How to Make an App For Beginners

Module 3
Lesson 5
Worksheet

Welcome!

In this worksheet, you'll practice conforming to protocols and implementing the delegate methods.

Just in case you don't remember how you got here (or if you need a refresher), Lesson 5 can be found here:

How To Build a Match Game - Lesson 5 (Protocol & Delegates)

Step 1: We'll be doing these exercises in Xcode, so start a new **Xcode Project**

Open Xcode and create a new project
(File Menu->New->Project).
Select "**Single View App**" under the "iOS" tab

Step #2: We're going to be doing this for the UITableView class. This is very similar to the UICollectionView class actually!

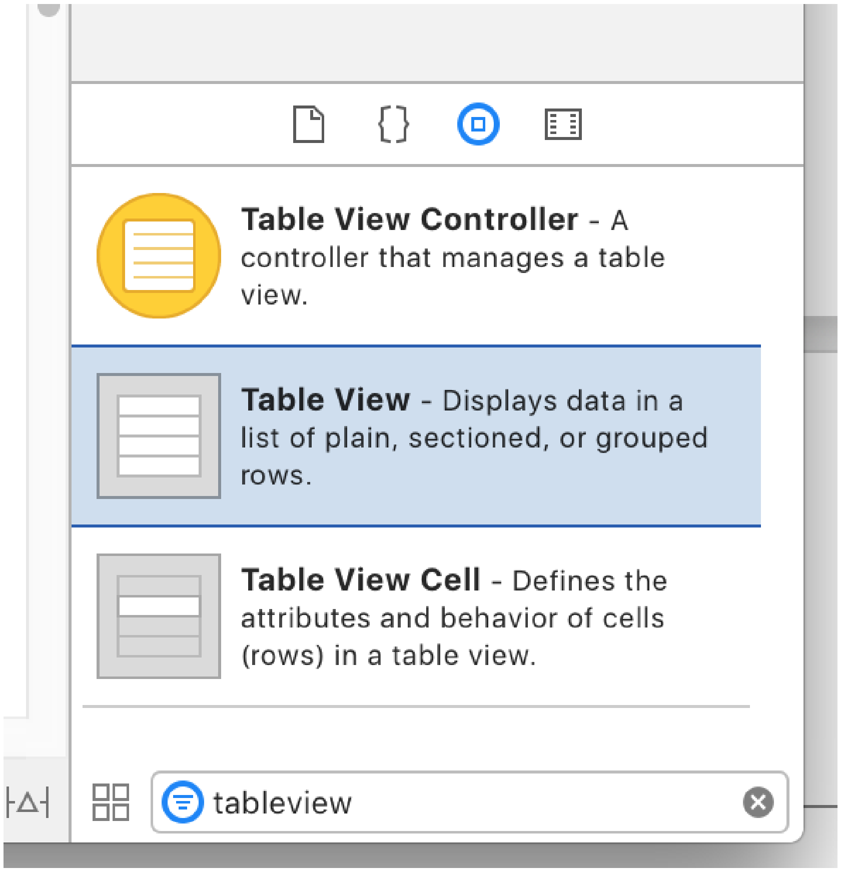


Table View Controller - A controller that manages a table view.

