

Developing Applications for iOS



Lecture 7: Table Views

Prof. Dr. Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Content

- UITableView
- Creating Table View MVCs
- UITableViewDataSource
- UITableViewDelegate

UITableView

Very important class for displaying data in a table

- One-dimensional table.
- It's a subclass of `UIScrollView`.
- Table can be a static or dynamic list of items.
- Lots and lots of customization via a `dataSource` protocol and a `delegate` protocol.
- Very efficient even with very large sets of data.

UITableView

Displaying multi-dimensional tables

- Usually done via a `UINavigationController` containing multiple MVC's where View is `UITableView`.
- Or, via the `UICollectionView` from iOS 6.0. Collection views provide the same general function as table views, except that a collection view is able to support more layouts.
- Collection views support customizable layouts that can be used to implement multi-column grids, circular layouts, and many more.

Kinds of UITableViews

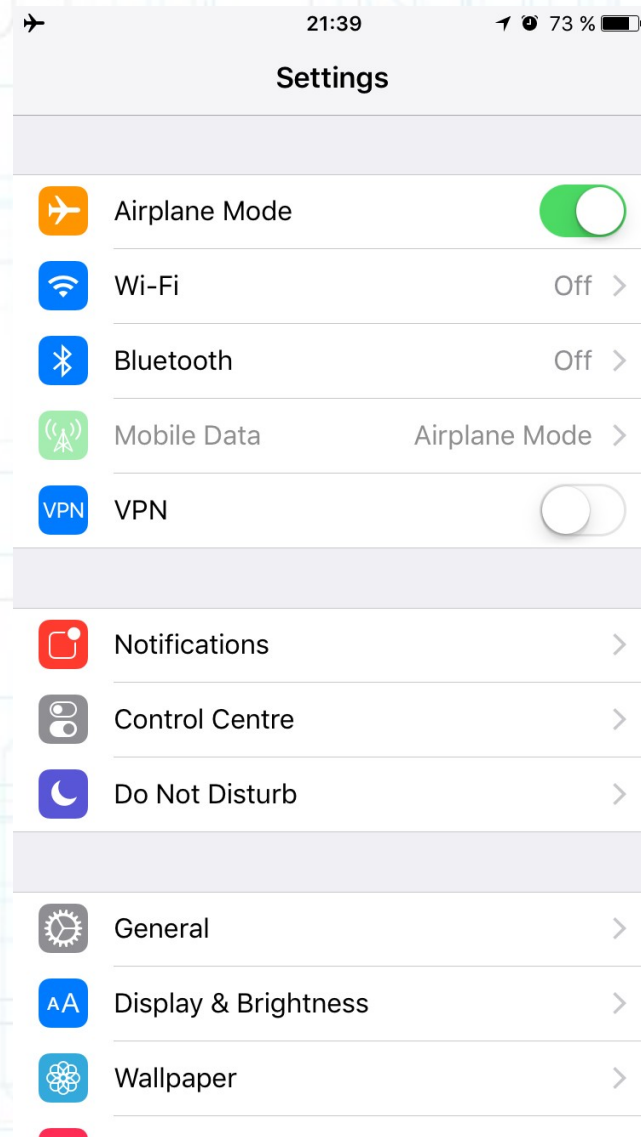
- Plain or Grouped.
- Static or Dynamic.
- Divided into sections or not.
- Different formats for each row in the table (including completely customized).

UITableView

UITableViewStylePlain

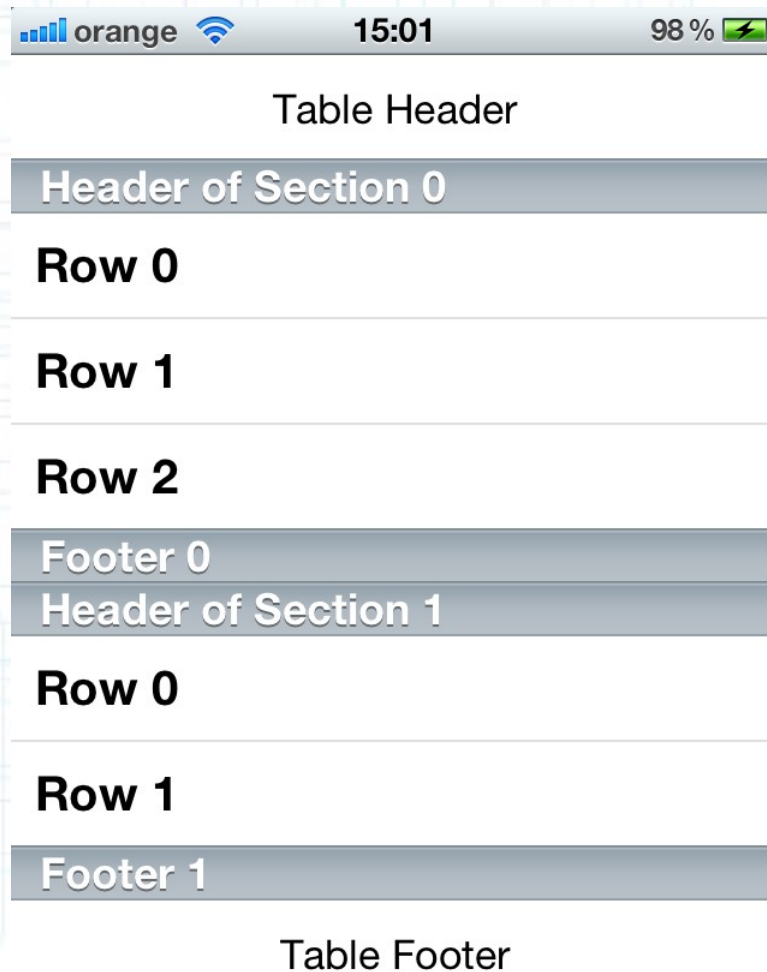


UITableViewStyleGrouped



UITableView

Plain Style

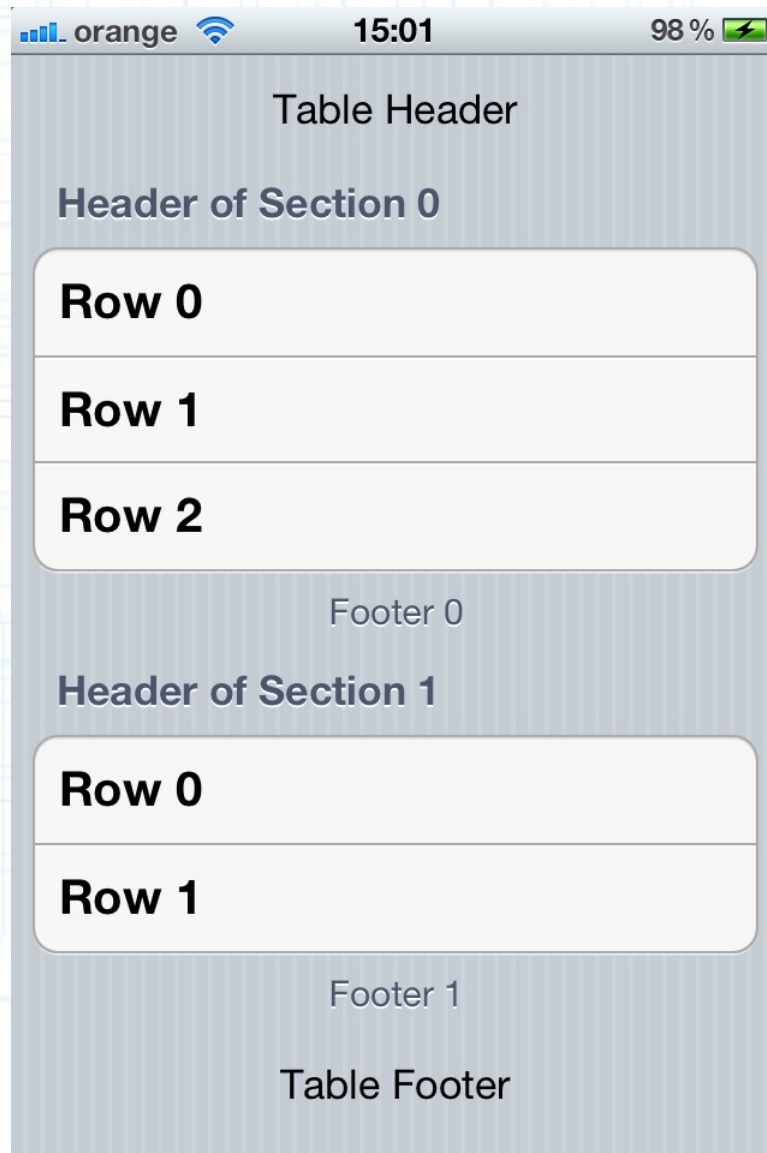


The image shows a screenshot of an iPhone's status bar at the top, displaying 'orange' as the carrier, a Wi-Fi signal icon, the time '15:01', and a battery level of '98 %' with a charging icon. Below the status bar is a UITableView with a Plain Style. The table contains two sections. The first section has a header 'Header of Section 0' and three rows labeled 'Row 0', 'Row 1', and 'Row 2'. The second section has a header 'Header of Section 1' and two rows labeled 'Row 0' and 'Row 1'. Each section also has a footer: 'Footer 0' for the first section and 'Footer 1' for the second section. The table is set against a light blue grid background.

