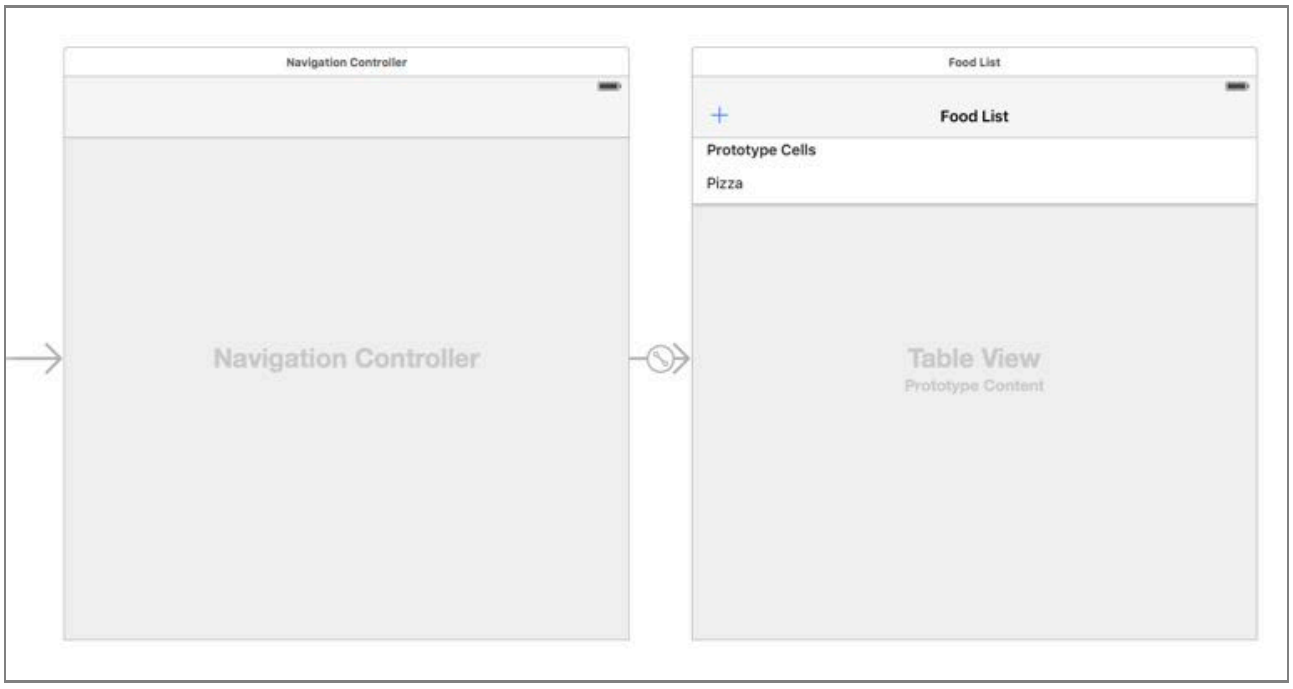
1.0 Lesson - Table Views for Beginners

Paul Solt - Paul@SuperEasyApps.com
SuperEasyApps.com

1.1 Lecture - Introduction to the UITableView

In this lesson you will learn how to create a UITableView that can add and remove data from an array in your app. You will work with multiple screens and sharing information between screens.

1.2 Tutorial - Table View and Navigation Controller UI Setup



1. Drag a UITableView onto your ViewController (Not TableViewController)
2. Add a UINavigationController (top bar)
3. Set the Navigation Root View Controller to the ViewController
4. Add Auto Layout constraints (–64 to top)
5. Create a Prototype Cell (Default)
6. Add a reuse identifier: “FoodCell”
7. Add an Add (+) Bar Button Item to the top left corner
8. Change the name of the Navigation Controller to “FoodListNavigationController”
9. Make the FoodListNavigationController initial view controller

1.3 Tutorial - Table View Delegate and DataSource

1. Create a connection with the Assistant Editor for the Table View

```
@IBOutlet weak var foodTableView: UITableView!
```

2. Set the delegate and dataSource properties in viewDidLoad()

```
override func viewDidLoad() {
    super.viewDidLoad()
    foodTableView.delegate = self
    foodTableView.dataSource = self
}
```

3. Make the ViewController conform to the delegate protocols UITableViewDelegate and UITableViewDataSource

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
```

4. Add the methods to show test data in the list

