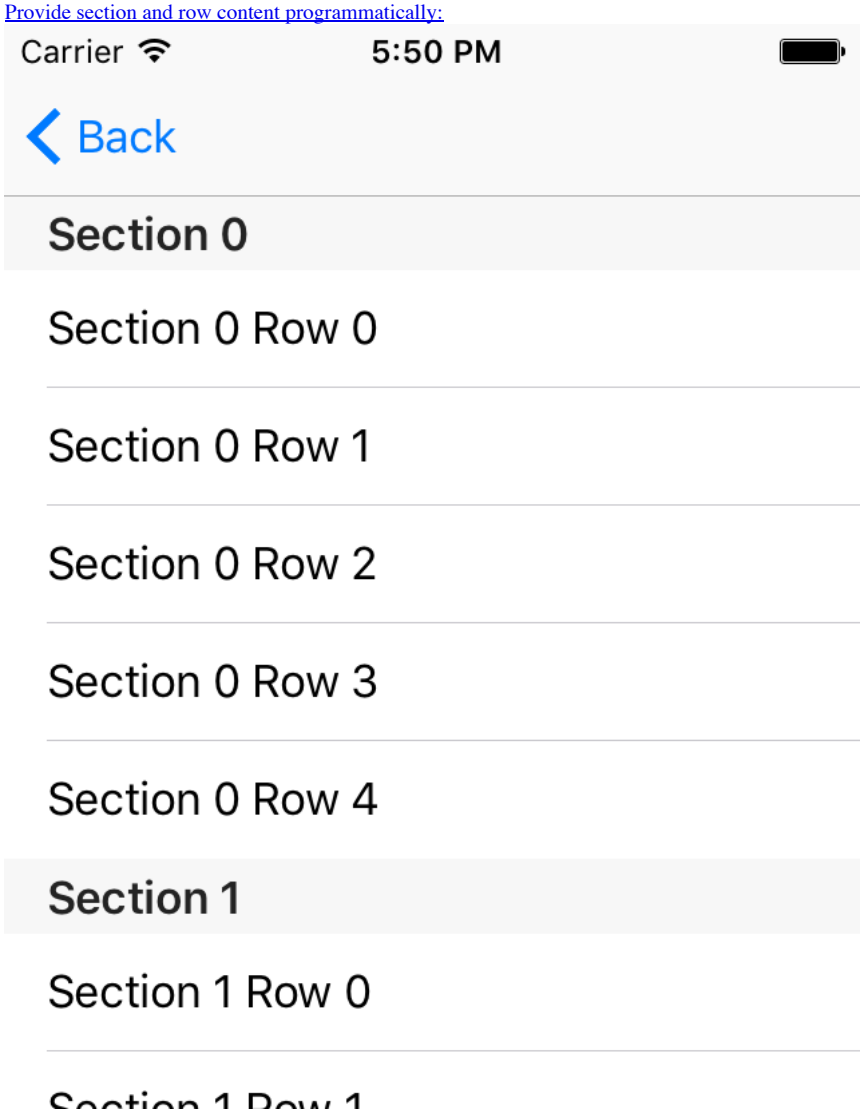
25.09.2016, Ralf Ebert » Tutorials »

iOS & Swift Tutorial: UITableViewController

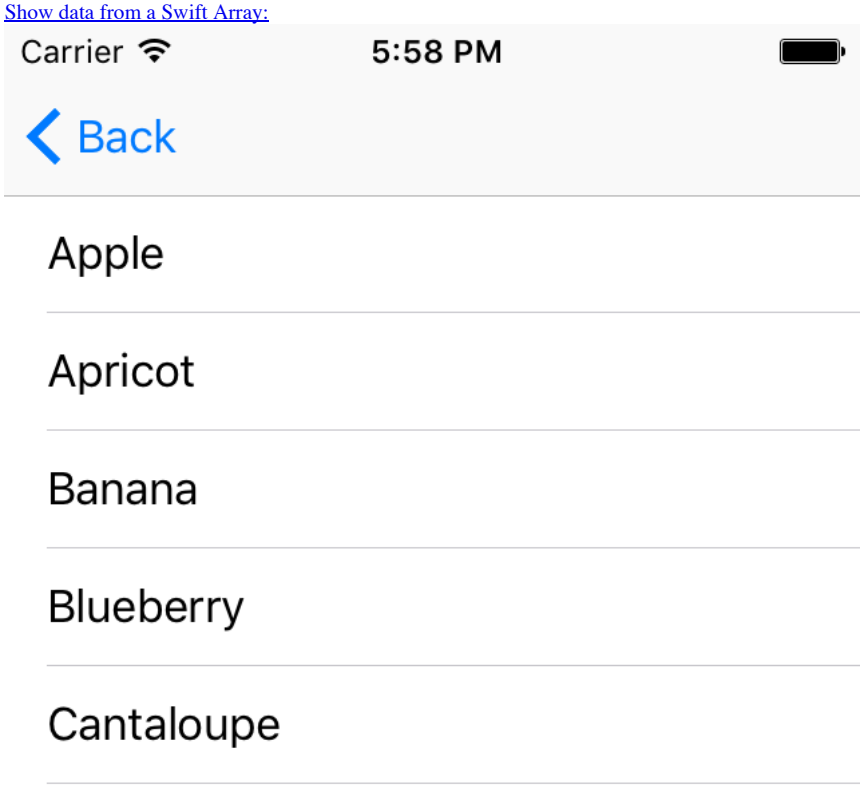
This tutorial shows how to:

- Configure table contents using the Storyboard:

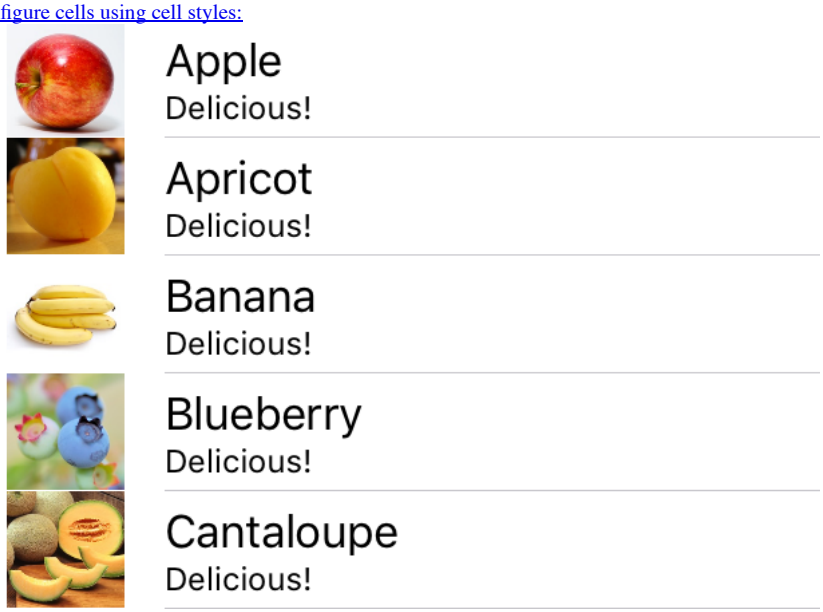
- Provide section and row content programmatically:



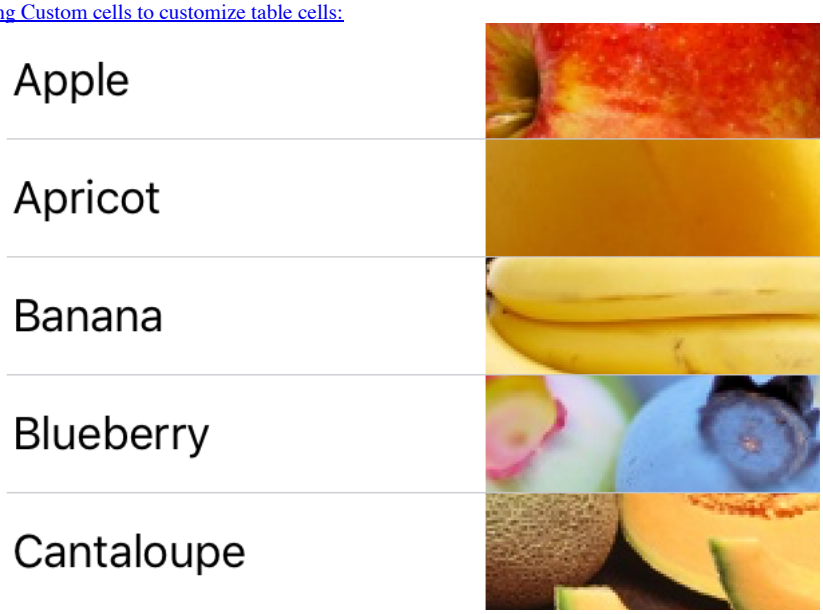
- Show data from a Swift Array:



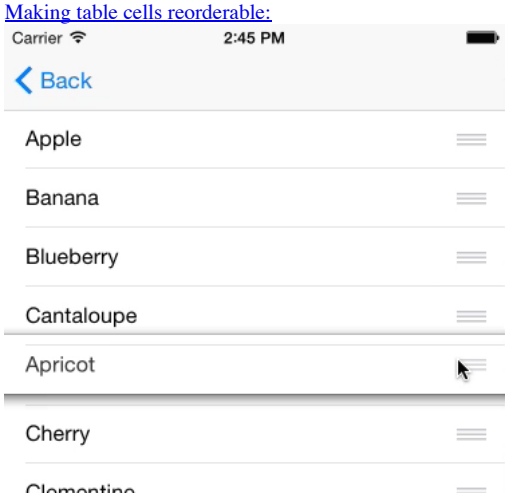
- Configure cells using cell styles:



- Using Custom cells to customize table cells:



- Making table cells reorderable:



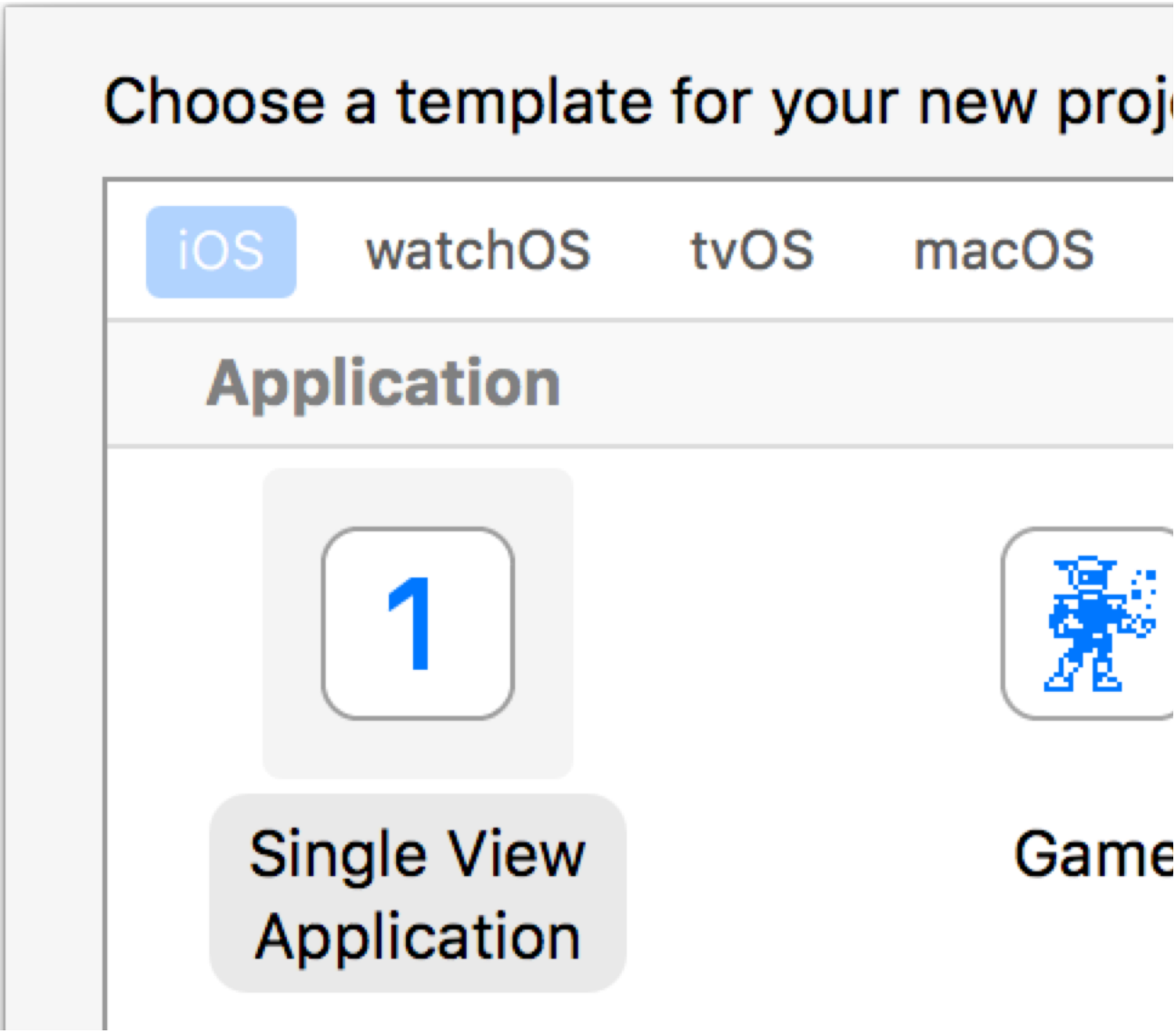
Requirements

For this tutorial you need basic programming skills and know how to use Xcode. You should know everything that is in the "How to develop an iPhone app" iOS Tutorial. To develop iOS apps for the iPhone/iPad using the latest Xcode 8, you'll need a Mac running the at least *Mac OS X 10.11 El Capitan*.