Table Header	
Header of Section 0	
Row 0	
Row 1	
Row 2	
Footer 0	
Header of Section 1	
Row 0	
Row 1	
Footer 1	
Table Footer	


UITableView

Grouped Style



UITableView


No Sections



orange 15:30 100%

Barcelona
Spain
Bucharest
Romania
Istanbul
Turkey
Madrid
Spain
Mersin
Turkey
Milano
Italy
Nice
France
Paris
France
Rome
Italy
Toulouse
France

Sections



orange 15:34 100%

France
Nice
France
Paris
France
Toulouse
France
Italy
Milano
Italy
Rome
Italy
Romania
Bucharest
Romania
Spain
Barcelona
Spain
Madrid
Spain
Turkey

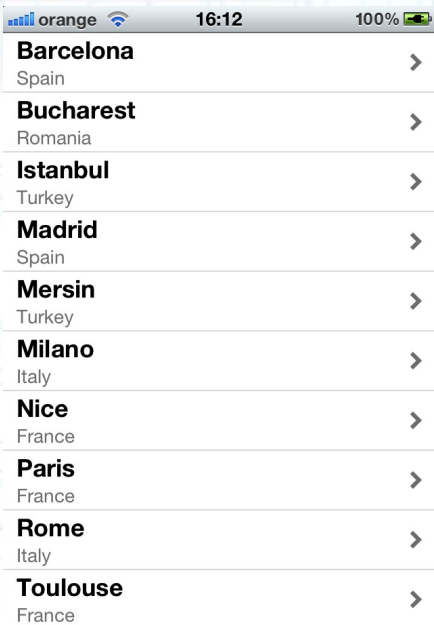
Cell Type

Subtitle

UITableViewCellStyleSubtitle

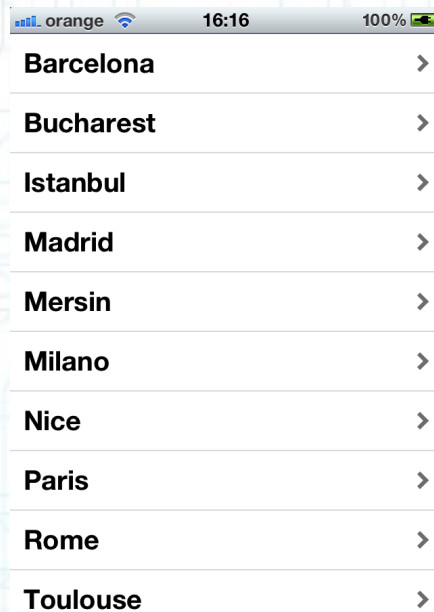
Right Detail

UITableViewCellStyleValue1



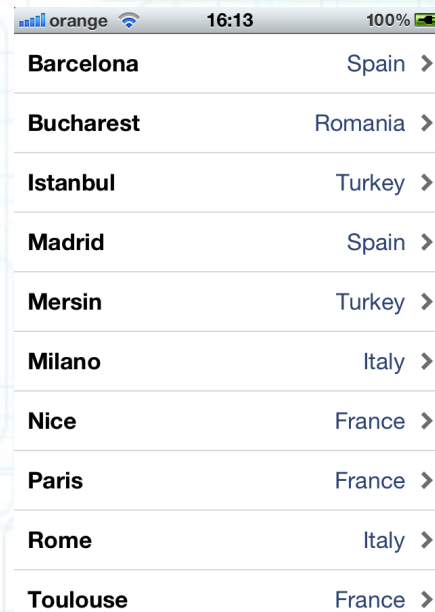
A screenshot of an iPhone screen displaying a list of cities. The status bar at the top shows 'orange' carrier, signal strength, Wi-Fi, time '16:12', and 100% battery. The list contains eight items, each with a city name in bold and its country below it, followed by a chevron icon.

Barcelona Spain	>
Bucharest Romania	>
Istanbul Turkey	>
Madrid Spain	>
Mersin Turkey	>
Milano Italy	>
Nice France	>
Paris France	>
Rome Italy	>
Toulouse France	>



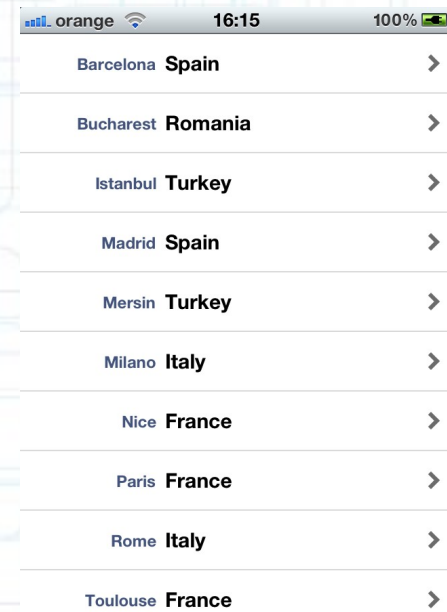
A screenshot of an iPhone screen displaying a list of cities. The status bar at the top shows 'orange' carrier, signal strength, Wi-Fi, time '16:16', and 100% battery. The list contains eight items, each with a city name in bold and its country below it, followed by a chevron icon.

Barcelona Spain	>
Bucharest Romania	>
Istanbul Turkey	>
Madrid Spain	>
Mersin Turkey	>
Milano Italy	>
Nice France	>
Paris France	>
Rome Italy	>
Toulouse France	>



A screenshot of an iPhone screen displaying a list of cities. The status bar at the top shows 'orange' carrier, signal strength, Wi-Fi, time '16:13', and 100% battery. The list contains eight items, each with a city name in bold, the country in blue, and a chevron icon.

Barcelona Spain	>
Bucharest Romania	>
Istanbul Turkey	>
Madrid Spain	>
Mersin Turkey	>
Milano Italy	>
Nice France	>
Paris France	>
Rome Italy	>
Toulouse France	>



A screenshot of an iPhone screen displaying a list of cities. The status bar at the top shows 'orange' carrier, signal strength, Wi-Fi, time '16:15', and 100% battery. The list contains eight items, each with the city name in blue, the country in bold, and a chevron icon.

Barcelona Spain	>
Bucharest Romania	>
Istanbul Turkey	>
Madrid Spain	>
Mersin Turkey	>
Milano Italy	>
Nice France	>
Paris France	>
Rome Italy	>
Toulouse France	>