```
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return 10
}
```

5. Create a row cell and add test data

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
```

```
UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("FoodCell", forIndexPath:
indexPath)
    cell.textLabel?.text = "Food"
    return cell
}
```

Links

- Table View Programming Guide for iOS
- Table View Reference
- UITableViewDelegate Reference
- UITableViewDataSource Reference

1.4 Tutorial - Table View Data Setup

1. Create your data or model array below your outlets, above viewDidLoad()

```
// Variables
var foodArray = ["Pizza", "Hoagie", "Thai Curry", "Vegetables"]
```

2. Update the tableView(_: numberOfRowsInSection:) method to use the foodArray's size

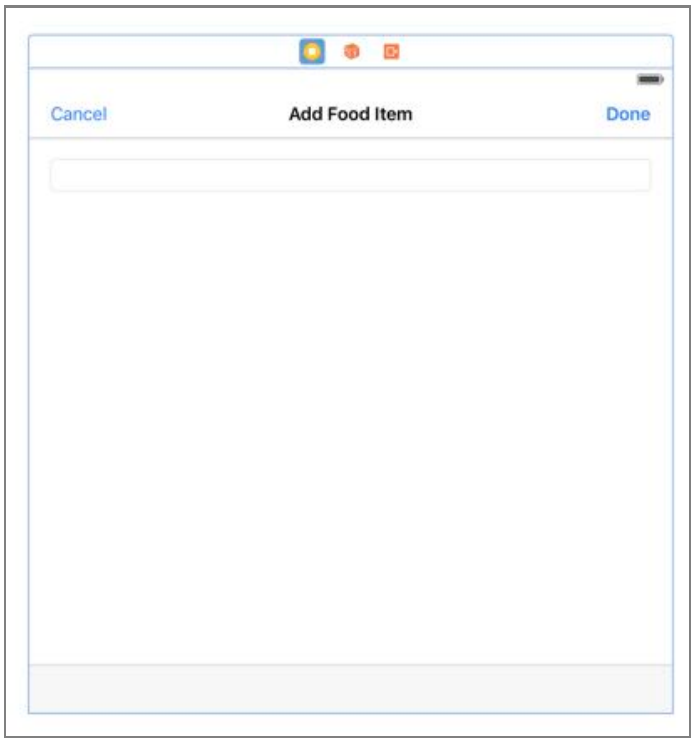
```
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return foodArray.count
}
```

3. Update the tableView(_: cellForRowAtIndexPath:) method to populate the cell with the data

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    let cell = tableView.dequeueReusableCellWithIdentifier("FoodCell", forIndexPath:
indexPath)
    if indexPath.row < foodArray.count {
        cell.textLabel?.text = foodArray[indexPath.row]
        // cell also has an imageView + detailTextLabel properties
    }
    return cell
}
```

1.5 Tutorial - Design and Setup the AddFoodViewController

Design the User Interface



1. Create a new AddFoodViewController.swift as a subclass of UIViewController
2. Drag a new View Controller in your Main.storyboard file
3. Set the new View Controller as a AddFoodViewController to link the code file
4. In the Main.storyboard set the AddFoodViewController's Simulated Metrics Top Bar to "Opaque Navigation Bar" instead of inferred
5. Drag a `UINavigationBar` onto the `AddFoodViewController`
6. Set the title to "Add Food Item"
7. Drag two Bar Button Items for the `Cancel` and `Done` buttons
8. Add a `UITextField` below the navigation bar
9. Setup Auto Layout constraints

Connect the Code to UI

1. Create an Outlet for the `UITextField` called `foodTextField`

```
@IBOutlet weak var foodTextField: UITextField!
```

2. Add two Actions for the `doneButtonPressed(_:)` and `cancelButtonPressed(_:)`

```
@IBAction func doneButtonPressed(sender: AnyObject) {
    print("done pressed")
}

@IBAction func cancelButtonPressed(sender: AnyObject) {
    print("cancel pressed")
}
```

1.6 Tutorial - Presenting a Modal View Controller

In the ViewController.swift file add code to show this new screen.

1. Connection an action called `addFoodButtonPressed(_:)`
2. Add the logic to create and show a the `AddFoodViewController`

```
@IBAction func addFoodButtonPressed(sender: AnyObject) {
    let foodController =
self.storyboard?.instantiateViewControllerWithIdentifier("AddFoodViewController") as!
AddFoodViewController
    // wrap in a temporary navigation controller
    let navigationFoodController = UINavigationController(rootViewController:
foodController)
    self.presentViewController(navigationFoodController, animated: true, completion: nil)
}
```

1.7 Tutorial - Delegate Protocols for Passing Messages

You will use a delegate protocol to communicate between the two code files.

1. Create a delegate protocol at the top of AddFoodViewController.swift

