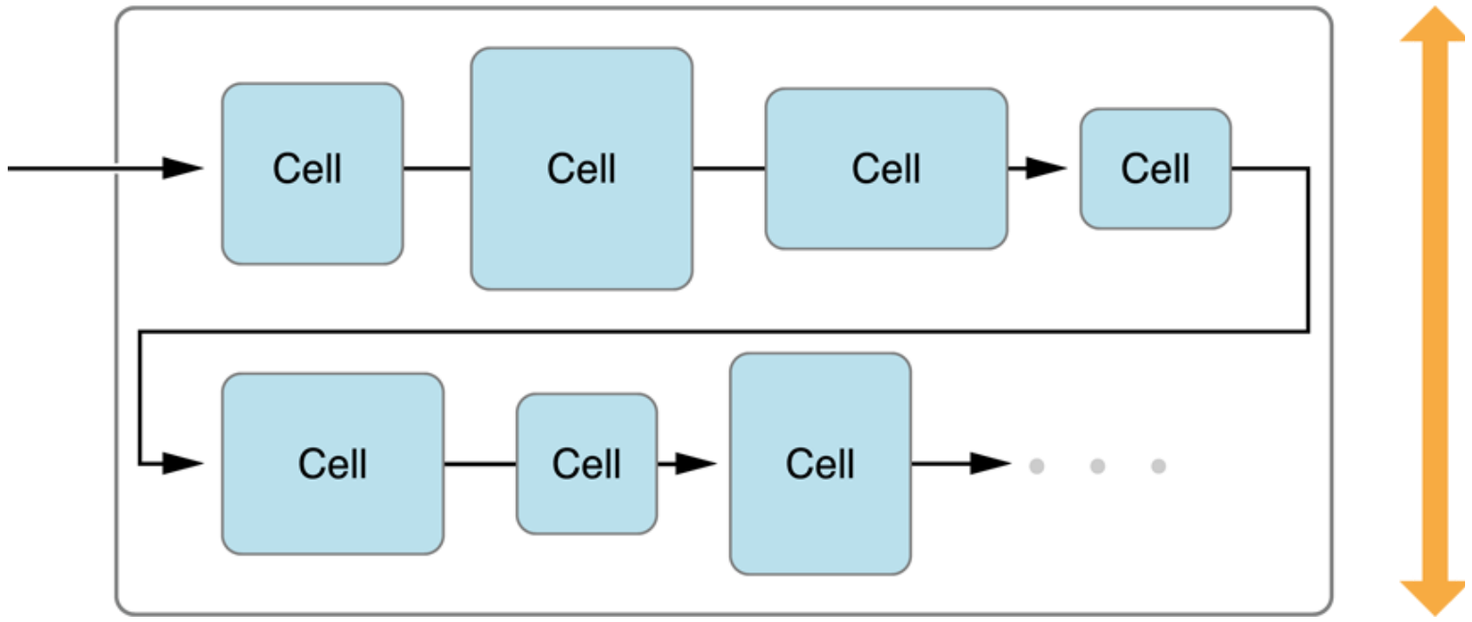
A ViewController specifically for containing a TableView (and nothing else).

Is basically a ViewController that automatically connects to a data source and delegate.

Has a .tableView property for accessing the Table View.

UICollectionViews

UICollectionViews



Manages an ordered collection of data items and presents them using customizable layouts.

UICollectionViews : What are they?

Just like UITableViews, except that instead of rows, we have a collection of cells.

Flexible and Customizable View

- Commonly used for grid-like layouts
- Other uses: stack, circular layouts, etc.

Apple provides us with UICollectionView

Similar implementation to UITableView

UICollectionViews : Methods

The following are pretty useful methods, the bolded ones are necessary:

numberOfSections

numberOfItemsInSection

cellForItemInIndexPath

didSelectItemAtIndexPath

sizeForItemAtIndexPath

TableViews/CollectionViews

This is a pretty high level look at these structures, there are a lot of details that I haven't mentioned (i.e. reloading data, connecting your view to the delegate/data source, etc.)

You will learn a lot more from watching the upcoming demo!

Check-Ins

Demo

Project 1 : Hangman

Due **Tonight** at 11:59pm

Next Lecture: CoreLocation, MapKit, AVFoundation, and CoreData