Basic

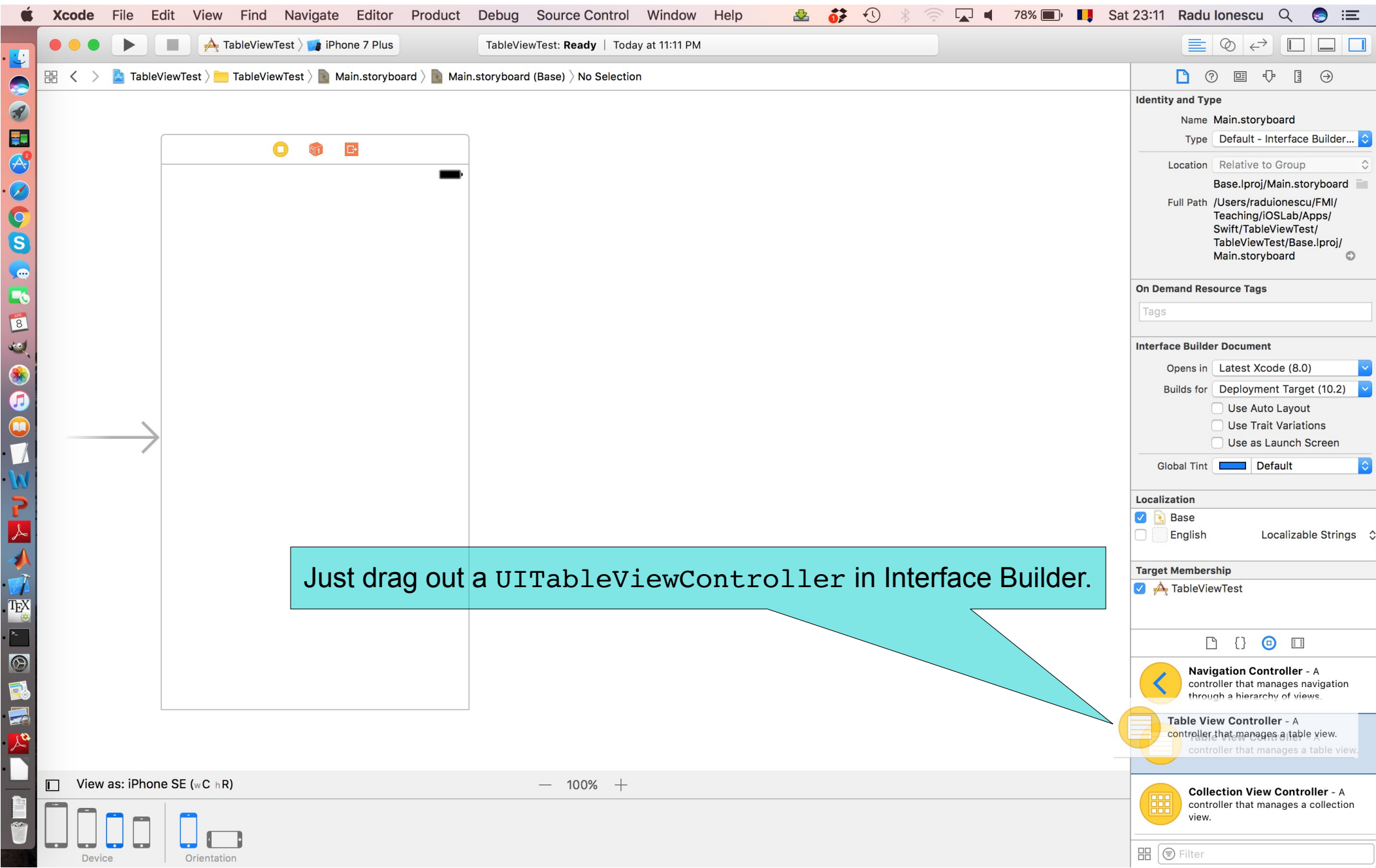
UITableViewCellStyleDefault

Left Detail

UITableViewCellStyleValue2

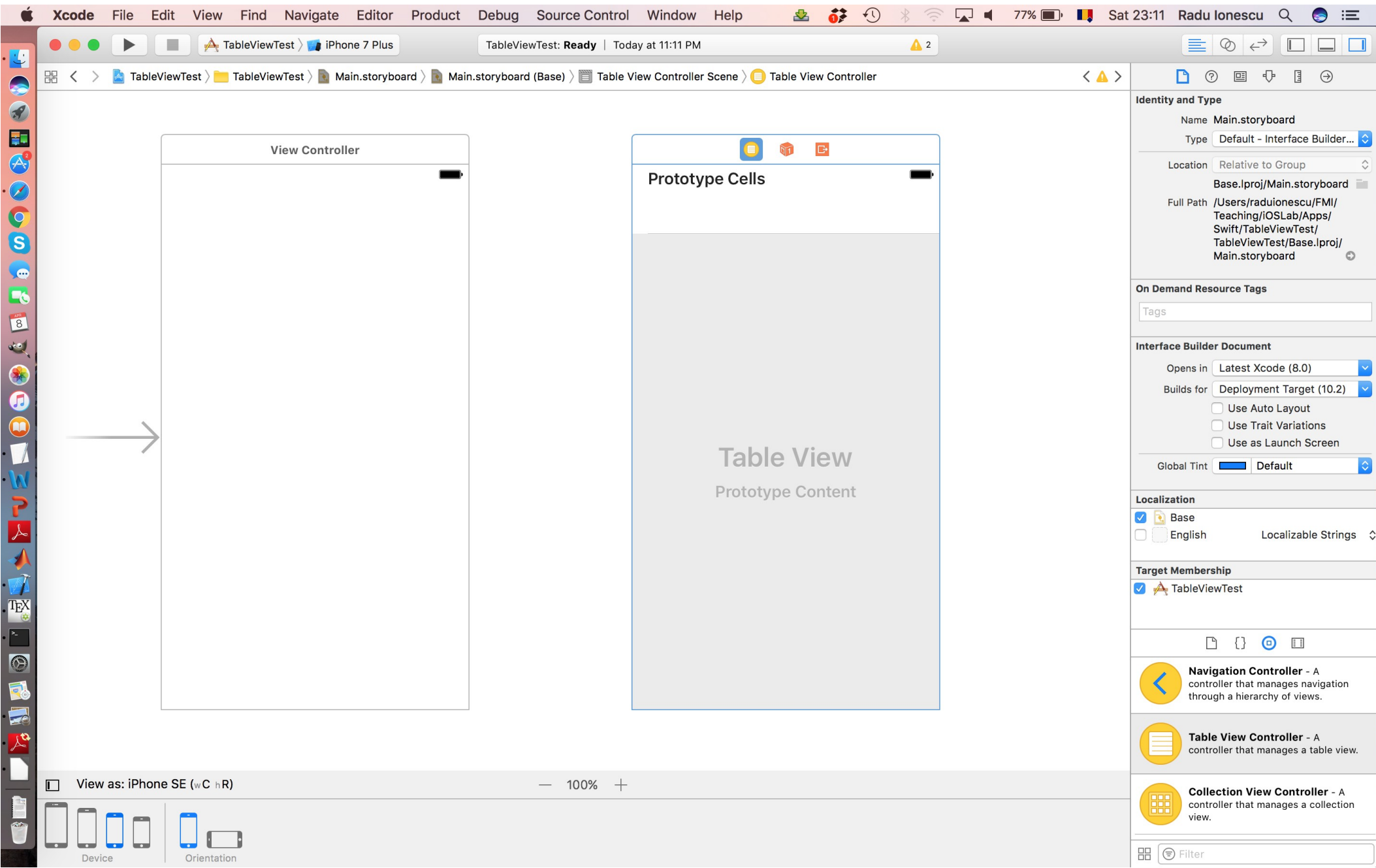
Creating Table View MVCs

`UITableViewController` is the iOS class used as the base class for MVC's that display `UITableViews`.



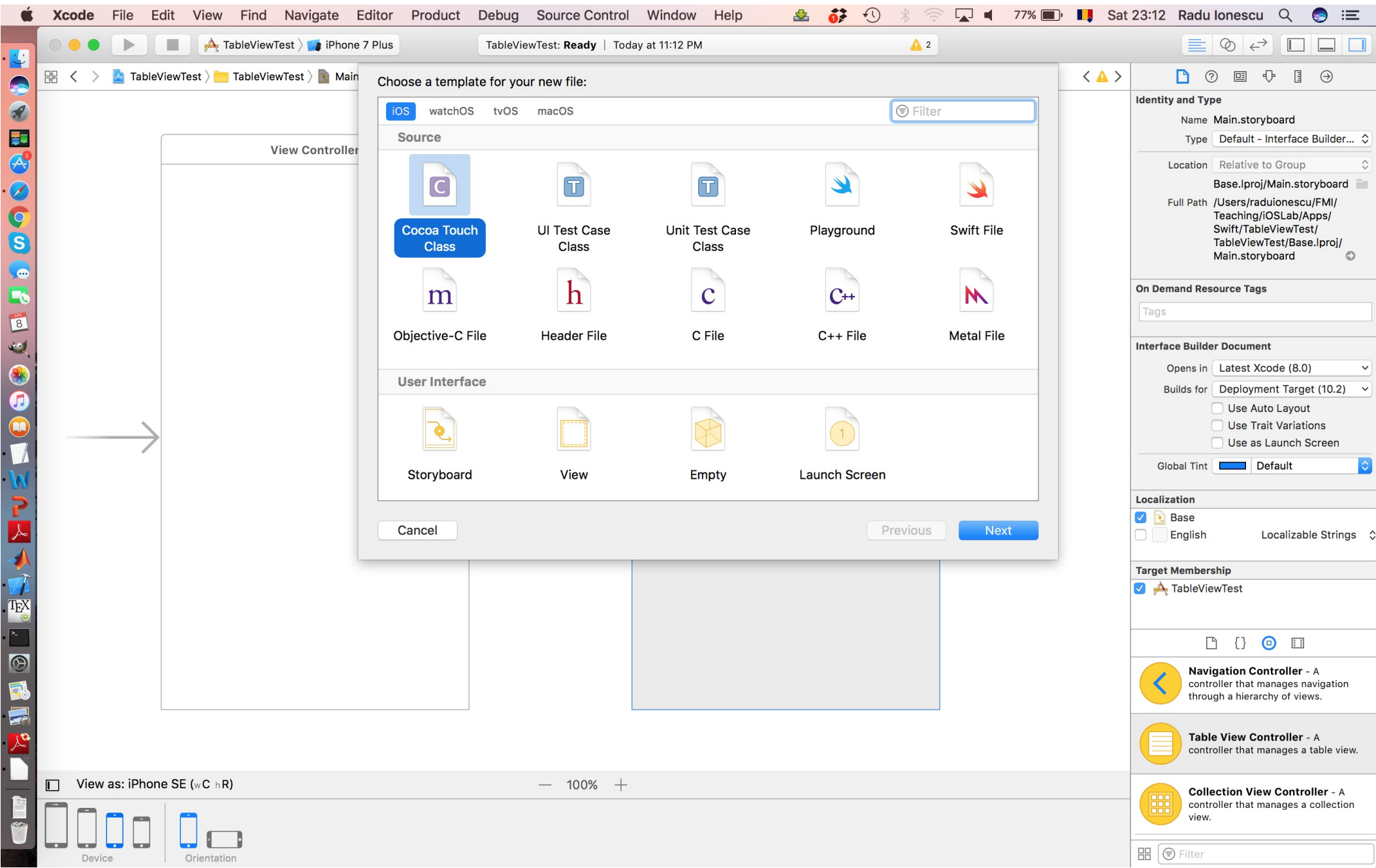
Creating Table View MVCs

`UITableViewController` is the iOS class used as the base class for MVC's that display `UITableViews`.