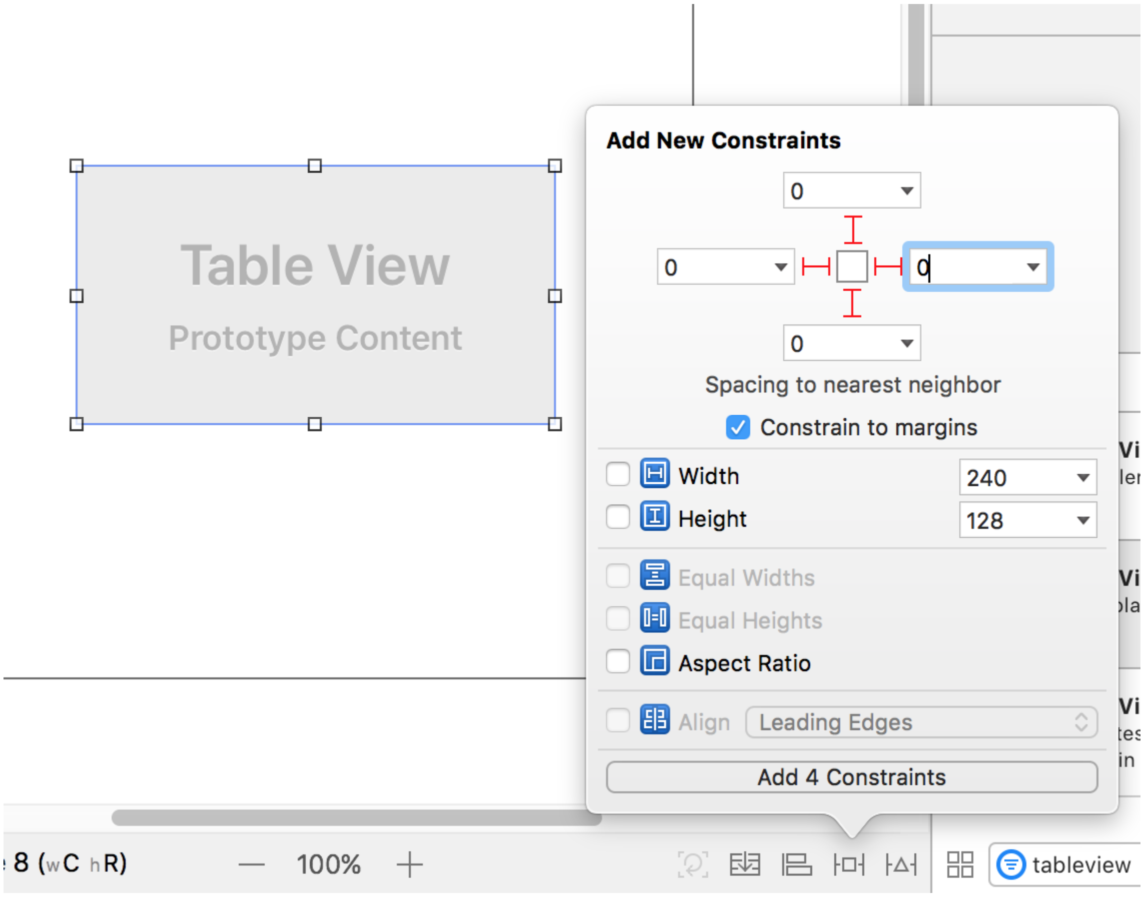
Table View - Displays data in a list of plain, sectioned, or grouped rows.

Table View Cell - Defines the attributes and behavior of cells (rows) in a table view.

tableview

Go to the storyboard and in the lower right hand corner, search for tableview. Drag and drop one into your view. Make sure you select the right one! (see screenshot above).

Step #3: Next, select the tableview in your view and click the “Add Constraints” button near the bottom of the screen and add 4 margin constraints, all set to zero.



Add New Constraints

0

0 0

0

Spacing to nearest neighbor

☑ Constrain to margins

□ Width 240

□ Height 128

□ Equal Widths

□ Equal Heights

□ Aspect Ratio

□ Align Leading Edges

Step #4: With the tableview still selected, go to the right hand inspector pane and set the number of prototype cells to 1.

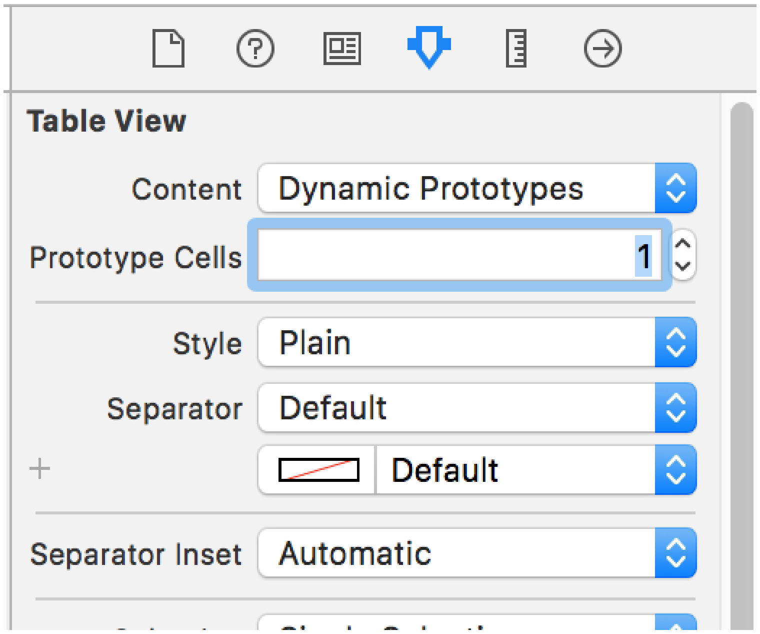


Table View

Content: Dynamic Prototypes

Prototype Cells: 1

Style: Plain

Separator: Default

Default

Separator Inset: Automatic

Step #5: Doing that will add a prototype cell. You'll see it both in your view and your document outline (the listing of elements in your view). Select it from the document outline and in the inspector panel, set the identifier to "BasicCell"