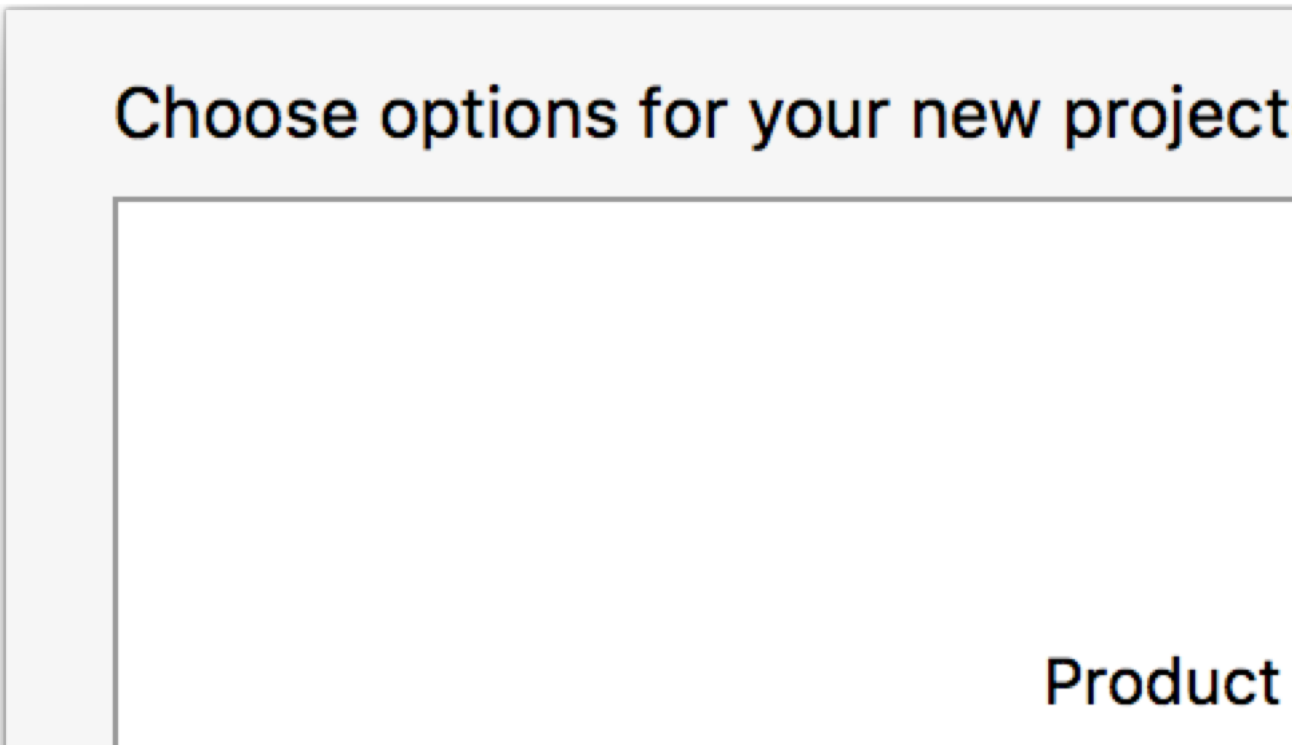
Creating an example project

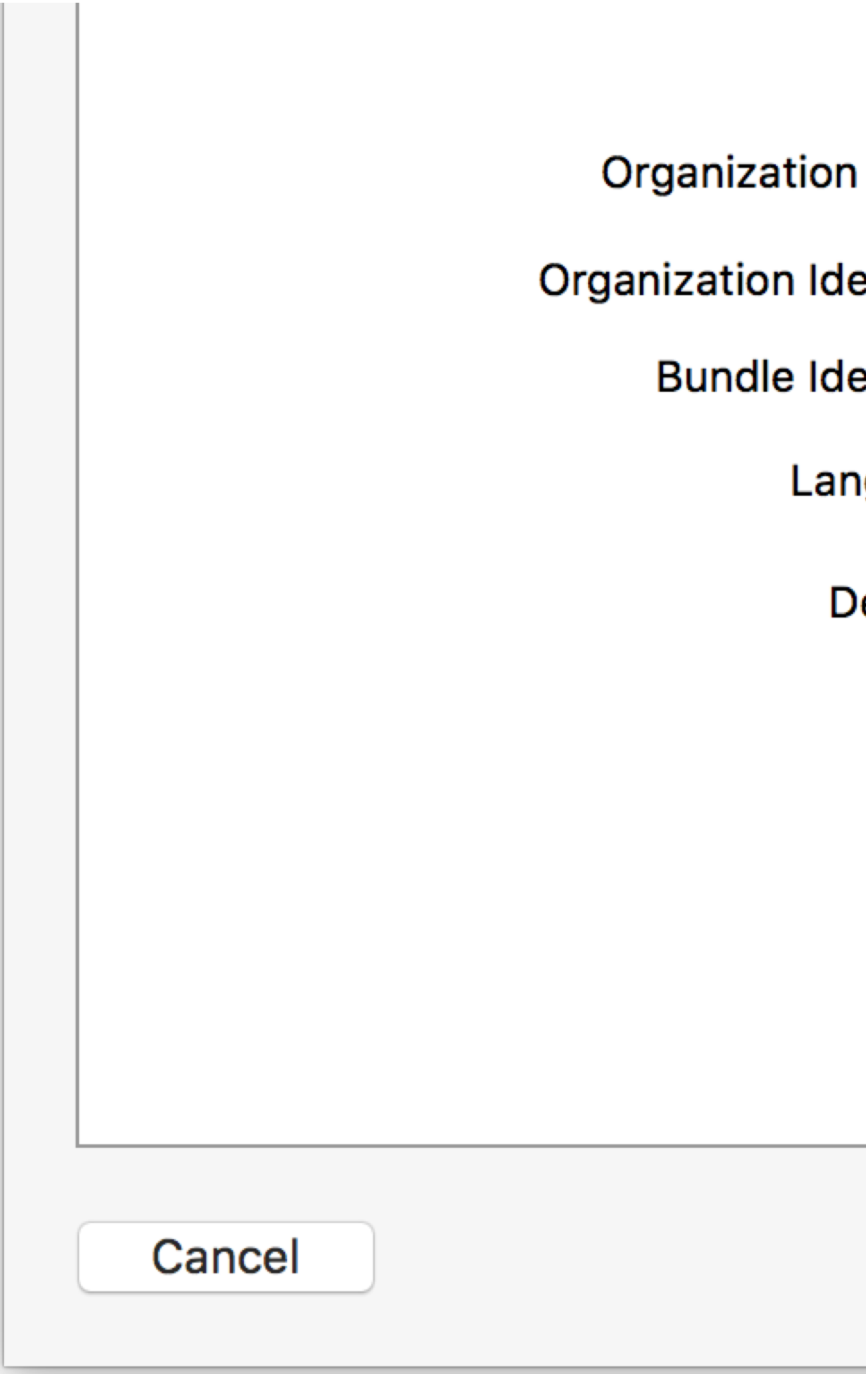
1. Run *Xcode* and check that you're using the latest Xcode version:

2. Create a new project with *File* > *New* > *Project* (⌘⇧N). Select *iOS* > *Application* > *Single View Application*:

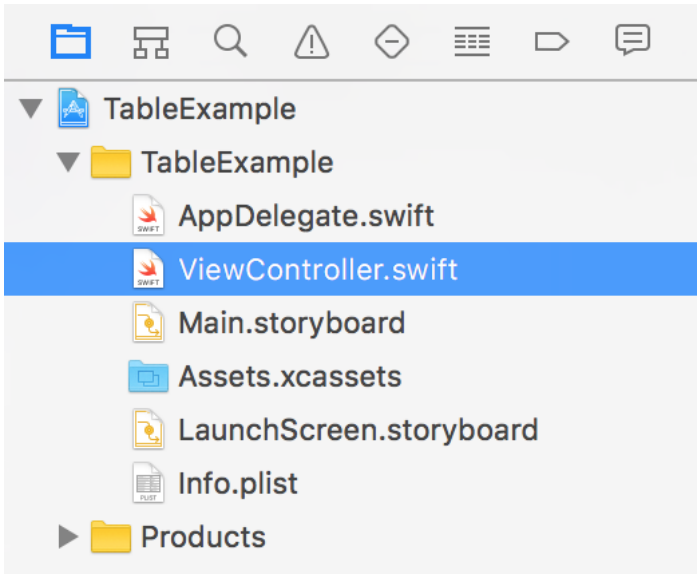


3. Name the app "*TableExample*".
   Choose *Swift* as *Language*, select *iPhone* for *Devices*.

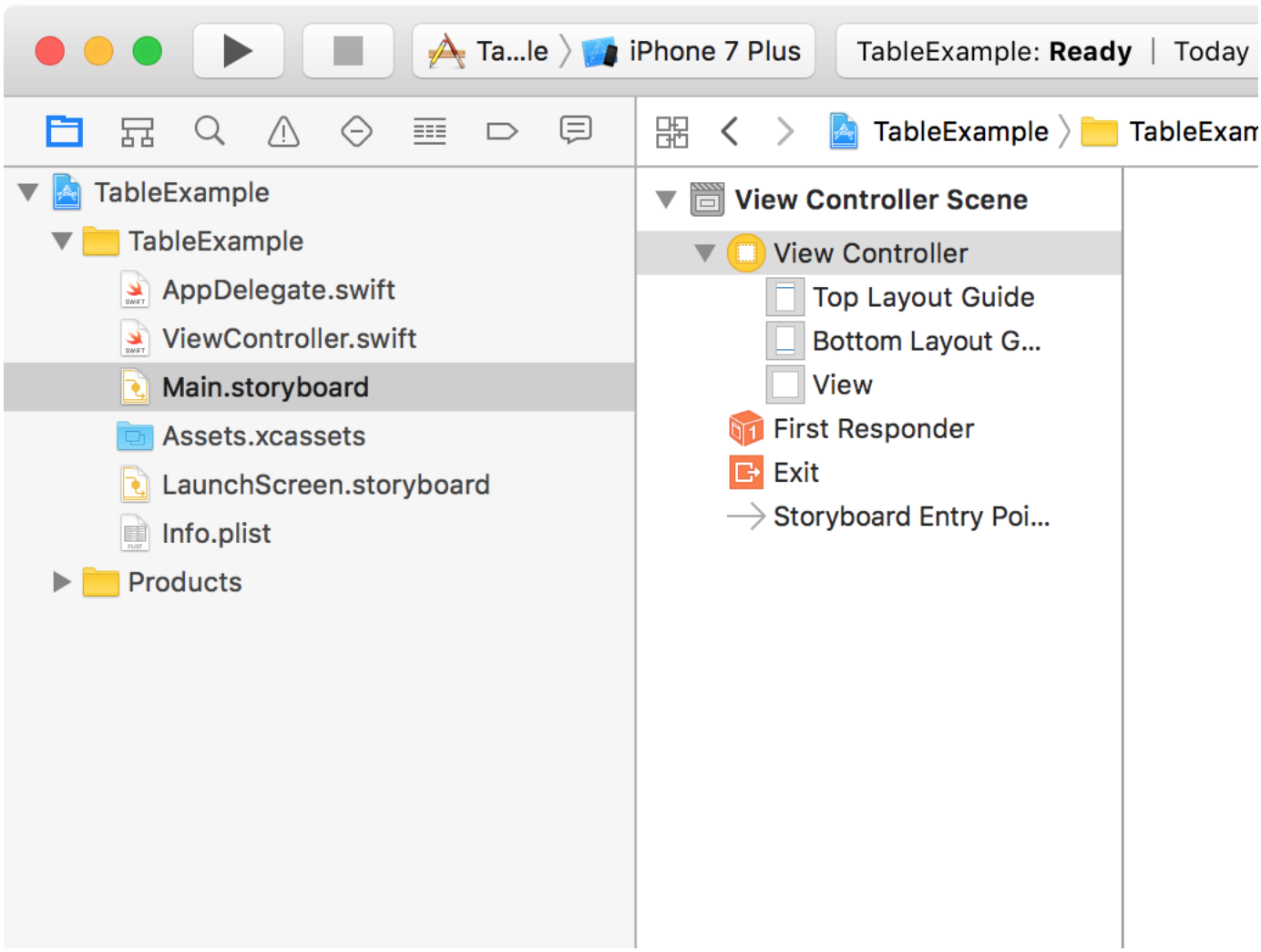



Organization
Organization Ide
Bundle Ide
Lan
D
Cancel

4. Delete the controller class *ViewController.swift* that was created automatically:

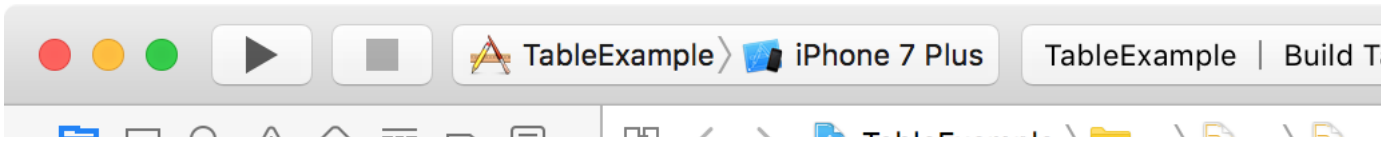


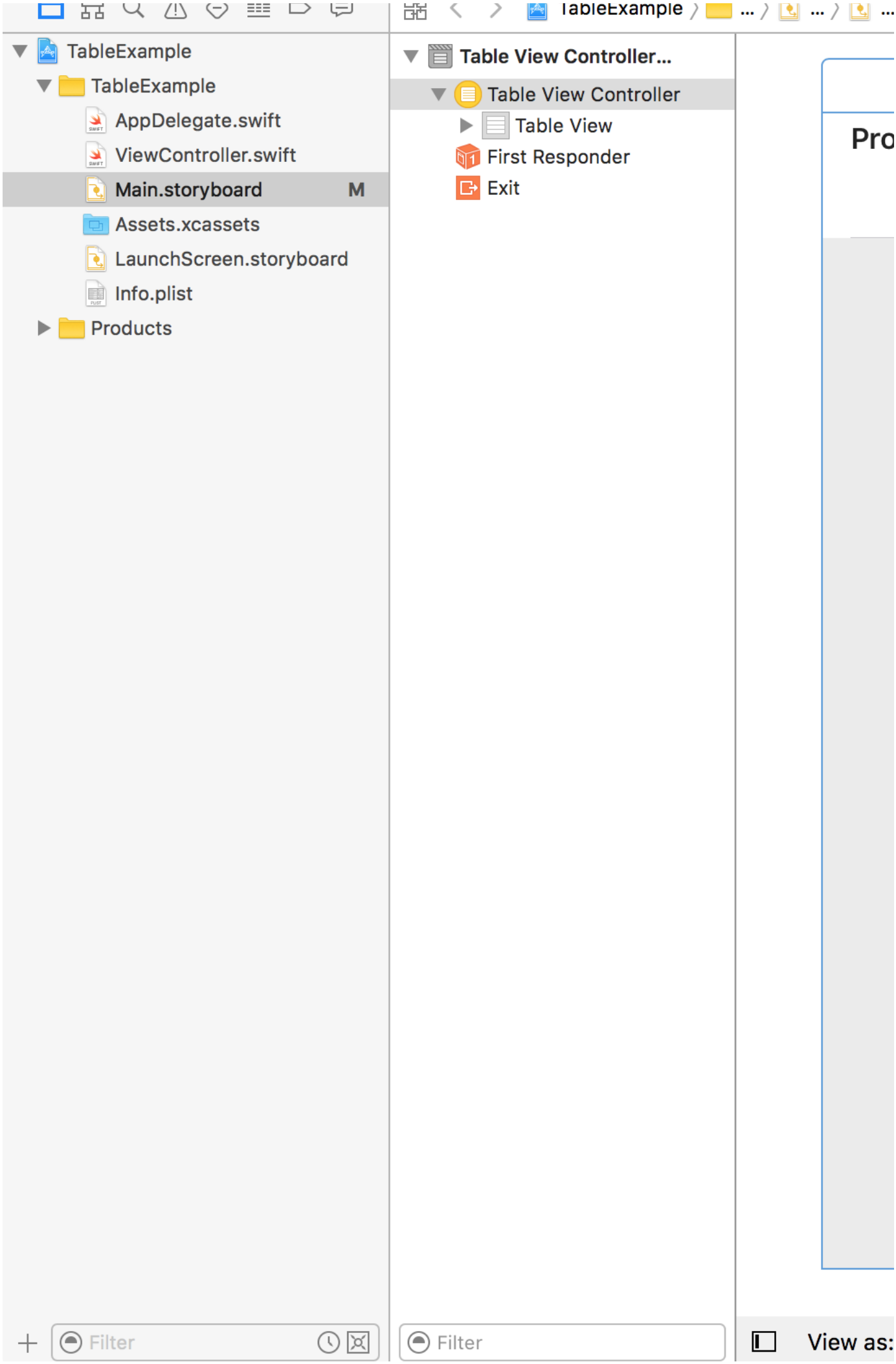
5. Open *Main.storyboard*, select the controller that was created automatically in the *Outline* and delete it with *Edit* > *Delete* ⌫ so that the storyboard is completely empty:



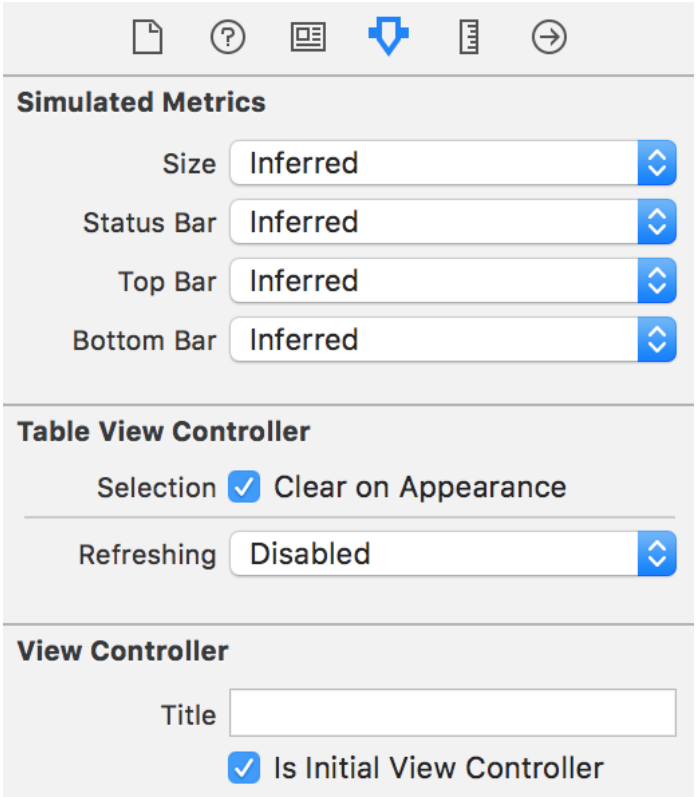
Configure table contents using the Storyboard

1. Drag in a new *Table View Controller* from the *Object Library*:

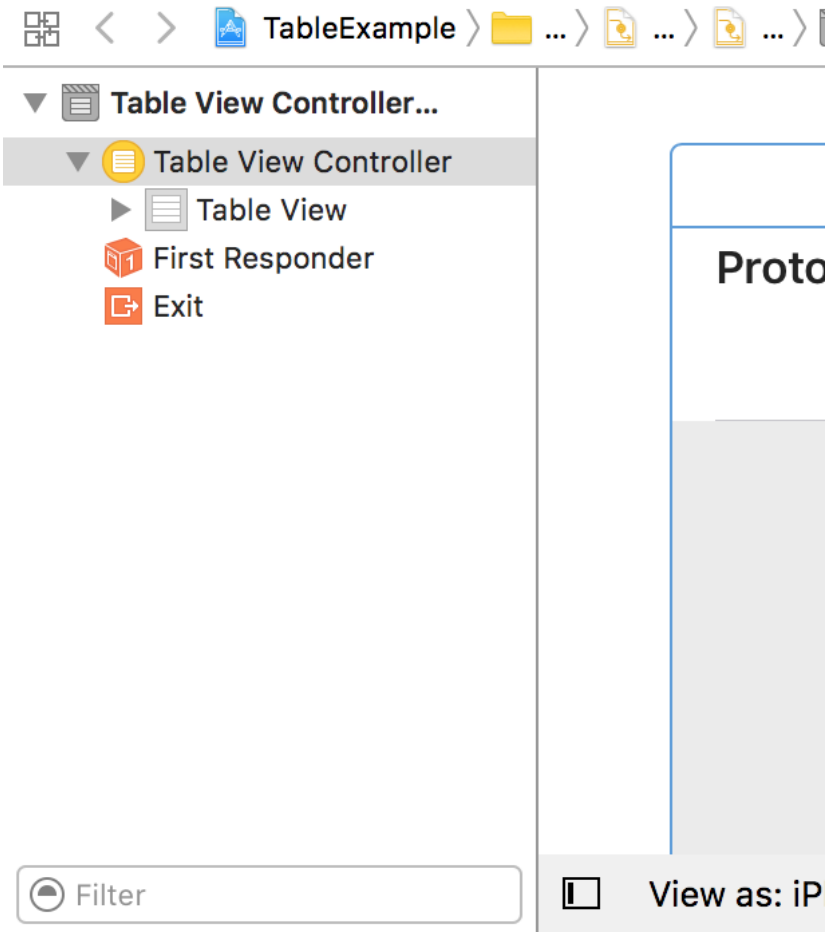



TableExample
TableExample
AppDelegate.swift
ViewController.swift
Main.storyboard
M
Assets.xcassets
LaunchScreen.storyboard
Info.plist
Products
Filter
TableExample
Table View Controller...
Table View Controller
Table View
First Responder
Exit
Filter
Pro
View as:

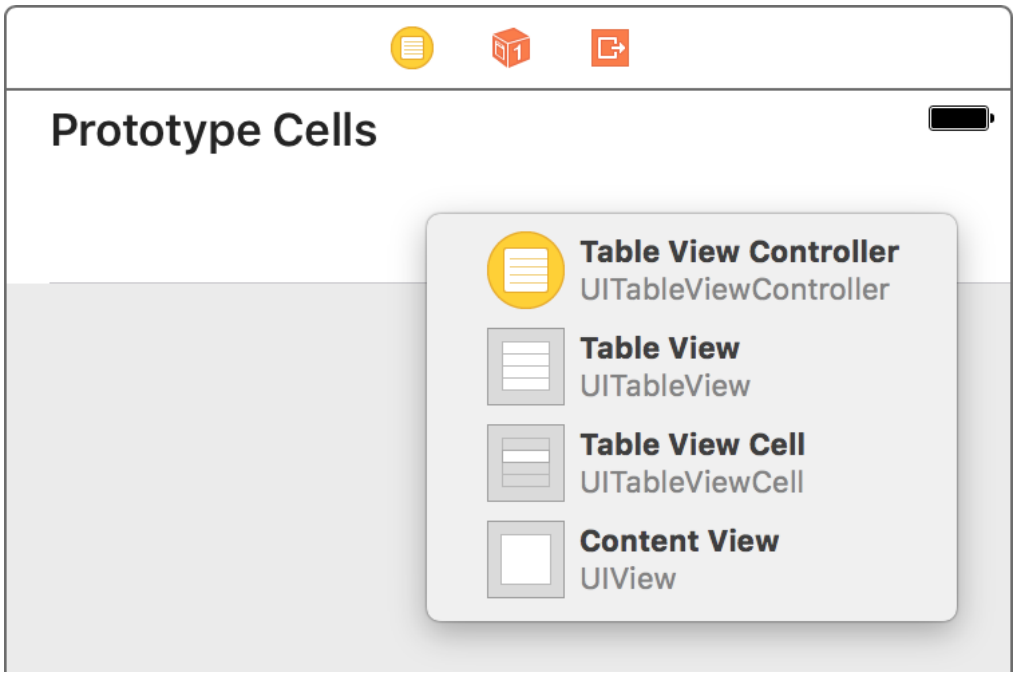
2. Configure the controller as *Initial View Controller*:



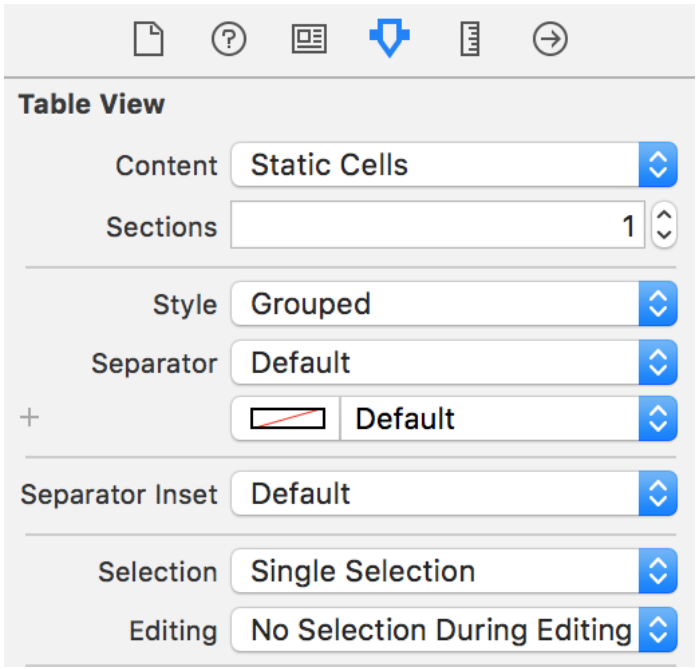
3. Show the *Document Outline* and familiarize yourself with the object structure. The *Table View Controller* manages its *Table View* consisting of a single *Table View Cell* row and an empty *Content View* in the cell:



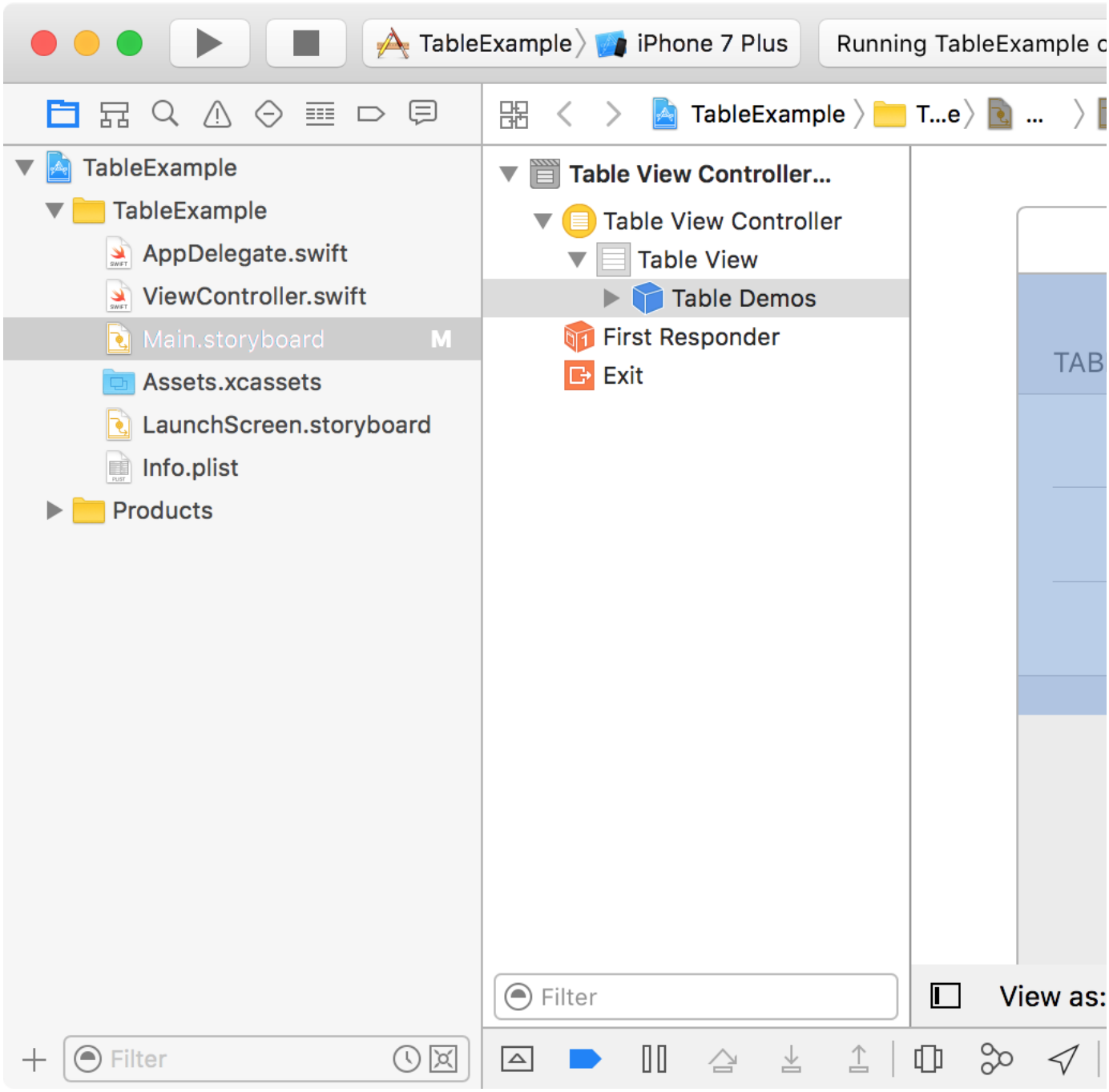
4. *Shift-Click* on the cell to see all views at that position and select *Table View*:



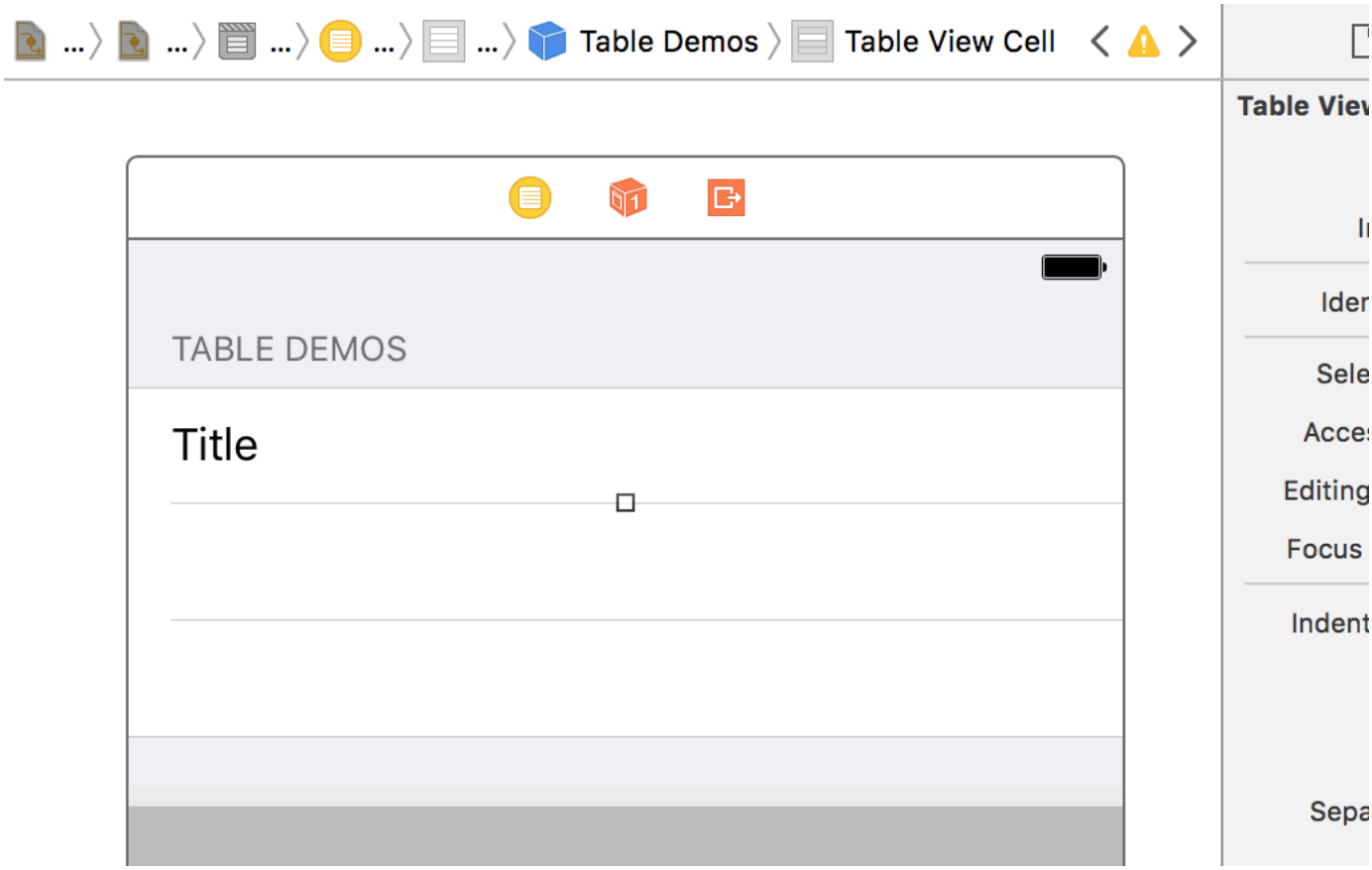
5. In the *Attributes Inspector* configure *Static Cells* for *Content* to configure the Table View cells in the storyboard instead of writing code to provide the data. Set the *Style* of the *Table View* to *Grouped*:



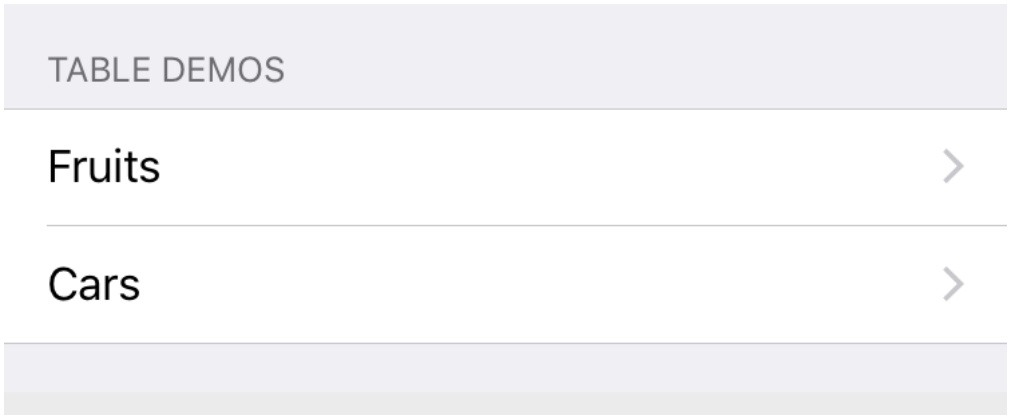
6. Select the section using the outline or with *Shift + Click*. Set *Table Demos* as *Header* text:



7. Select the first cell and configure *Basic* as *Style* for all cells. A *Basic Cell* has one *Label* by default:



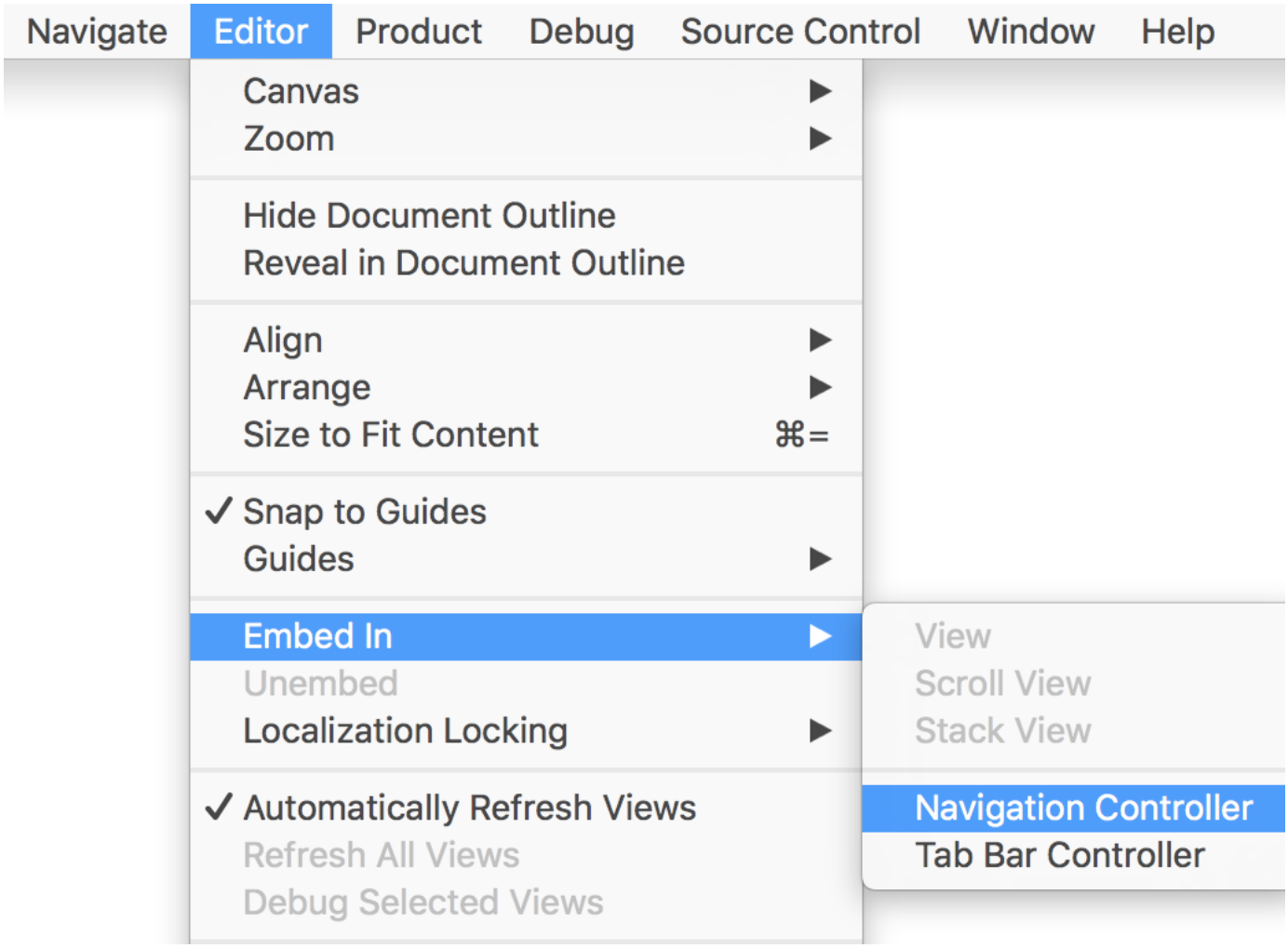
8. Configure *Basic* as *Style* for the second cell as well and delete the third cell. Configure the label texts like this:



9. Run the app with ⌘R.

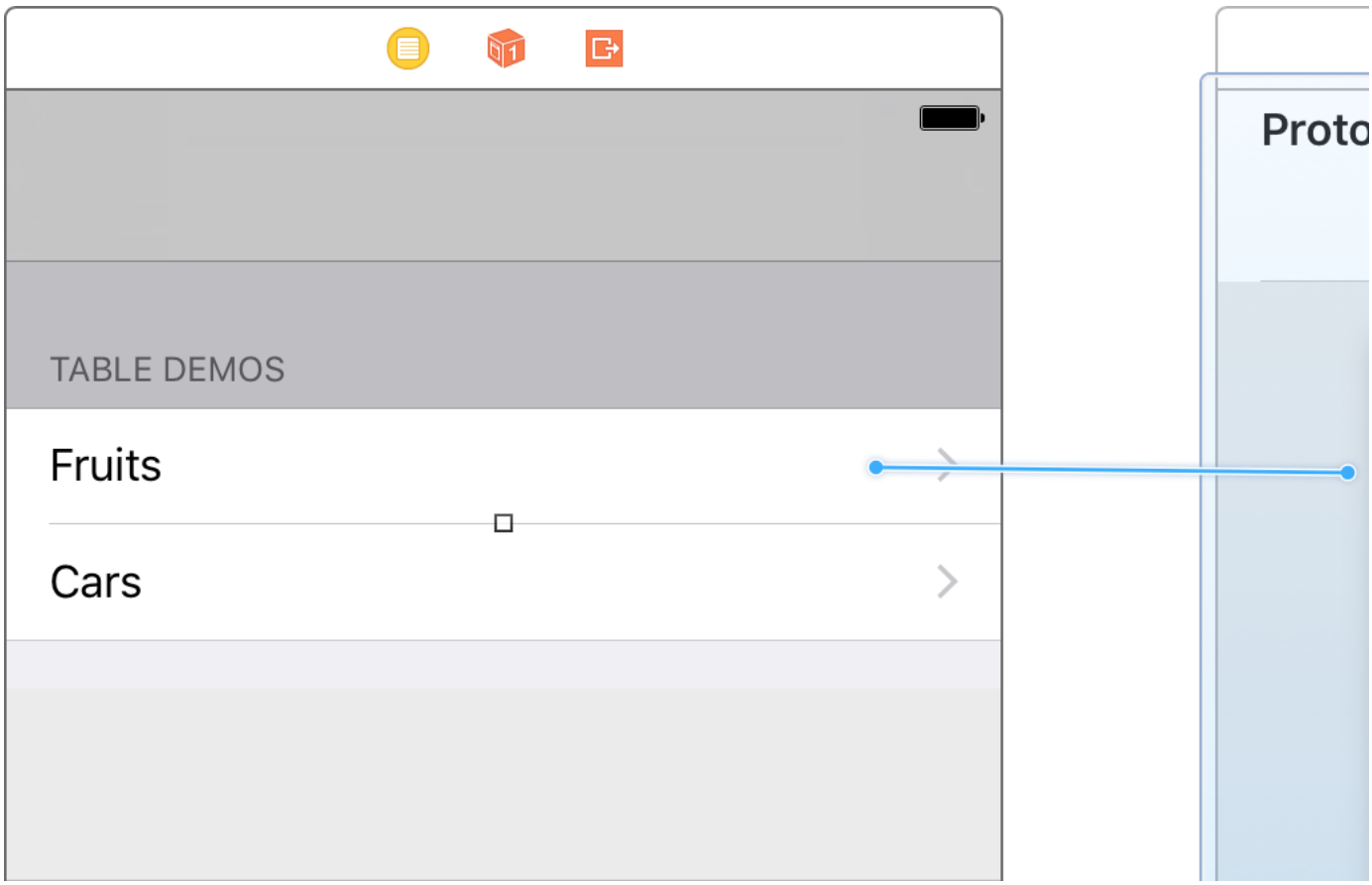
Creating a Storyboard Segue

1. Select the *Table View Controller* and use *Editor* > *Embed In* > *Navigation Controller* to embed the controller into a *UINavigationController* that manages a navigation stack:

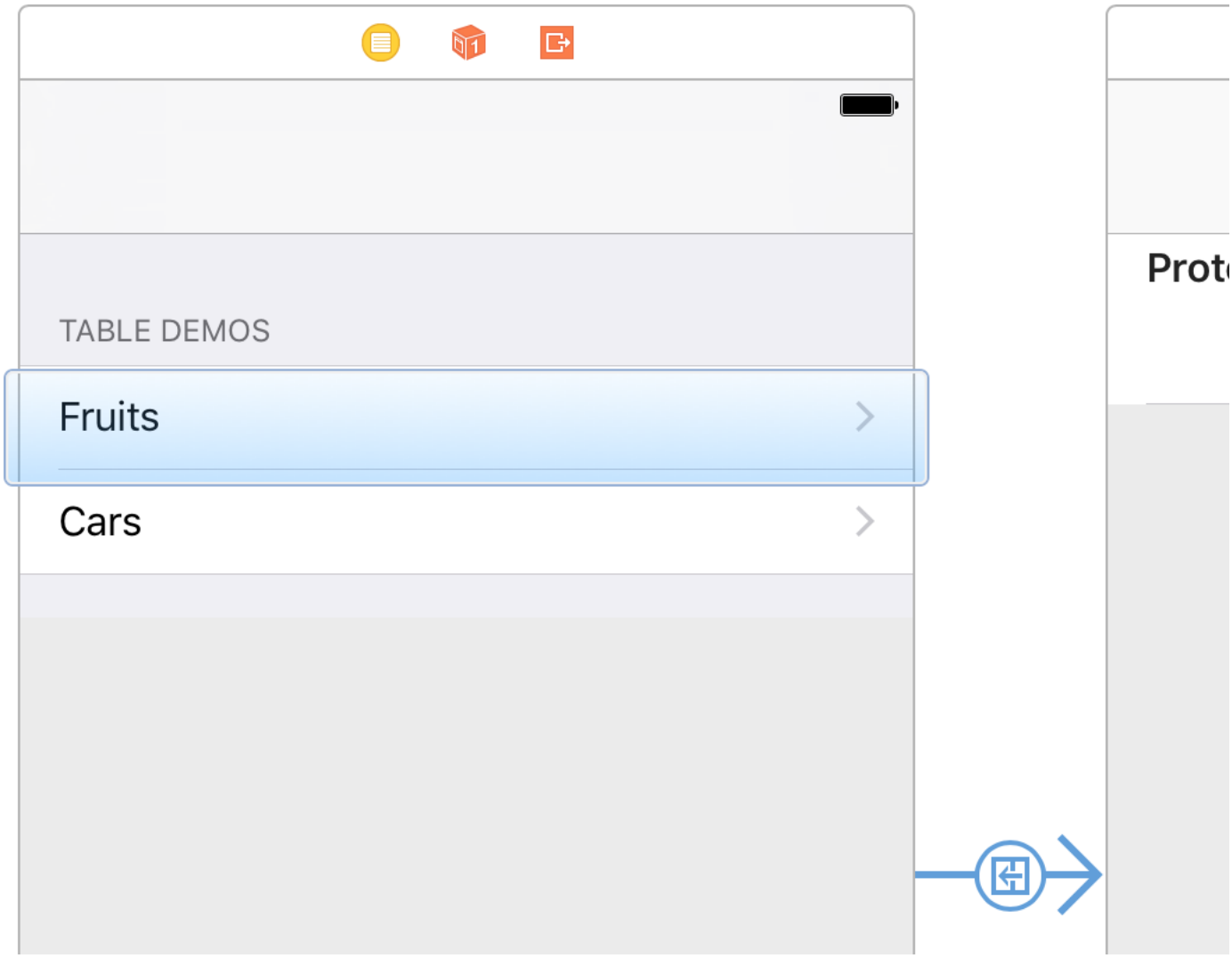


2. Drag in another *Table View Controller* from the *Object Library* behind the existing controller.

3. Drag a connection from the first cell of the controller to the second controller with *Ctrl* pressed down / with the right mouse key. Create a *Selection Segue > show*:

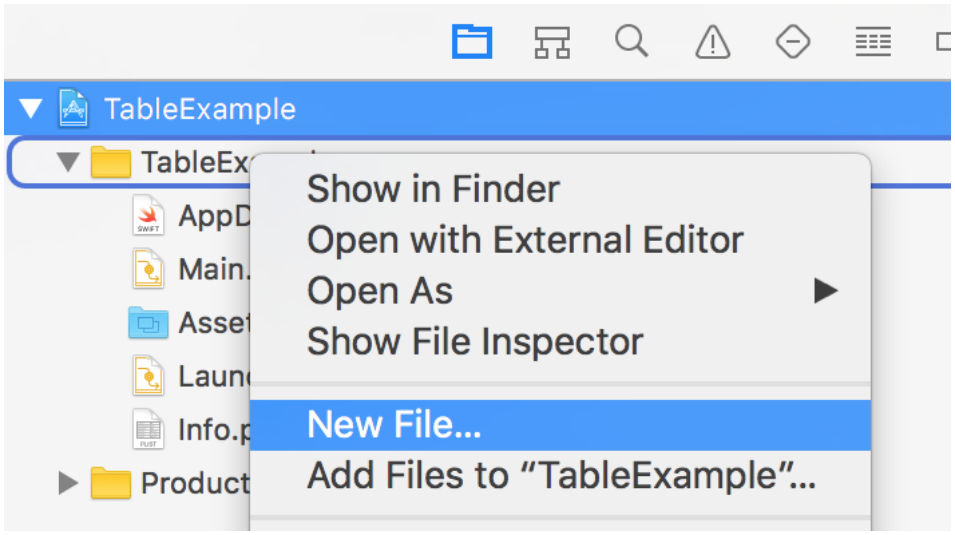


4. Select the segue to check that it connects the cell to the other Table View Controller:

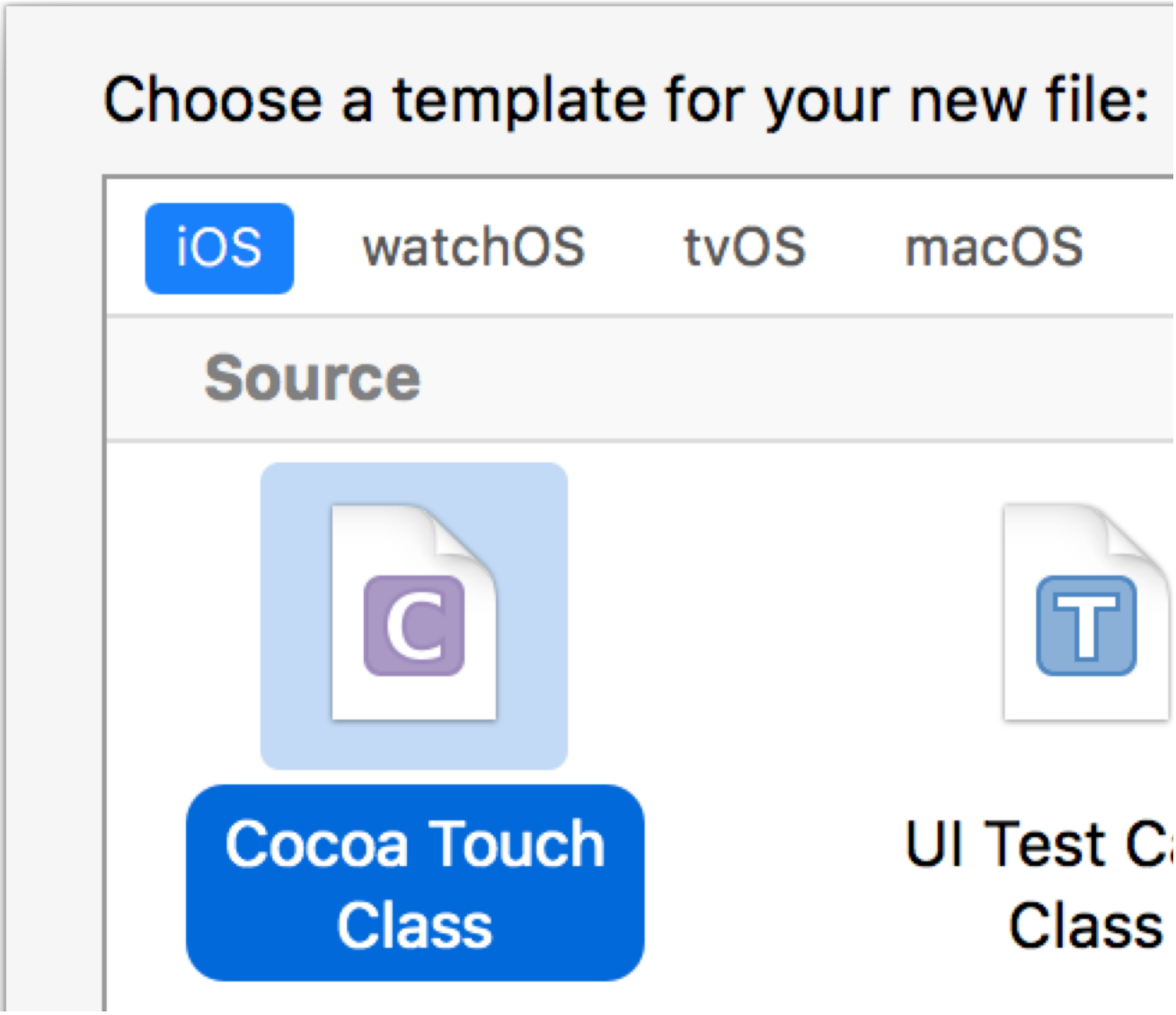


Provide section and row content programmatically

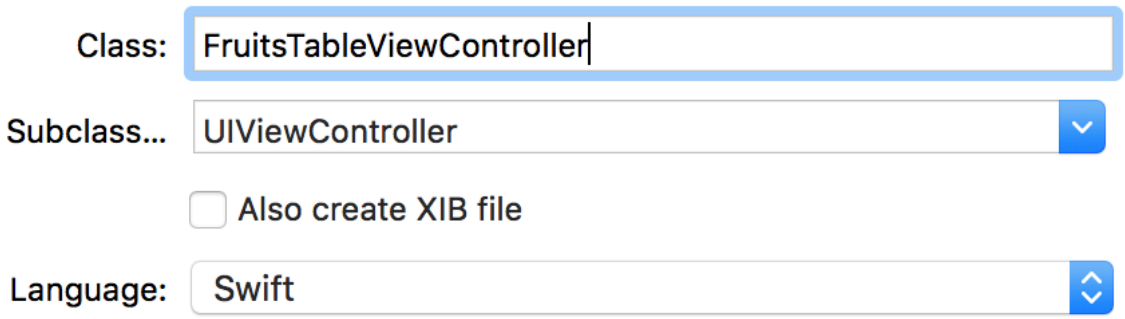
1. Create a new source code file:



2. Choose *Cocoa Touch Class* for *iOS*:



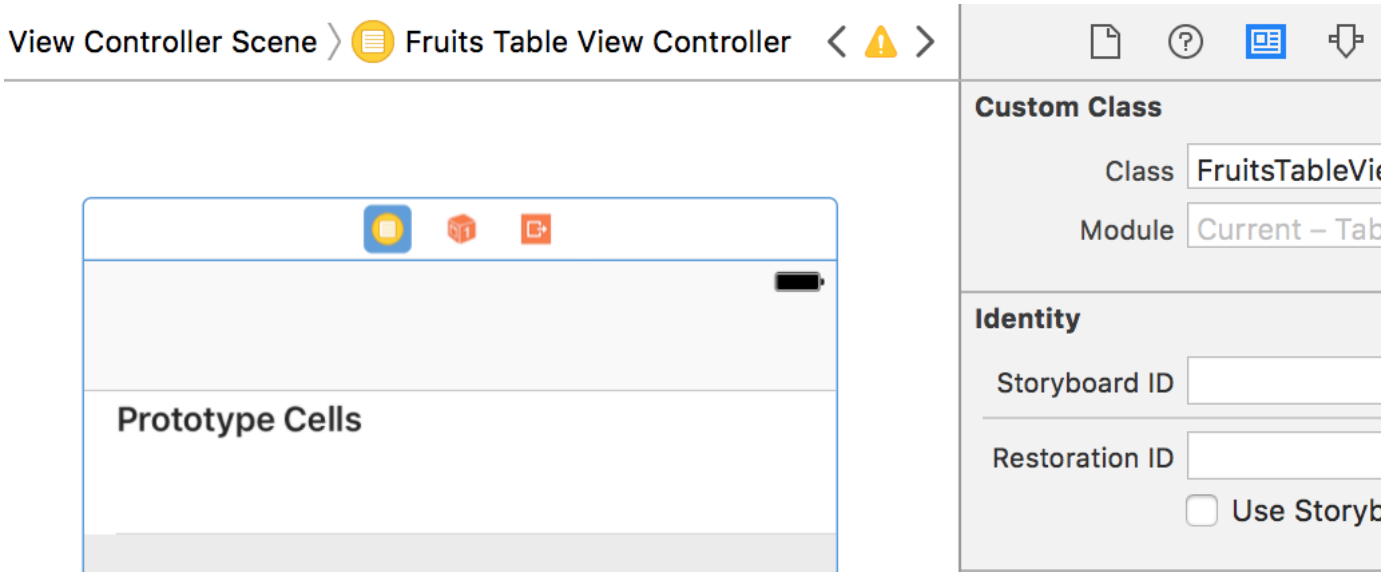
3. Create a subclass of *UITableViewController* in *Swift* and name it *FruitsTableViewController*:



4. In the *FruitsTableViewController* class, customize the *numberOfSectionsInTableView* and *tableView:numberOfRowsInSection:* methods from the *UITableViewDataSource* protocol to return a fixed number of sections and rows and remove the *#warning* comments:

```
// MARK: - Table view data source

override func numberOfSections(in tableView: UITableView) -> Int {
    return 3
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection sect
    return 5
}
```

5. Open the storyboard and set *FruitsTableViewController* as class for the second *Table View Controller*:

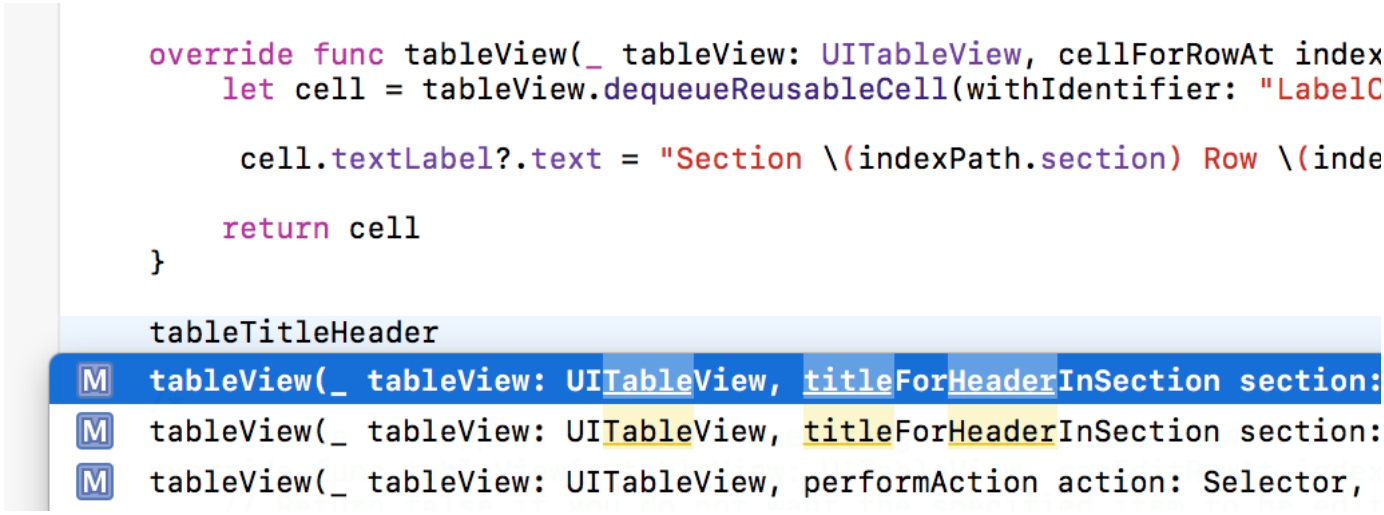


6. Select the *Prototype cell*, set *Basic* as *Style* and configure *LabelCell* as *Identifier*. The *identifier* is used in the controller implementation to create cells according to the prototype from the Storyboard:

7. Uncomment the *tableView:cellForRowAtIndexPath:* method and customize it to create cells according to the *Prototype cell* and configure the cell text to show the section and row numbers:

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "LabelCell", for: indexPath)

    cell.textLabel?.text = "Section \(indexPath.section) Row \(indexPath.row)"

    return cell
}
```

8. Overwrite the *tableView:titleForHeaderInSection:* method using the Xcode code completion:



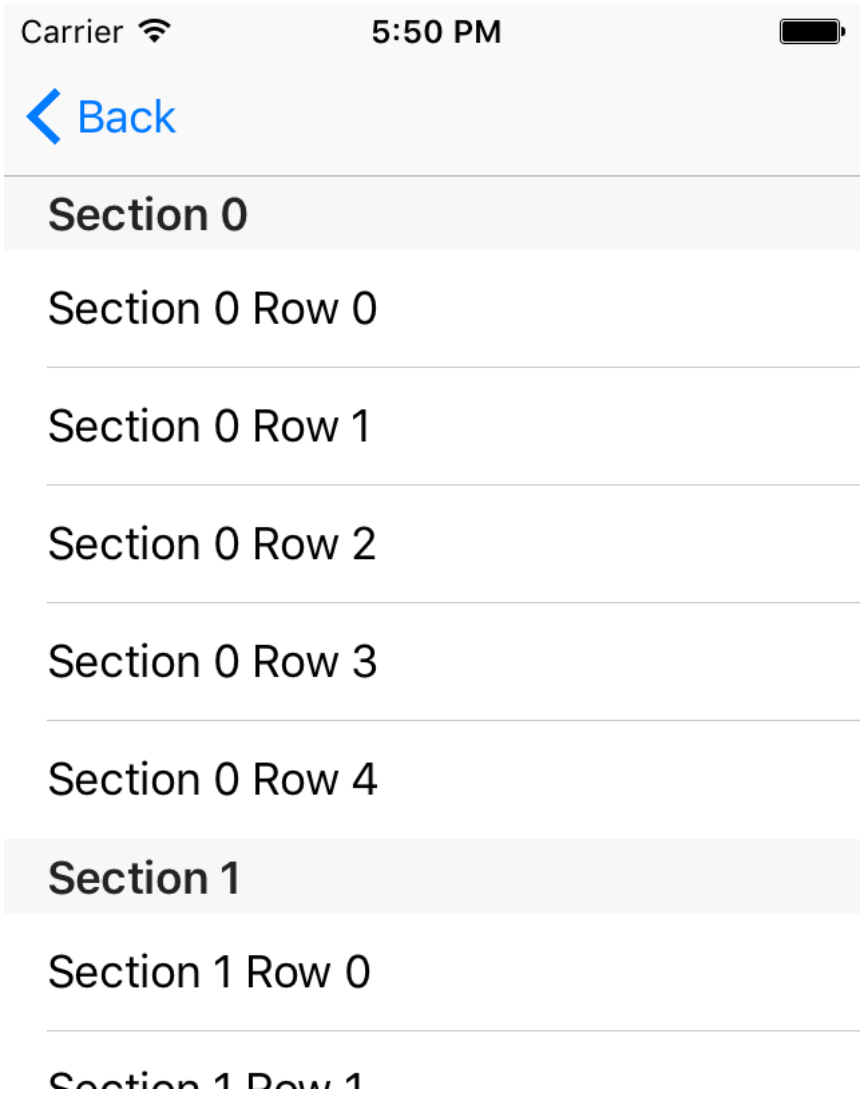
Implement the method to return a title according to the section number:

```
override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    return "Section \(section)"
}
```

9. Remove the remaining code in the class, this should be the code in the class:

```
import UIKit

class FruitsTableViewController: UITableViewController {

    // MARK: - Table view data source

    override func numberOfSections(in tableView: UITableView) -> Int {
        return 3
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 5
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "LabelCell", for: indexPath)

        cell.textLabel?.text = "Section \(indexPath.section) Row \(indexPath.row)"

        return cell
    }

    override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
        return "Section \(section)"
    }

}
```

10. Run the app with ⌘R:



Show data from a Swift Array

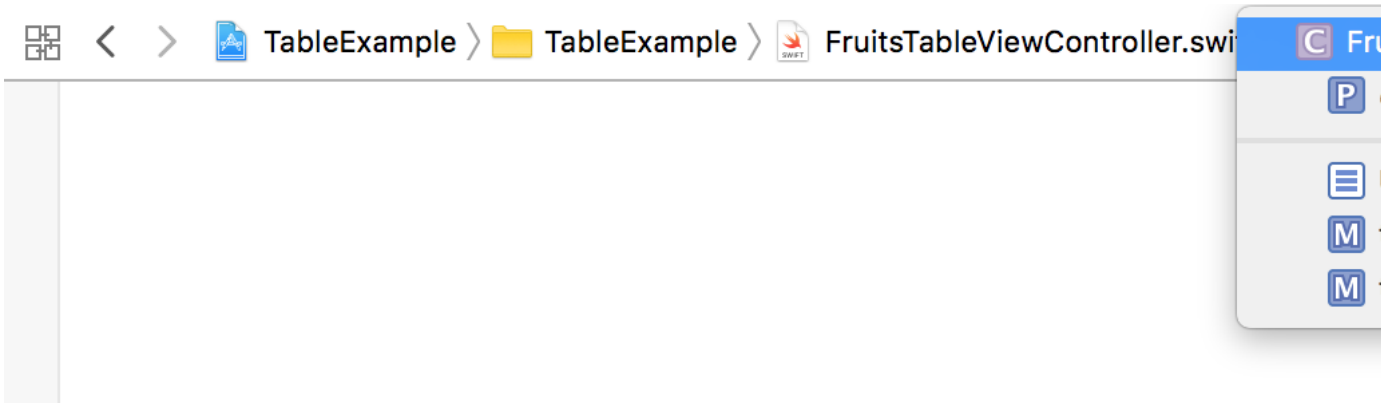
1. Customize the *FruitsTableViewController* to use data from a *Swift Array*:

```
class FruitsTableViewController: UITableViewController {

    var fruits = ["Apple", "Apricot", "Banana", "Blueberry", "Cantaloupe", "Cherry",
        "Clementine", "Coconut", "Cranberry", "Fig", "Grape", "Grapefruit",
```

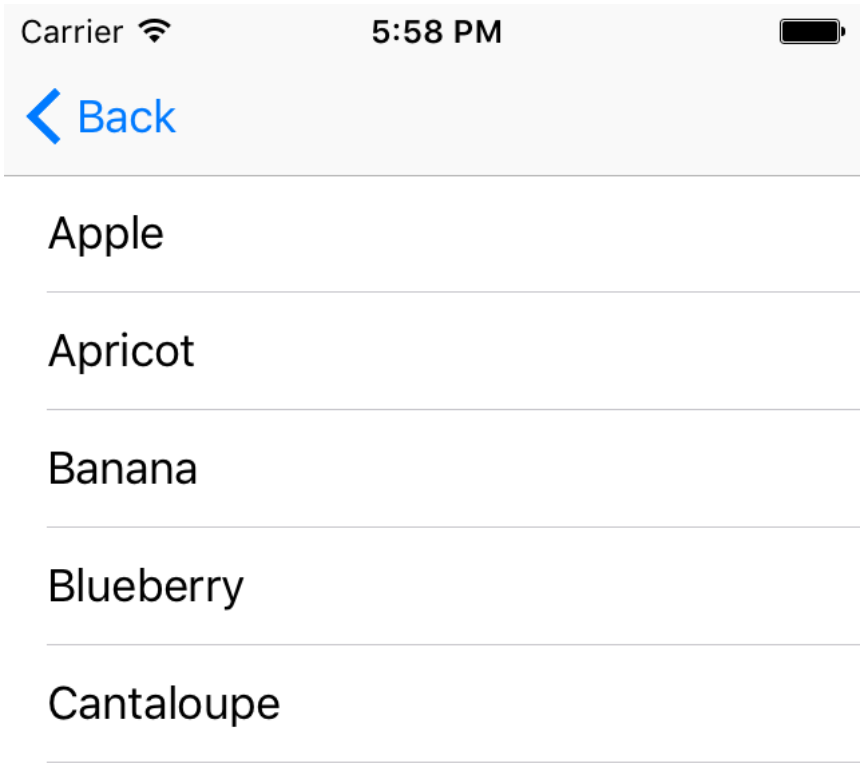
```
            "Kiwi fruit", "Lemon", "Lime", "Lychee", "Mandarine", "Mango",
            "Melon", "Nectarine", "Olive", "Orange", "Papaya", "Peach",
            "Pear", "Pineapple", "Raspberry", "Strawberry"]

        // MARK: - UITableViewDataSource

        override func numberOfSections(in tableView: UITableView) -> Int {
            return 1
        }

        override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
            return fruits.count
        }

        override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
            let cell = tableView.dequeueReusableCell(withIdentifier: "LabelCell", for: indexPath)

            cell.textLabel?.text = fruits[indexPath.row]