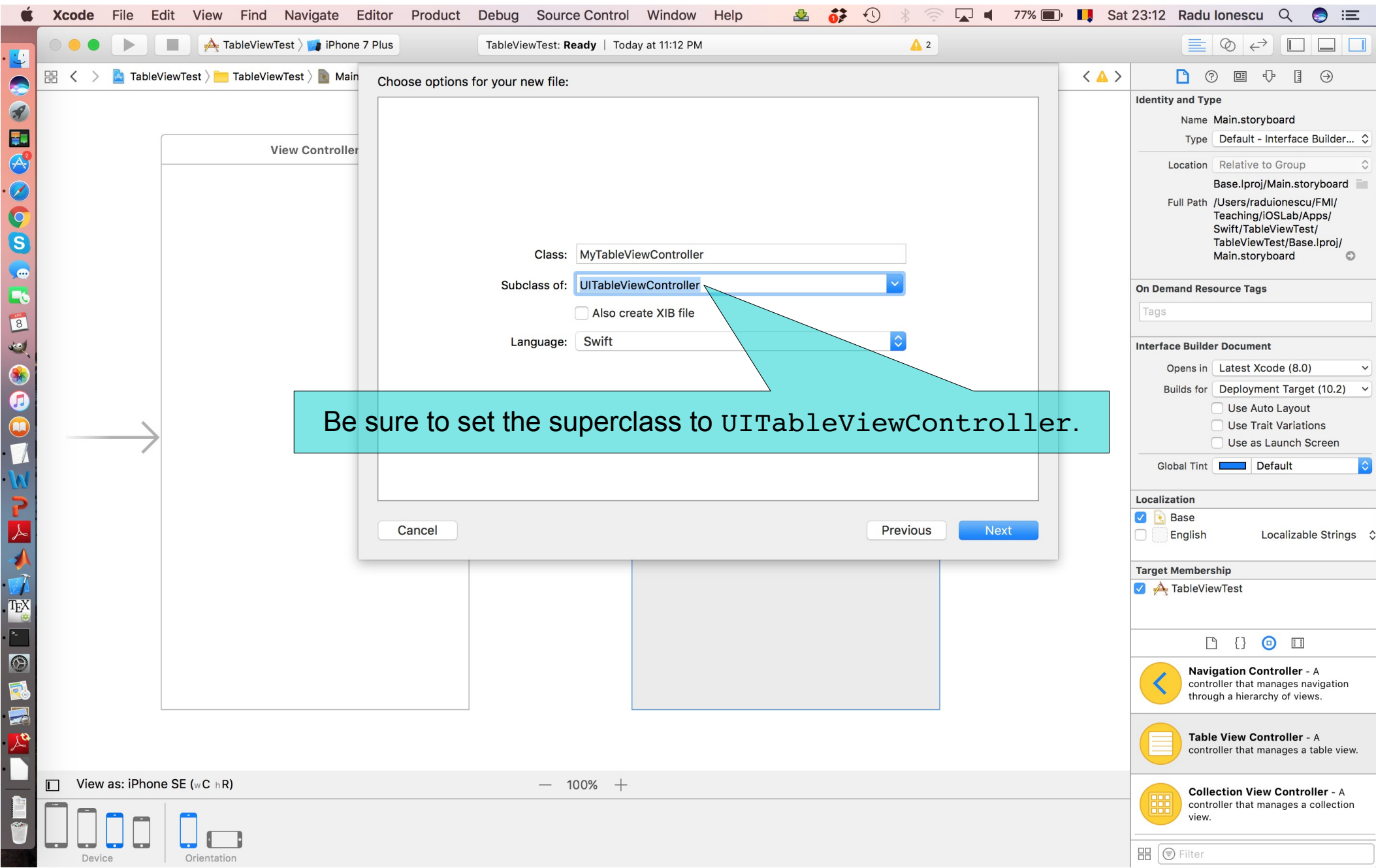
Creating Table View MVCs

Choose “New File ...” from the File menu to create a custom subclass of `UITableViewController`.



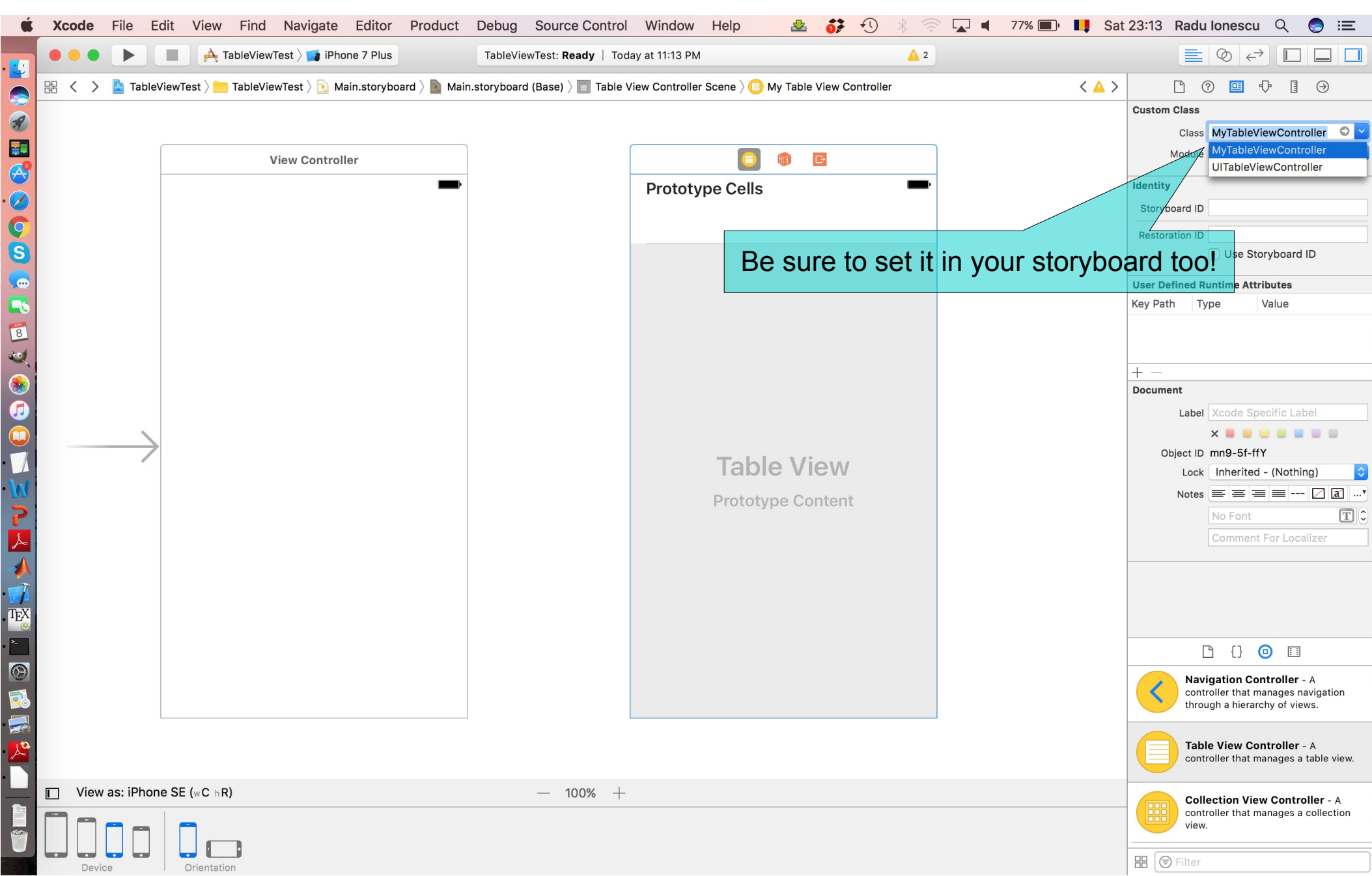
Creating Table View MVCs

Choose “New File ...” from the File menu to create a custom subclass of `UITableViewController`.



Creating Table View MVCs

Choose “New File ...” from the File menu to create a custom subclass of `UITableViewController`.



Creating Table View MVCs

You can customize both the look of the table view and its cells from Interface Builder.

The screenshot displays the Xcode IDE with a storyboard open. A callout box with a light blue background and black text points to a 'Table View' component in the storyboard. The text inside the callout reads: "Click on the table view (not the table view controller) to see its properties in the Inspector." The storyboard shows a 'View Controller' on the left and a 'Table View' on the right. The 'Table View' is labeled 'Table View' and 'Prototype Content'. The right-hand 'Inspector' panel is open, showing the 'Table View' properties. The 'Content' is set to 'Dynamic Prototypes'. The 'Style' is 'Plain'. The 'Selection' is 'Single Selection'. The 'Scrolling' options are 'Scrolling Enabled', 'Bounces', and 'Bounce Vertically'. The 'Navigation Controller' and 'Table View Controller' are also listed in the 'Inspector' panel.

Table View

- Content: Dynamic Prototypes
- Prototype Cells: 1
- Style: Plain
- Separator: Default
- Separator Inset: Default
- Selection: Single Selection
- Editing: No Selection During Editing
- Section Index
 - Display Limit: 0
 - Text: Default
 - Background: Default
 - Tracking: Default
- Scroll View
 - Style: Default
 - Scroll Indicat...
 - ☒ Shows Horizontal Indicator
 - ☒ Shows Vertical Indicator
 - Scrolling
 - ☒ Scrolling Enabled
 - ☐ Paging Enabled
 - ☐ Direction Lock Enabled
 - Bounce
 - ☒ Bounces
 - ☐ Bounce Horizontally
 - ☒ Bounce Vertically

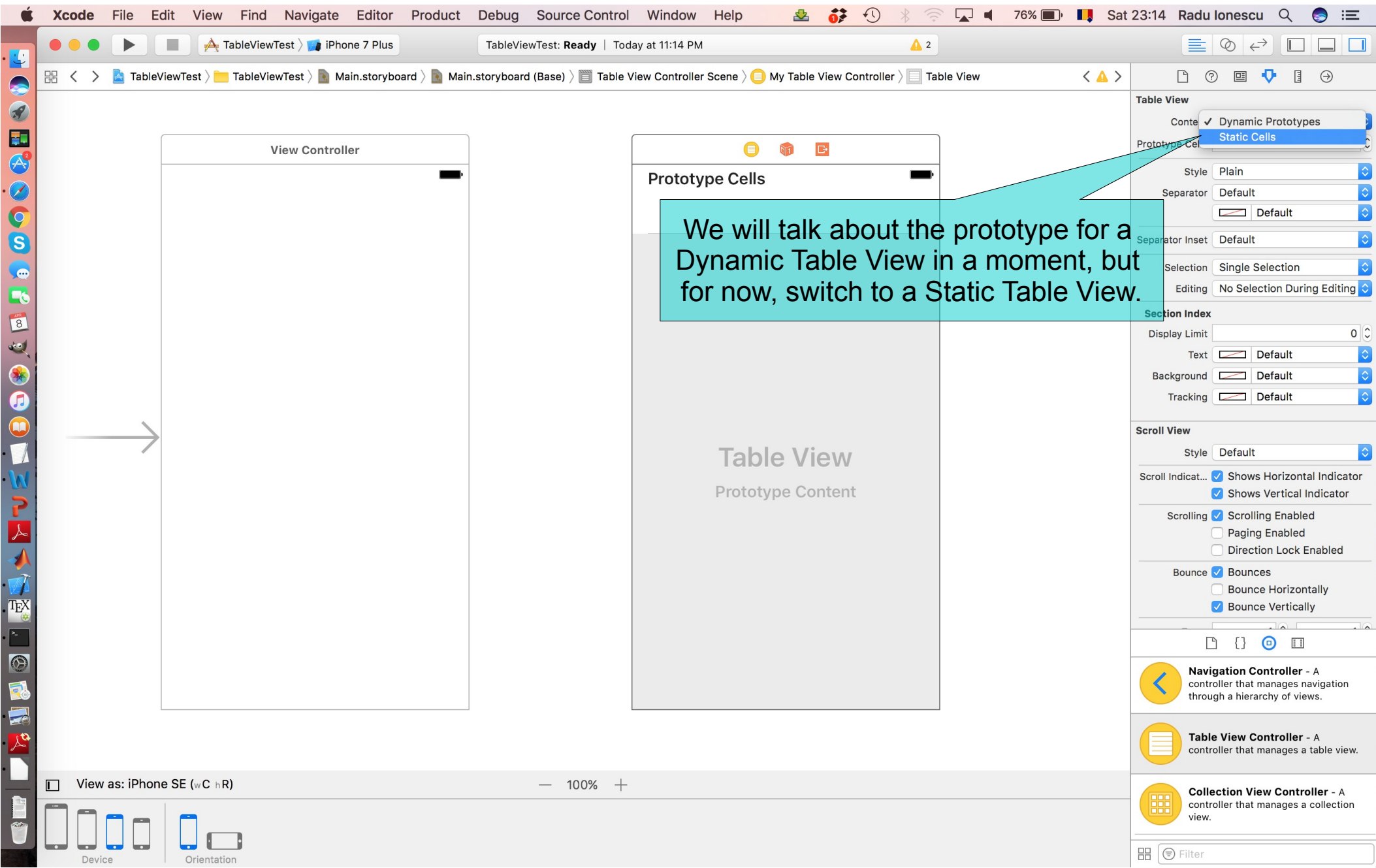
Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