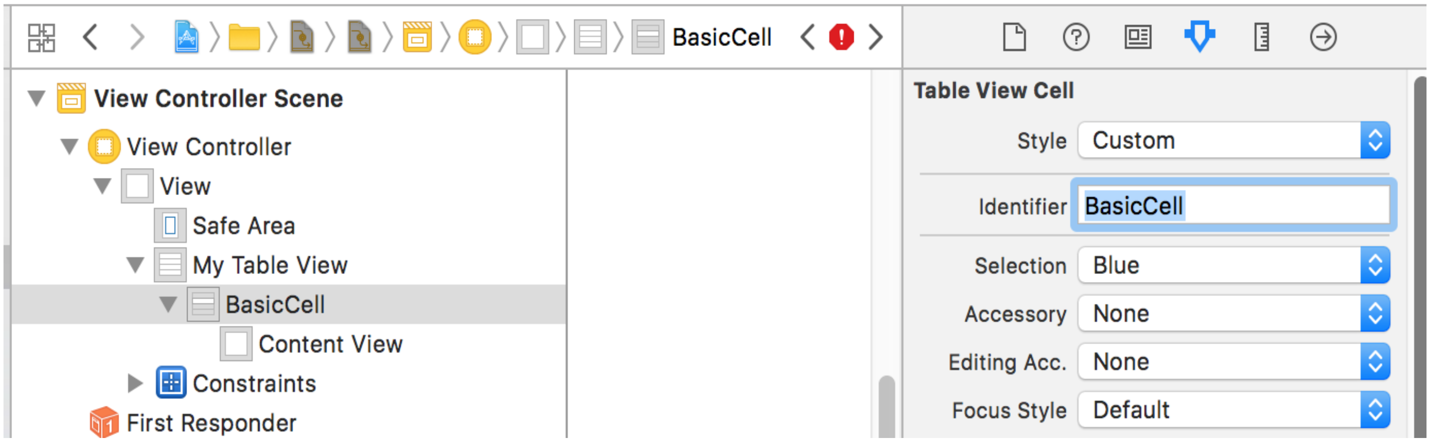


Table View Cell

Style: Custom

Identifier: BasicCell

Selection: Blue

Accessory: None

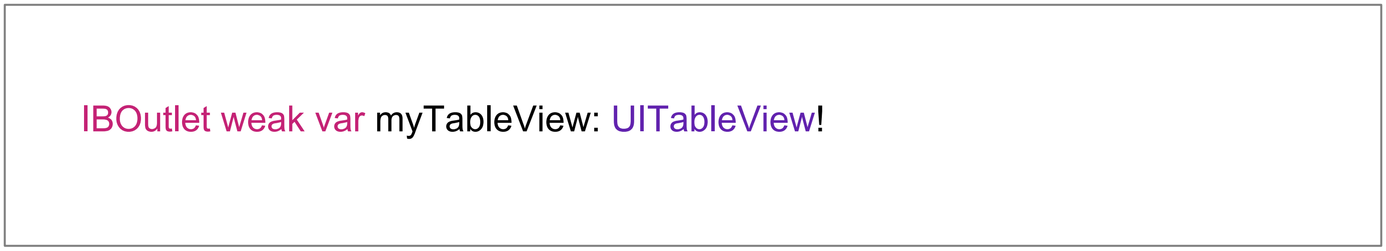
Editing Acc.: None

Focus Style: Default

Step #6: Connect the tableview as an IBOutlet property in your ViewController. If you don’t remember how to do this, refer to Module 1, Lesson 8 – Connecting Elements in the Storyboard.

I called my IBOutlet property, “myTableView”.

Now in your ViewController, you’ll have a property that looks like this: (mine says myTableView but yours will say whatever you named your property)



```
IBOutlet weak var myTableView: UITableView!
```

After you do this, run your project to make sure it’s not crashing. If it is, you might have connected your IBOutlet property incorrectly. Start watching at the 14 min mark of Module 1, Lesson 8 for troubleshooting steps.