```
protocol AddFoodViewControllerDelegate {
```

```
    func foodController(foodController: AddFoodViewController, didAddFood food: String)
    func foodControllerDidCancel(foodController: AddFoodViewController)
}
```

2. Create a delegate property in AddFoodViewController.swift

```
var delegate: AddFoodViewControllerDelegate? = nil
```

3. Call the delegate's methods in AddFoodViewController.swift

```
@IBAction func doneButtonPressed(sender: AnyObject) {
    if foodTextField.text?.isEmpty == false { // Don't exit without text
        delegate?.foodController(self, didAddFood: foodTextField.text!)
    }
}

@IBAction func cancelButtonPressed(sender: AnyObject) {
    delegate?.foodControllerDidCancel(self)
}
```

4. In ViewController.swift `addFoodButtonPressed(_:)` set the foodController.delegate property

```
@IBAction func addFoodButtonPressed(sender: AnyObject) {
    let foodController =
self.storyboard?.instantiateViewControllerWithIdentifier("AddFoodViewController") as!
AddFoodViewController
    foodController.delegate = self // required for delegate protocol message
    // wrap in a temporary navigation controller
    let navigationFoodController = UINavigationController(rootViewController:
foodController)
    self.presentViewController(navigationFoodController, animated: true, completion: nil)
}
```

5. Conform to the AddFoodViewControllerDelegate protocol at the top of ViewController.swift

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource,
AddFoodViewControllerDelegate {
```

6. Add method stubs for the two AddFoodViewControllerDelegate methods

```
func foodController(foodController: AddFoodViewController, didAddFood food: String) {
    print("Food: \(food)")
}
```

```
func foodControllerDidCancel(foodController: AddFoodViewController) {
    print("canceled")
}
```

Links

- Protocols
- Delegation

1.8 Tutorial - Adding a New Food Item and Editing the Table View

1. Add logic to add the food item to the list in ViewController.swift
2. Dismiss the modal popup using the `dismissViewControllerAnimated(_:completion)` method

```
func foodController(foodController: AddFoodViewController, didAddFood food: String) {
    print("Food: \(food)")
    foodArray.append(food)
    let indexPath = NSIndexPath(forRow: foodArray.count - 1, inSection: 0)

    foodTableView.insertRowsAtIndexPaths([indexPath], withRowAnimation: .Automatic)
    dismissViewControllerAnimated(true, completion: nil)
}

func foodControllerDidCancel(foodController: AddFoodViewController) {
    print("canceled")

    dismissViewControllerAnimated(true, completion: nil)
}
```

Links

- Inserting and Deleting Rows and Sections - apple.com

1.9 Tutorial - Enable Editing of the Table View

1. Add the default `editButtonItem` to the top right bar button item using code in viewDidLoad() of ViewController

```
override func viewDidLoad() {
    super.viewDidLoad()
    foodTableView.delegate = self
    foodTableView.dataSource = self
    navigationItem.rightBarButtonItem = self.editButtonItem()
}
```

2. Make an outlet for the addButton at the top of ViewController.swift

```
@IBOutlet weak var addButton: UIBarButtonItem!
```

3. Override Apple's method setEditing(_: animated) to change states

```
// Editing
override func setEditing(editing: Bool, animated: Bool) {
    super.setEditing(editing, animated: animated)
    if editing {
        addButton.enabled = false
    } else {
        addButton.enabled = true
    }
    foodTableView.setEditing(editing, animated: animated)
}
```

1.10 Tutorial - Removing Items From the List

1. Enable row editing for all rows by implementing the method

```
func tableView(tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) ->
Bool {
    return true
}
```

2. Enable row deletion by returning the editing style

```
func tableView(tableView: UITableView, editingStyleForRowAtIndexPath indexPath:
NSIndexPath) -> UITableViewCellEditingStyle {
    return .Delete
```

```
}
```

3. Add code to remove an entry from the foodArray as well as the tableView

```
func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {

    if editingStyle == .Delete {
        // remove from array
        foodArray.removeAtIndex(indexPath.row)
        // remove from UI
        tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Automatic)
    }
}
```

Links

- Inserting and Deleting Rows and Sections - apple.com
- Table View Programming Guide for iOS