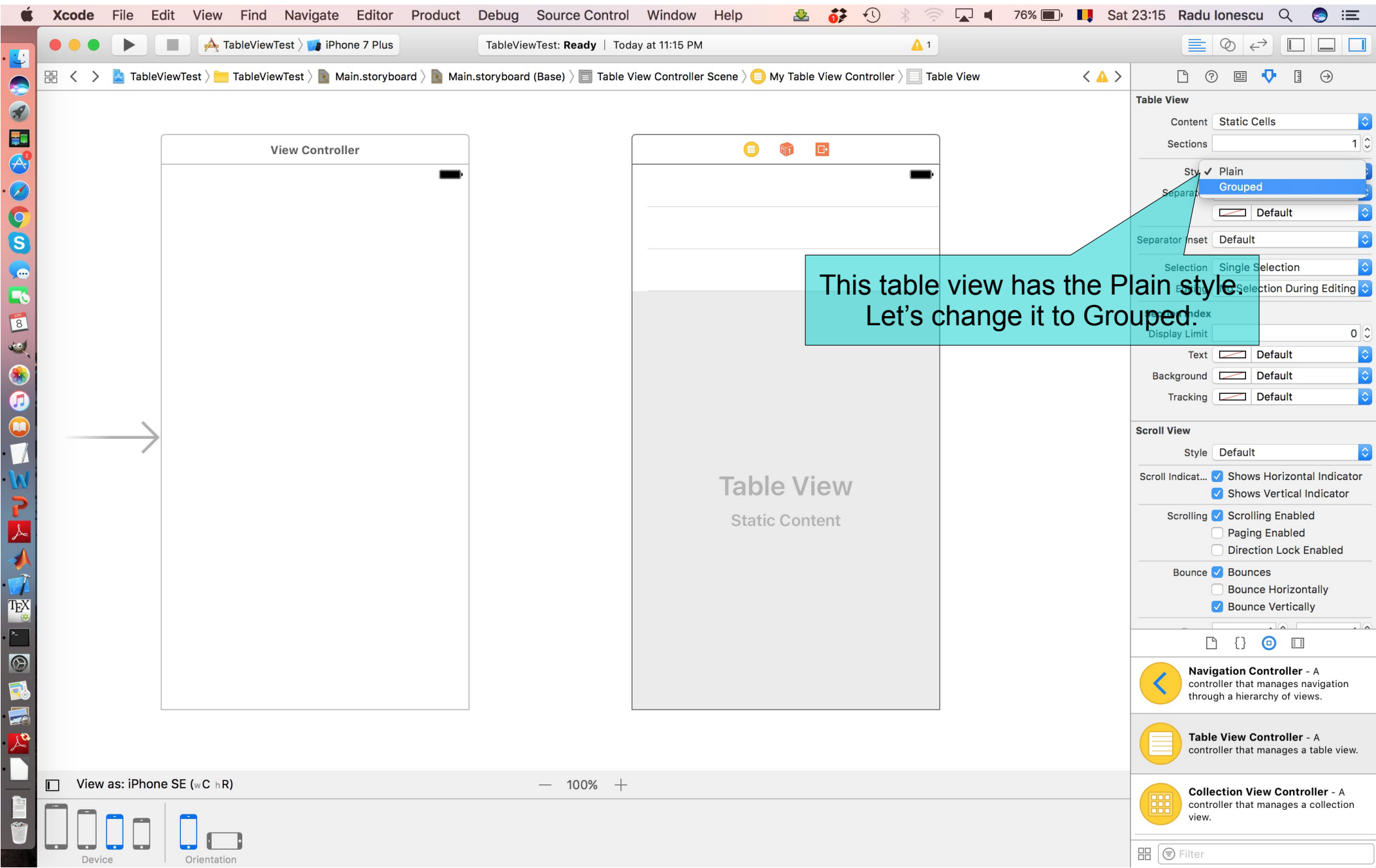
Creating Table View MVCs

You can customize both the look of the table view and its cells from Interface Builder.



Creating Table View MVCs

You can customize both the look of the table view and its cells from Interface Builder.



Creating Table View MVCs

And you can change the look of each cell as well.

The screenshot shows the Xcode IDE with a storyboard open. The storyboard contains a 'View Controller' and a 'Table View' with 'Static Content'. A blue callout box points to a cell in the table view with the text: 'Click on a cell that you want to change and set its attributes in the Inspector.' The right-hand side of the interface shows the 'Table View Cell' inspector, which includes settings for Style (Custom), Identifier (Reuse Identifier), Selection (Default), Accessory (None), Editing Acc. (None), Focus Style (Default), Indentation (Level 0, Width 10), Indent While Editing (checked), Shows Re-order Controls (unchecked), Separator (Default Insets), Content Mode (Scale To Fill), Semantic (Unspecified), Tag (0), Interaction (User Interaction Enabled checked, Multiple Touch unchecked), Alpha (1), Background (white), Tint (blue), Drawing (Opaque checked, Hidden unchecked, Clears Graphics Context checked, Clip To Bounds checked). Below the inspector, there are links to 'Navigation Controller', 'Table View Controller', and 'Collection View Controller'. The bottom of the screen shows the 'View as: iPhone SE (w C h R)' and 'Device' and 'Orientation' settings.

Click on a cell that you want to change and set its attributes in the Inspector.

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0 Level, 10 Width

☒ Indent While Editing

☐ Shows Re-order Controls

Separator: Default Insets

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

Interaction: ☒ User Interaction Enabled, ☐ Multiple Touch

Alpha: 1

Background: [Color Picker]

Tint: [Color Picker]

Drawing: ☒ Opaque, ☐ Hidden, ☒ Clears Graphics Context, ☒ Clip To Bounds

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

View as: iPhone SE (w C h R)

100%

Device Orientation

Creating Table View MVCs

And you can change the look of each cell as well.

Change Style to Basic, Right Detail, Left Detail, Subtitle and notice the cell layout each time.

The screenshot displays the Xcode development environment. On the left, a storyboard is visible with a 'View Controller' and a 'Table View' containing 'Static Content'. A large arrow points from the View Controller to the Table View. The right-hand side shows the 'Table View Cell' configuration panel. The 'Style' is set to 'Subtitle', and the 'Image' is set to 'Image'. The 'Identifier' is 'Reuse Identifier'. The 'Selection' is 'Default', 'Accessory' is 'None', 'Editing Acc.' is 'None', and 'Focus Style' is 'Default'. The 'Indentation' is set to 0 Level and 10 Width. The 'Indent While Editing' checkbox is checked, and 'Shows Re-order Controls' is unchecked. The 'Separator' is 'Default Insets'. The 'View' section shows 'Content Mode' as 'Scale To Fill', 'Semantic' as 'Unspecified', and 'Tag' as 0. The 'Interaction' section has 'User Interaction Enabled' checked and 'Multiple Touch' unchecked. The 'Alpha' is 1, and the 'Background' is 'Default'. The 'Drawing' section has 'Opaque' checked, 'Hidden' unchecked, and 'Clears Graphics Context' checked. The bottom of the screen shows the 'View as: iPhone SE (w C h R)' and a zoom level of 100%.