Step #7: In the viewDidLoad method of the ViewController, let’s set the ViewController as the **delegate** and **datasource** of our table view:

```
override func viewDidLoad() {
    super.viewDidLoad()

    myTableView.delegate = self
    myTableView.dataSource = self

}
```

Xcode might complain and say that the ViewController doesn’t conform to the protocols but we’re about to fix that in the next step.

Step #8: Let’s declare that the ViewController conforms to both the UITableViewDelegate protocol and UITableViewDataSource protocol. Add these two protocols beside the parent class:

```
class ViewController: UIViewController, UITableViewDelegate,
```

Step #9: Let’s create some data to display in the tableview. Right below your tableview outlet property, declare the following array:

```
let dataArray = ["bird", "dog", "cat", "turtle", "bear"]
```

At this point, your ViewController class looks like this:

```
import UIKit

class ViewController: UIViewController, UITableViewDelegate,
UITableViewDataSource {

    @IBOutlet weak var myTableView: UITableView!
    let dataArray = ["bird", "dog", "cat", "turtle", "bear"]

    override func viewDidLoad() {
        super.viewDidLoad()

        myTableView.delegate = self
        myTableView.dataSource = self

    }
```

Step #10: Now that we've declared that the ViewController class conforms to those protocols AND we've set it as the delegate and datasource of the tableview, it's time to implement the required protocol methods from those protocols.

Underneath the didReceiveMemoryWarning method, type the following functions:

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


}


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

Step #11: In the “numberOfRowsInSection” method, we need to tell the tableview how many rows of data we have. So let’s return the count of the dataArray:

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        return dataArray.count
```

Step #12: In the “cellForRowAt” method, we need to specify a tableview cell for the tableview to display. The tableview tells us which row it’s trying to show and it’s up to us to get a cell, configure it for the data in that row and then give it back to the tableview.

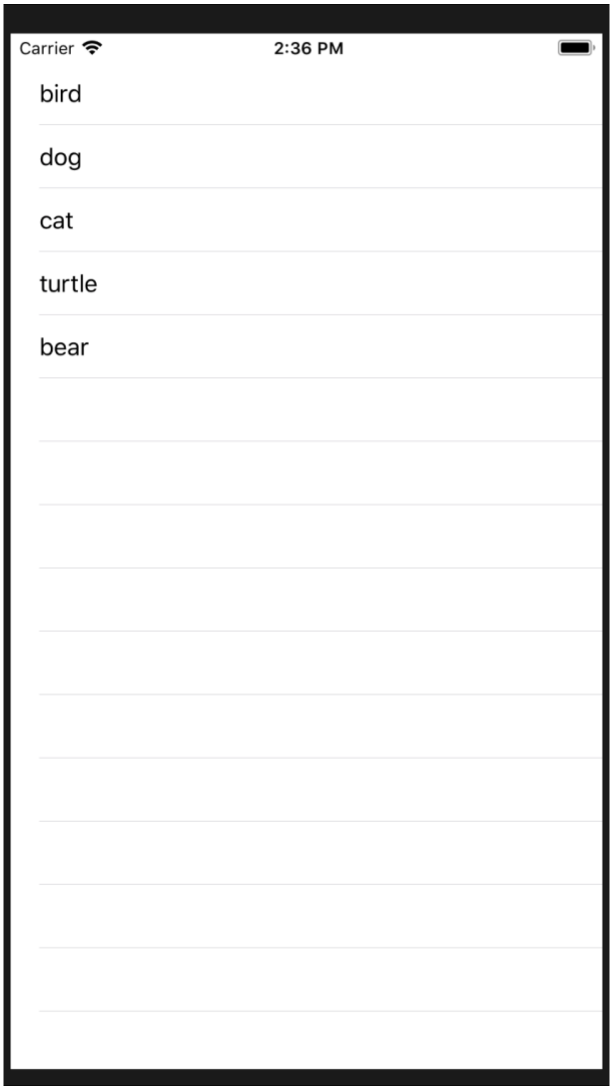
```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

        // Get the data for this row
        let rowData = dataArray[indexPath.row]

        // Get a cell to display
        let cell = tableView.dequeueReusableCell(withIdentifier: "BasicCell", for: indexPath)
        cell.textLabel?.text = rowData

        // Return the cell
        return cell
}
```

Step #13: When you run your project, you should see:



bird

dog

cat

turtle

bear

You’re done! If you completed this exercise, celebrate and let me know on Twitter!
https://twitter.com/CodeWithChris

To see my project, download it here:
Completed Project