            return cell
        }

    }
```

Hint: You can also remove the *numberOfSections* method - by default tables have one section.

Hint: The *"// MARK: "* comment groups the methods, for example when choosing a method from the *Jump Bar*:

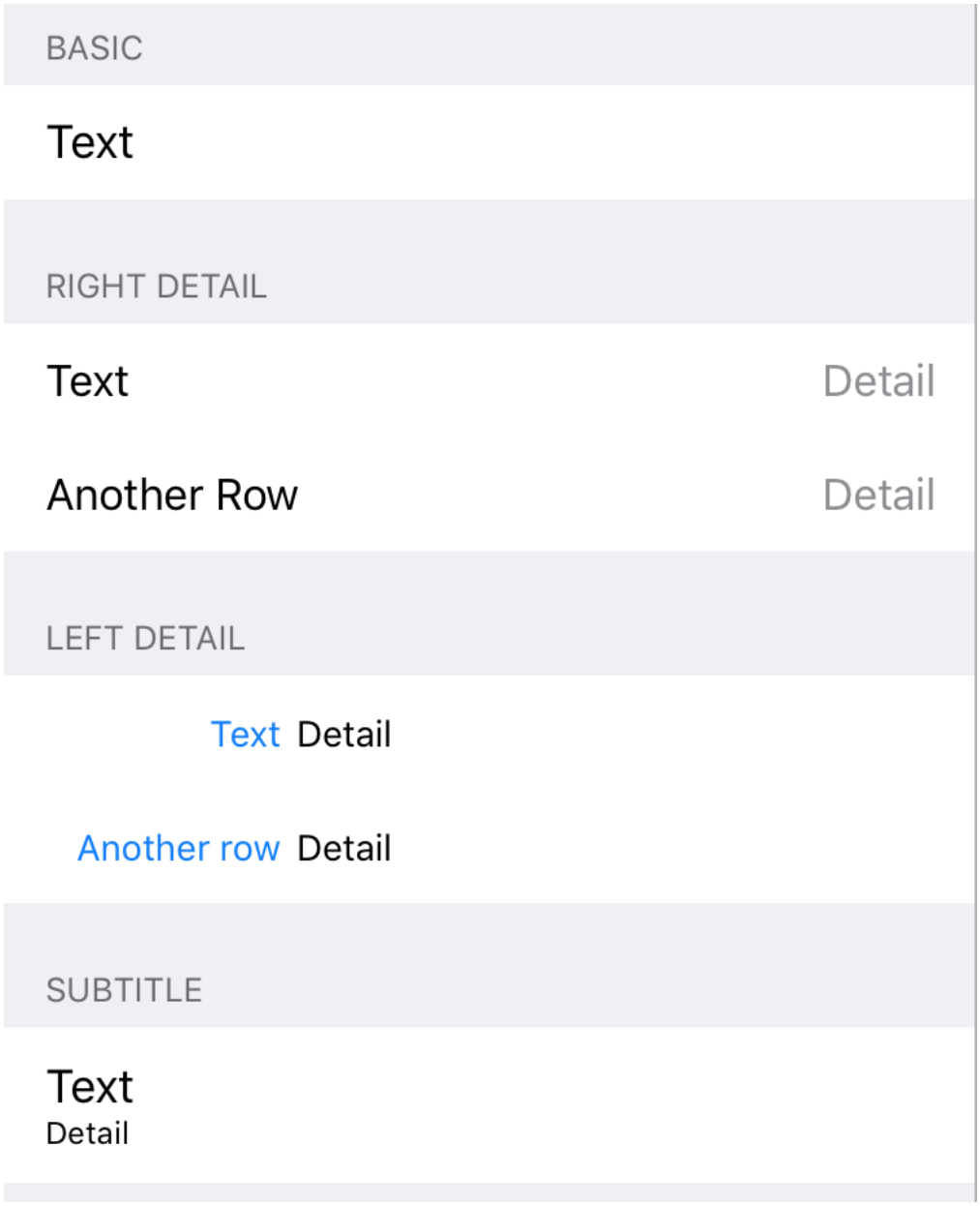


2. Run the app with ⌘R:

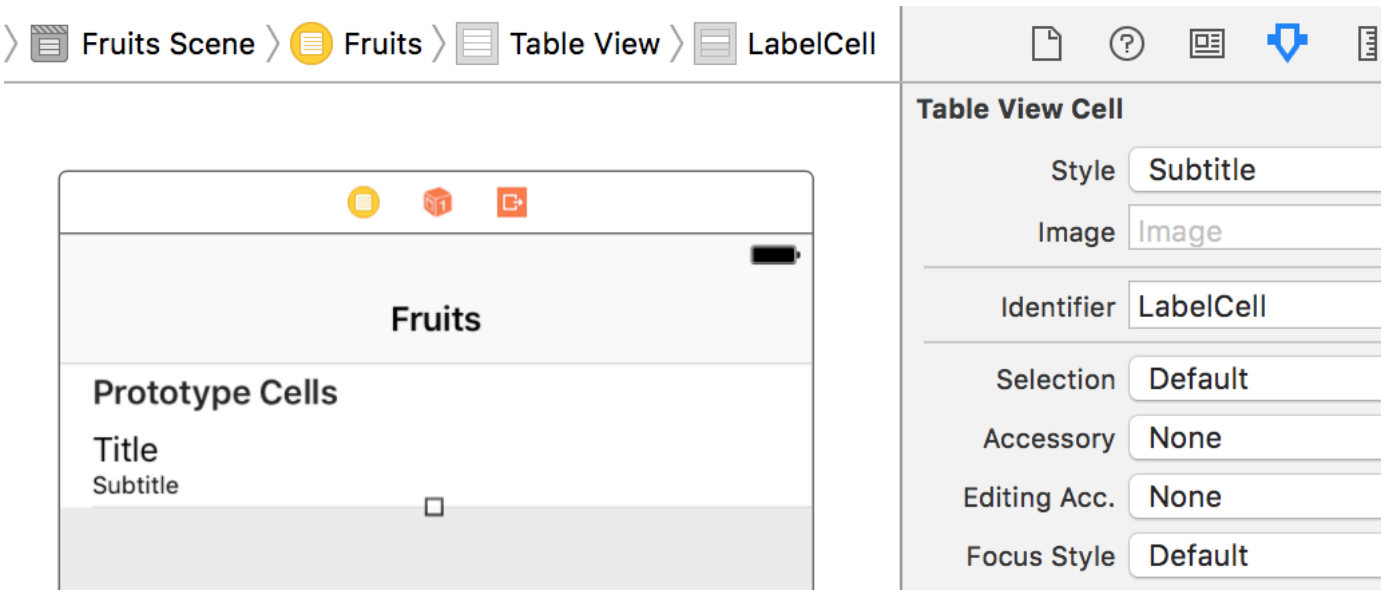


Configure cells using cell styles

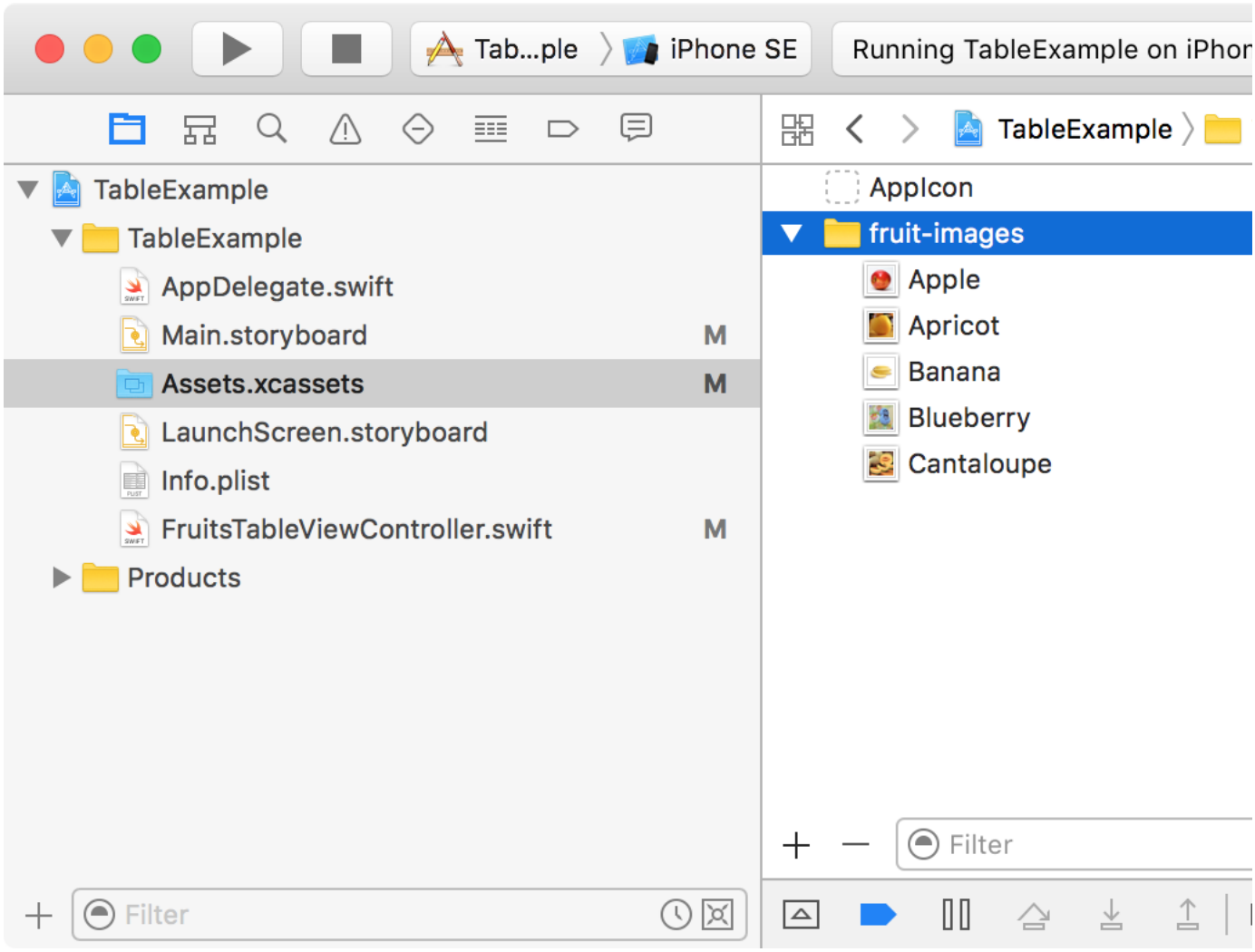
1. There are four different types of cell styles that have views out of the box - *Basic*, *Left/Right Detail* and *Subtitle*:



For the FruitTableViewController in the storyboard, select the first cell and set the style *Subtitle*:



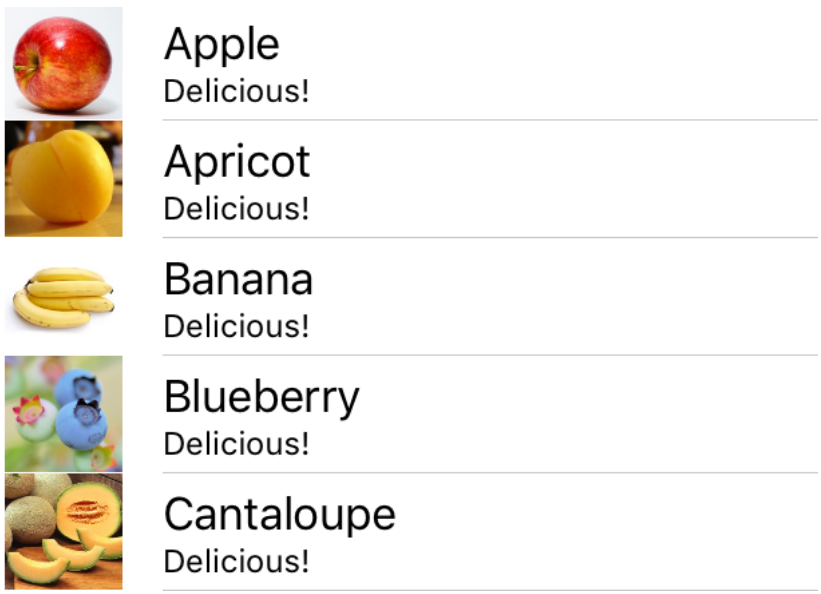
2. Download fruit-images.zip and add the images to the *Assets.xcassets* in your project:



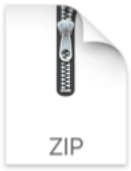
3. Extend the implementation of *tableView(cellForRowAt:)* in *FruitsTableViewController* to show a detail text and an image (all cells have an optional *imageView* that gets created as soon as the *imageView* property is accessed):

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "LabelCell", for: indexPath)

    let fruitName = fruits[indexPath.row]
cell.textLabel?.text = fruitName
cell.detailTextLabel?.text = "Delicious!"
cell.imageView?.image = UIImage(named: fruitName)

    return cell
}
```

4. Run the app and check that the table looks like this:



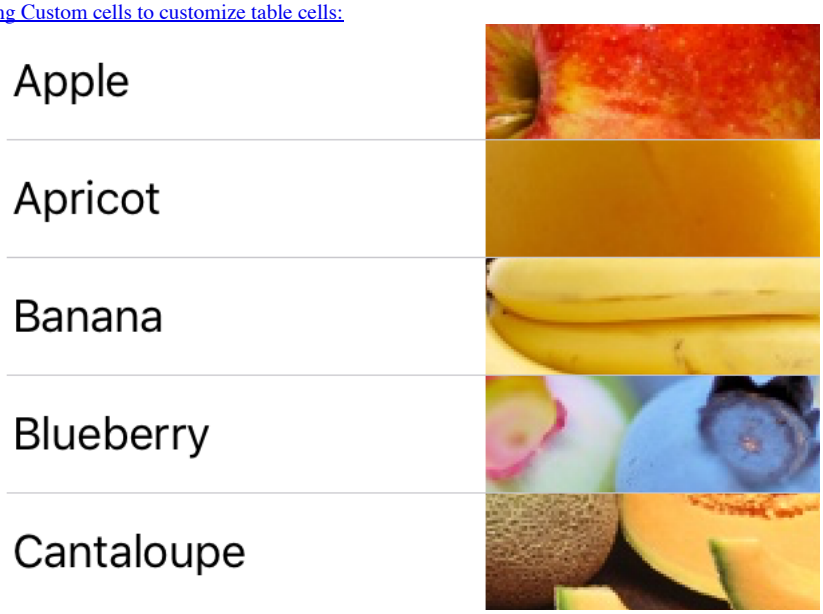
Example code



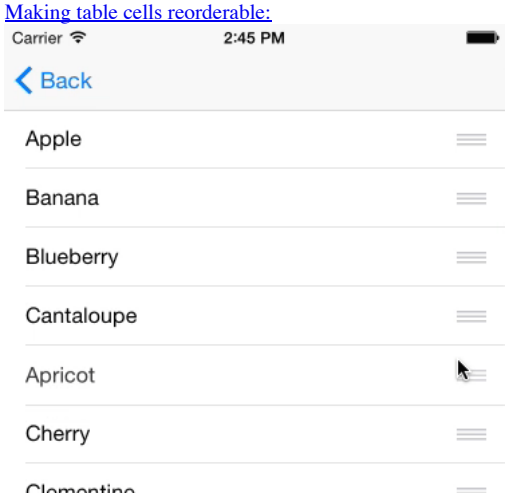
TableExample.zip

More UITableViewController tutorials

- Using Custom cells to customize table cells:



- Making table cells reorderable:



More information

- *Table View Programming Guide for iOS*
- *Sample Code: TableView Fundamentals for iOS*
- Fruit images are available under a *Creative Common* license at *Wikimedia Commons*, see *copyright.txt* in fruit-images.zip

Ralf Ebert » Tutorials »

- Swift Tutorial
- iPhone-Apps programmieren
- JSON-Daten in einem UITableView anzeigen
- Watch OS 2 Tutorial: WatchKit Connectivity
- Watch OS 2 Tutorial: Complications
- iOS & Swift Tutorial: How to develop an iPhone app

- iOS & Swift Tutorial: UITableViewController
- iOS & Swift Tutorial: Multipeer Connectivity
- Rails Deployment Tutorial
- Git Tutorial

Buch „iOS-10-Apps entwickeln mit Swift und Xcode 8“

About the author

Ralf Ebert is an independent software developer and trainer for Mac OS X and iOS. He makes the Page Layers and Straight ahead apps and conducts iOS trainings in Germany since 2009.