Table View Cell

- Style: Subtitle
- Image: Image
- Identifier: Reuse Identifier
- Selection: Default
- Accessory: None
- Editing Acc.: None
- Focus Style: Default
- Indentation: 0 Level, 10 Width
- ☒ Indent While Editing
- ☐ Shows Re-order Controls
- Separator: Default Insets

View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: ☒ User Interaction Enabled, ☐ Multiple Touch
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: ☒ Opaque, ☐ Hidden, ☒ Clears Graphics Context

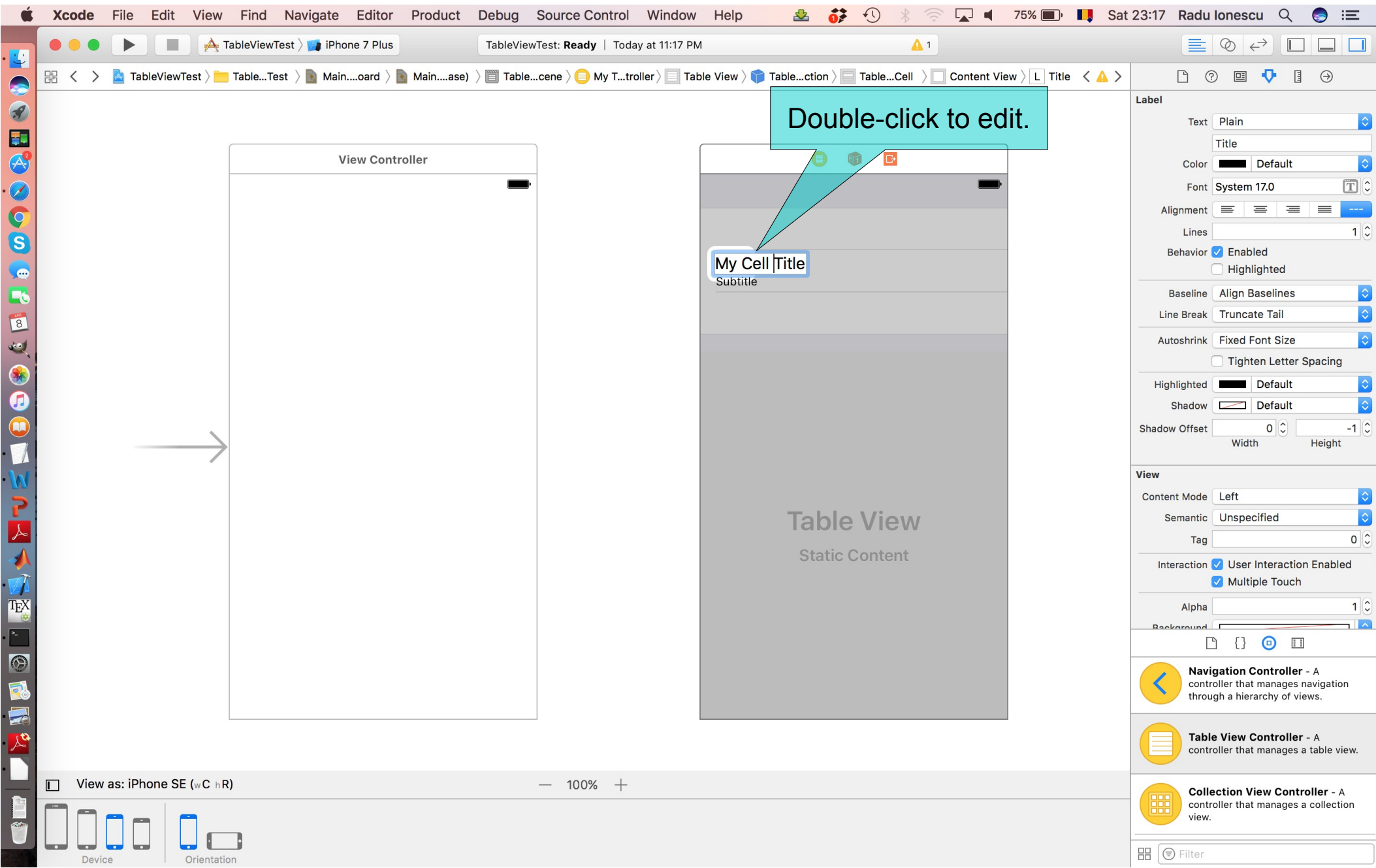
Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

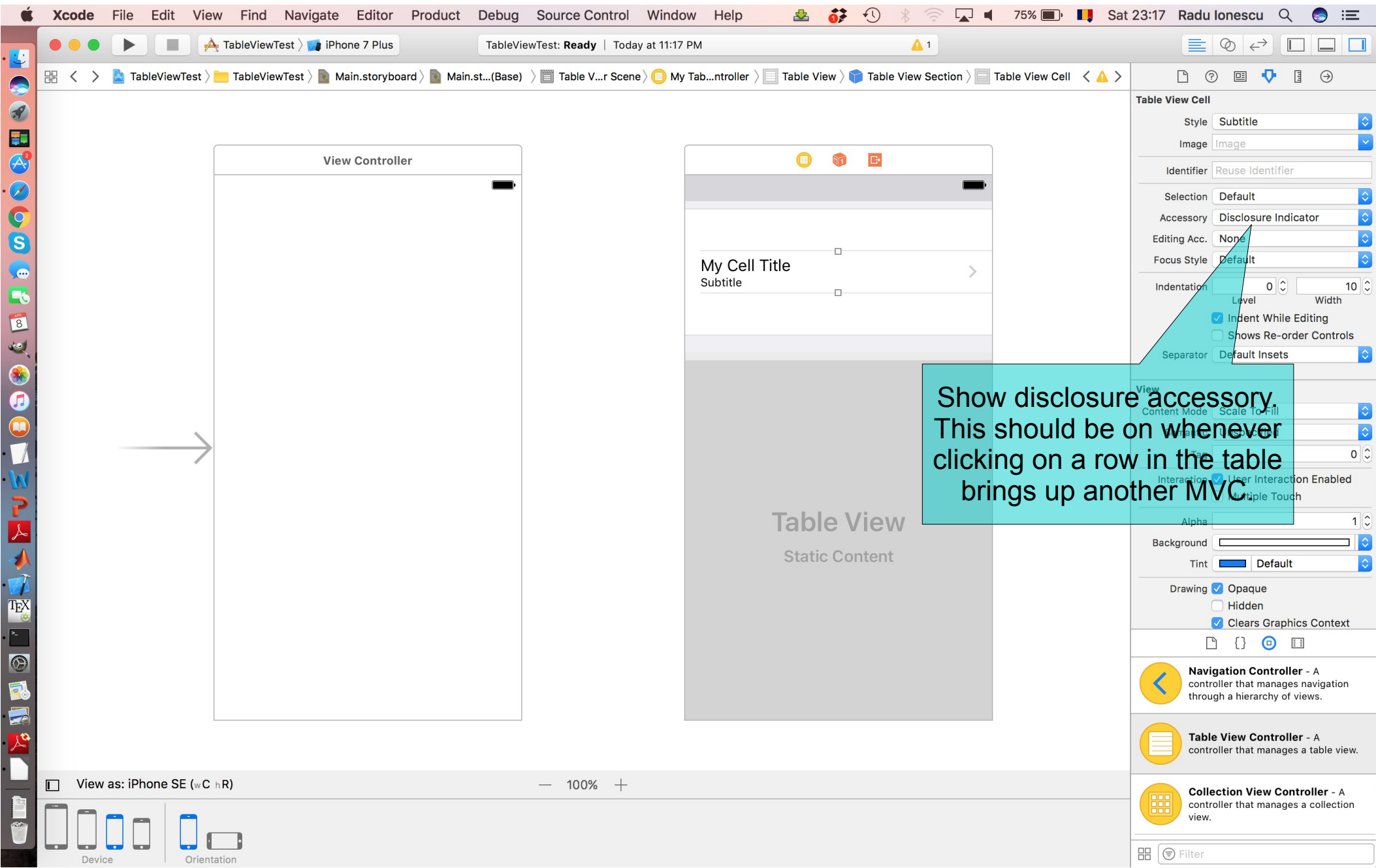
Creating Table View MVCs

And you can change the look of each cell as well.



Creating Table View MVCs

And you can change the look of each cell as well.



Creating Table View MVCs

And you can change the look of each cell as well.

The screenshot shows the Xcode interface with a storyboard open. On the left, a 'View Controller' is visible. In the center, a 'Table View' is shown with a 'Static Content' section containing a single cell. The cell has a title 'My Cell Title' and a subtitle 'Subtitle'. A callout box points to the 'Checkmark' accessory in the right-hand inspector, indicating how to show a checkmark for multiple selection.

Table View Cell Inspector Settings:

- Style: Subtitle
- Image: Image
- Identifier: Reuse Identifier
- Selection: Default
- Accessory: Checkmark
- Editing Acc.: None
- Focus Style: Default
- Indentation: 0 (Level), 10 (Width)
- ☒ Indent While Editing
- ☐ Shows Re-order Controls
- Separator: Default Insets
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: ☒ Opaque, ☐ Hidden, ☒ Clears Graphics Context

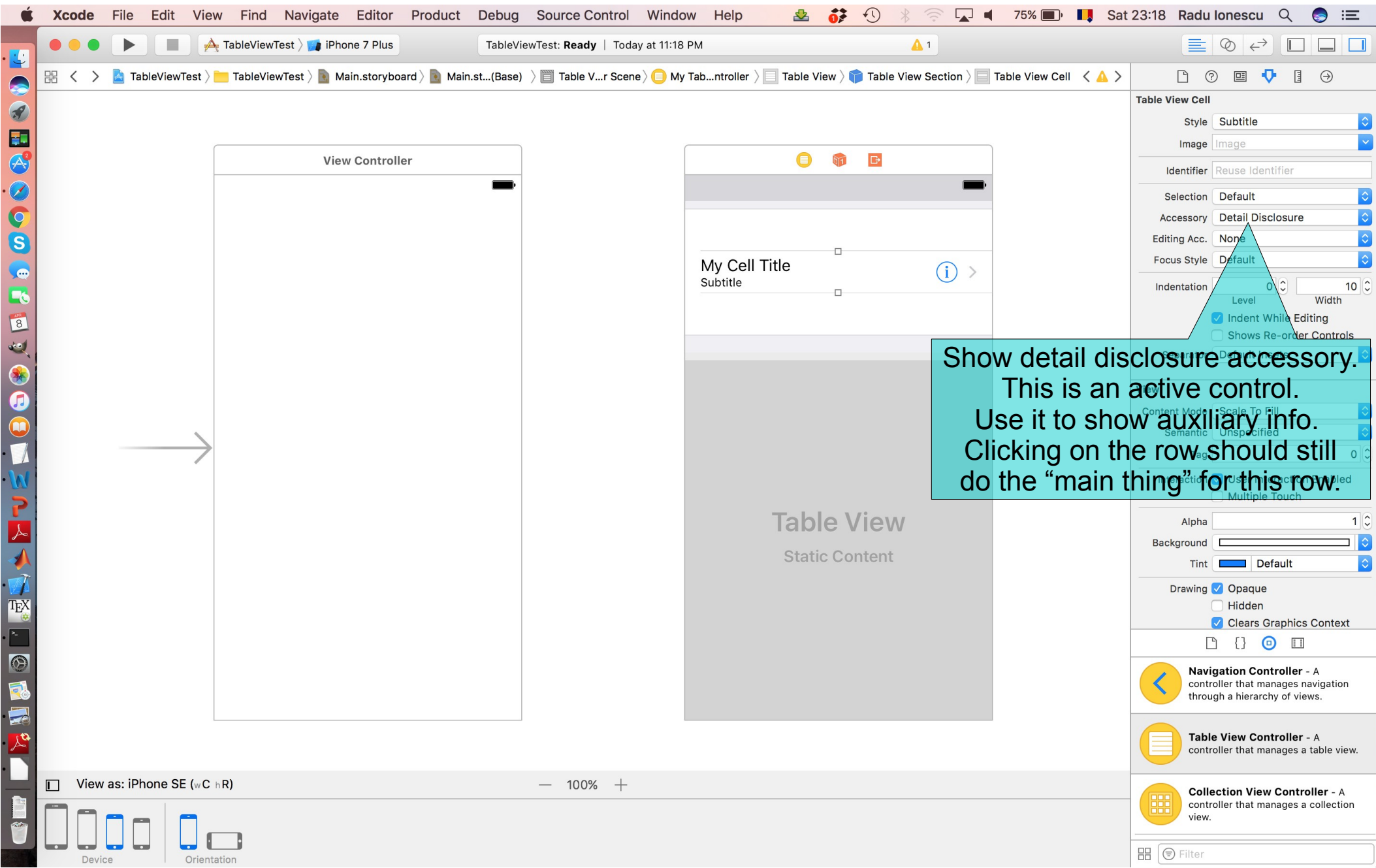
Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Creating Table View MVCs

And you can change the look of each cell as well.



Creating Table View MVCs

User taps on the blue detail disclosure below?

The screenshot shows the Xcode IDE with a storyboard open. The storyboard contains a 'View Controller' and a 'Table View'. The 'Table View' has a 'Static Content' section with a cell containing 'My Cell Title' and 'Subtitle'. A blue detail disclosure button (an 'i' icon) is located to the right of the subtitle. A blue triangle highlights this button. A text box at the bottom of the screen contains the following text:

This will be sent to your UITableViewController :

```
func tableView(_ tableView: UITableView,
               accessoryButtonTappedForRowWith indexPath: IndexPath)
```

The right sidebar shows the 'Table View Cell' settings, including 'Style' (Subtitle), 'Image' (Image), 'Identifier' (Reuse Identifier), 'Selection' (Default), 'Accessory' (Detail Disclosure), 'Editing Acc.' (None), 'Focus Style' (Default), 'Indentation' (0 Level, 10 Width), 'Indent While Editing' (checked), 'Shows Re-order Controls' (unchecked), and 'Separator' (Default Insets). The 'View' settings show 'Content Mode' (Scale To Fill), 'Semantic' (Unspecified), 'Tag' (0), 'Interaction' (User Interaction Enabled), 'Alpha' (1), 'Background' (Default), and 'Tint' (Default). The 'Drawing' settings show 'Opaque' (checked), 'Hidden' (unchecked), and 'Clears Graphics Context' (checked). The bottom of the screen shows the 'View as: iPhone SE (w C h R)' and '100%' zoom level. The bottom right corner shows a list of view controllers: 'Table View Controller - A controller that manages a table view.' and 'Collection View Controller - A controller that manages a collection view.'

Creating Table View MVCs

Notice that some cell styles can have an image.

You can set this in the code as well (more in a moment on this).

The screenshot shows the Xcode IDE with a storyboard open. On the left, a 'View Controller' is shown. In the center, a 'Table View' is displayed with a single cell containing 'My Cell Title' and 'Subtitle'. A blue information icon (i) is next to the cell. On the right, the 'Table View Cell' properties panel is open. The 'Image' property is highlighted, and a dropdown menu is showing 'Apple' as a selected option. A blue callout box with the text 'Let's set an image.' points to the 'Image' property. The bottom of the screen shows the 'View as: iPhone SE (w C h R)' and 'Device' and 'Orientation' settings.

Table View Cell

- Style: Subtitle
- Image: Apple
- Identifier: reuseIdentifier
- Selection: Default
- Accessory: Detail Disclosure
- Editing Acc.: None
- Focus Style: Default
- Indentation: 0 10
- Level: Width
- ☒ Indent While Editing
- ☐ Shows Re-order Controls
- Separator: Default Insets

View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: ☒ User Interaction Enabled
- ☐ Multiple Touch
- Alpha: 1
- Background: Default
- Tint: Default
- Drawing: ☒ Opaque
- ☐ Hidden
- ☒ Clears Graphics Context

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Creating Table View MVCs

Notice that some cell styles can have an image.

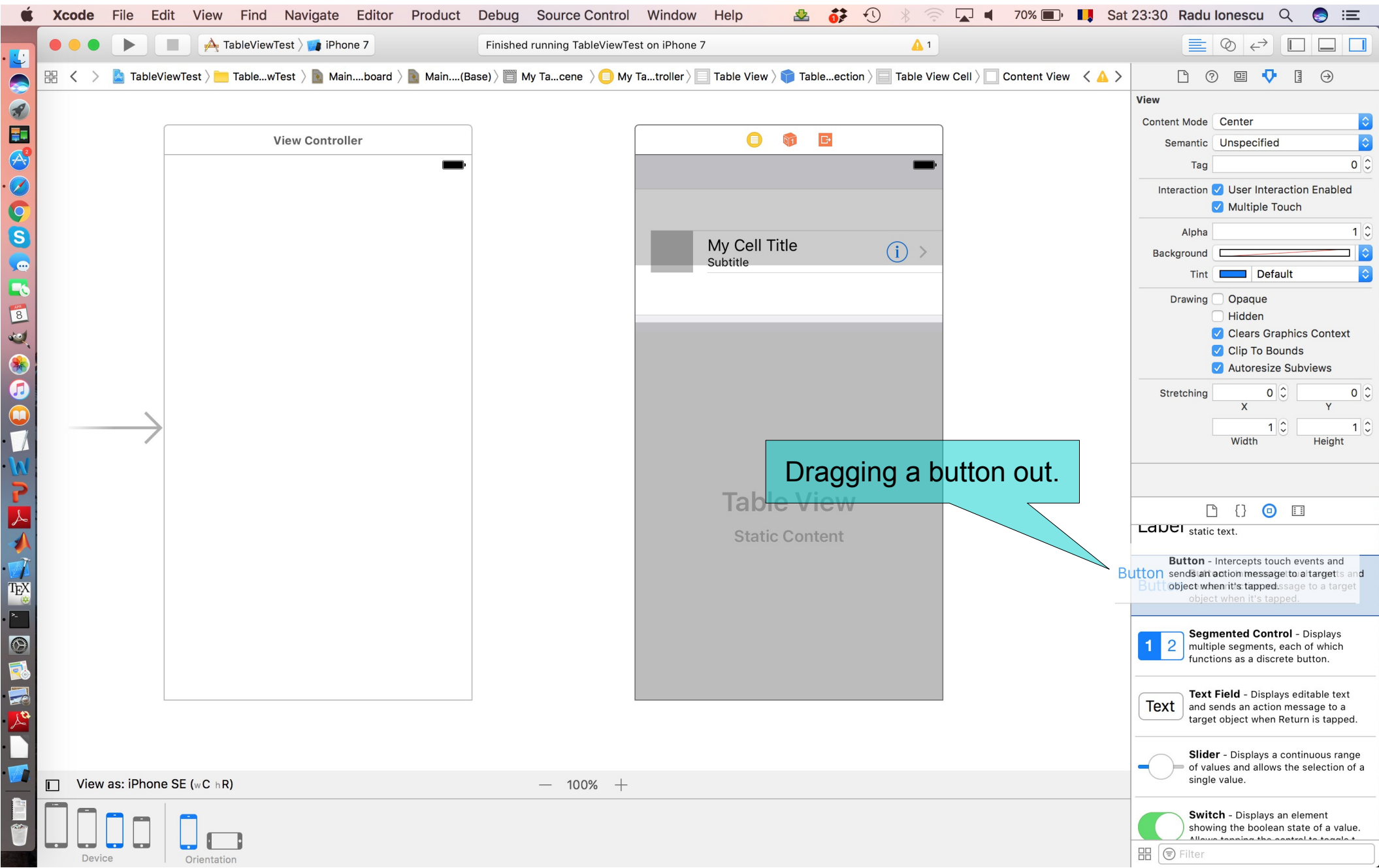
You can set this in the code as well (more in a moment on this).

The screenshot displays the Xcode development environment. The main storyboard area shows a 'View Controller' on the left and a 'Table View' on the right. The 'Table View' is labeled 'Static Content'. A large arrow points from the 'View Controller' to the 'Table View'. The right-hand side of the interface shows the 'Attributes Inspector' panel, which is currently set to 'Table View Cell'. The 'Style' is 'Subtitle', and the 'Image' is 'Apple'. The 'Identifier' is 'Reuse Identifier'. The 'Selection' is 'Default', 'Accessory' is 'Detail Disclosure', 'Editing Acc.' is 'None', and 'Focus Style' is 'Default'. The 'Indentation' is set to 0 Level and 10 Width. The 'Indent While Editing' checkbox is checked. The 'Shows Re-order Controls' checkbox is unchecked. The 'Separator' is 'Default Insets'. The 'View' section shows 'Content Mode' as 'Scale To Fill', 'Semantic' as 'Unspecified', and 'Tag' as 0. The 'Interaction' section shows 'User Interaction Enabled' checked and 'Multiple Touch' unchecked. The 'Alpha' is 1, and the 'Background' is 'Default'. The 'Drawing' section shows 'Opaque' checked, 'Hidden' unchecked, and 'Clears Graphics Context' checked. The bottom of the interface shows the 'View as: iPhone SE (w C h R)' and a zoom level of 100%. The bottom-left corner shows the 'Device' and 'Orientation' settings.

Xcode interface showing a storyboard with a View Controller and a Table View. The Table View is labeled "Table View Static Content". The right-hand side shows the Attributes Inspector for a Table View Cell, with settings for Style (Subtitle), Image (Apple), Identifier (Reuse Identifier), Selection (Default), Accessory (Detail Disclosure), Editing Acc. (None), Focus Style (Default), Indentation (0 Level, 10 Width), Indent While Editing (checked), Shows Re-order Controls (unchecked), Separator (Default Insets), Content Mode (Scale To Fill), Semantic (Unspecified), Tag (0), Interaction (User Interaction Enabled checked, Multiple Touch unchecked), Alpha (1), Background (Default), Drawing (Opaque checked, Hidden unchecked, Clears Graphics Context checked).

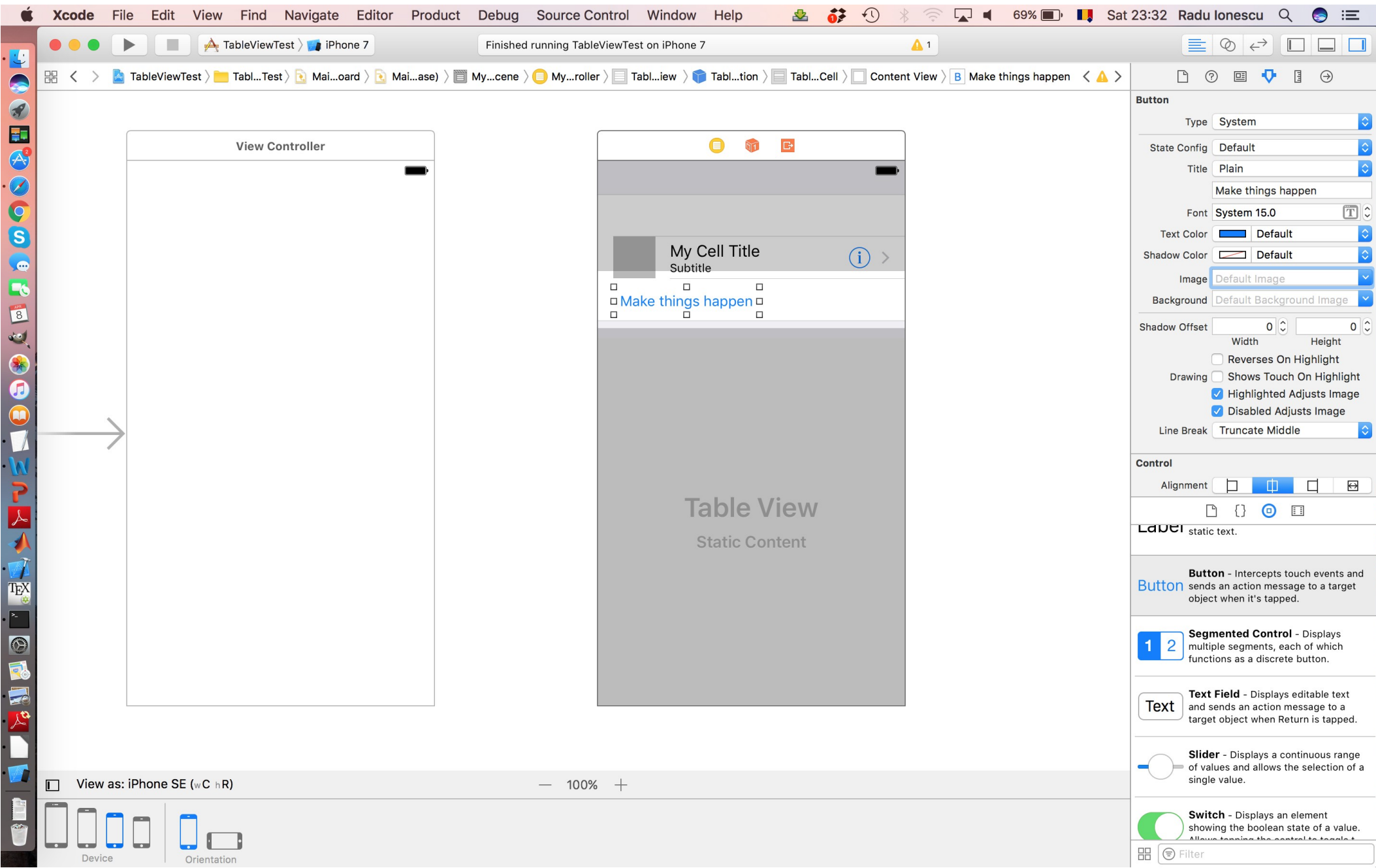
Creating Table View MVCs

In the Custom style, you can drag out views and wire them up as outlets!



Creating Table View MVCs

In the Custom style, you can drag out views and wire them up as outlets!



Creating Table View MVCs

In the Custom style, you can drag out views and wire them up as outlets!

CTRL + drag to associate an action to this button.

```
10
11 class MyTableViewController: UITableViewController
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15
16         // Uncomment the following line to pres
17         // self.clearsSelectionOnViewWillAppear
18
19         // Uncomment the following line to disp
20         // self.navigationItem.rightBarButtonItem
21     }
22
23     override func didReceiveMemoryWarning() {
24         super.didReceiveMemoryWarning()
25         // Dispose of any resources that can be
26     }
27
28     // MARK: - Target Actions
29
30     // MARK: - Table view data source
31
32     /*
33     override func numberOfSections(in tableView: UITableView)
34         // #warning Incomplete implementation,
35         return 0
36     }
37
38     override func tableView(_ tableView: UITableView)
39         // #warning Incomplete implementation,
40         return 0
41     }
42
43     *
44     override func tableView(_ tableView: UITableView)
45         let cell = tableView.dequeueReusableCell
46
47         // Configure the cell...
```

Button

Type: System

State Config: Default

Title: Plain

Make things happen

Font: System 15.0

Text Color: Default

Shadow Color: Default

Image: Default Image

Background: Default Background Image

Shadow Offset: 0 0

Width: 0 Height: 0

☐ Reverses On Highlight

☐ Shows Touch On Highlight

☒ Highlighted Adjusts Image

☒ Disabled Adjusts Image

Line Break: Truncate Middle

Control

Alignment: [Icons]

Label: static text.

Button - Intercepts touch events and sends an action message to a target object when it's tapped.

1 2 Segmented Control - Displays multiple segments, each of which functions as a discrete button.

Text Text Field - Displays editable text and sends an action message to a target object when Return is tapped.

Slider - Displays a continuous range of values and allows the selection of a single value.

Switch - Displays an element showing the boolean state of a value. Allow tapping the control to toggle.

Filter

Creating Table View MVCs

In the Custom style, you can drag out views and wire them up as outlets!

The screenshot displays the Xcode IDE with the following components:

- Storyboard:** A table view cell is shown with a title "My Cell Title", a subtitle "Subtitle", and a button labeled "Make things happen".
- Swift File (MyTableViewController.swift):**

```
11 class MyTableViewController: UITableViewController
12
13 override func viewDidLoad() {
14     super.viewDidLoad()
15
16     // Uncomment the following line to pres
17     // self.clearsSelectionOnViewWillAppear
18
19     // Uncomment the following line to disp
20     // self.navigationItem.rightBarButtonItemIt
21 }
22
23 override func didReceiveMemoryWarning() {
24     super.didReceiveMemoryWarning()
25     // Dispose of any resources that can be
26 }
27
28 // MARK: - Target Actions
29
30 // MARK: - Table view data source
31
32 /*
33 34 override func numberOfSections(in tableView: UITableView)
35     // #warning Incomplete implementation,
36     return 0
37 }
38
39 override func tableView(_ tableView: UITableView)
40     // #warning Incomplete implementation,
41     return 0
42 }
43
44 *
45 override func tableView(_ tableView: UITableView)
46     let cell = tableView.dequeueReusableCell
47
48 // Configure the cell...
```
- Connection Dialog:** A dialog box is open for connecting an action. The "Object" is "My Table View Contr...", the "Name" is "makeThingsHappenNow", the "Type" is "Any", the "Event" is "Touch Up Inside", and the "Arguments" are "None".
- Right Panel:** A list of UI elements with their descriptions:
 - Button:** Intercepts touch events and sends an action message to a target object when it's tapped.
 - Segmented Control:** Displays multiple segments, each of which functions as a discrete button.
 - Text Field:** Displays editable text and sends an action message to a target object when Return is tapped.
 - Slider:** Displays a continuous range of values and allows the selection of a single value.
 - Switch:** Displays an element showing the boolean state of a value.

Creating Table View MVCs

In the Custom style, you can drag out views and wire them up as outlets!

The screenshot displays the Xcode IDE with a project named 'TableViewTest' running on an iPhone 7 simulator. The interface on the left shows a table with a title 'My Cell Title', a subtitle 'Subtitle', and a single row with a cell containing 'Make things happen'. The code editor in the center shows the implementation of MyTableViewController, including viewDidLoad, didReceiveMemoryWarning, and a makeThingsHappenNow() action. The right sidebar shows the Identity and Type panel for MyTableViewController.swift.

```
class MyTableViewController: UITableViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        // Uncomment the following line to pres
        // self.clearsSelectionOnViewWillAppear

        // Uncomment the following line to disp
        // self.navigationItem.rightBarButtonItemIt
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be
    }

    // MARK: - Target Actions
    @IBAction func makeThingsHappenNow() {
        print("We make things happen")
    }

    // MARK: - Table view data source

    /*
    override func numberOfSections(in tableView: UITableView) {
        // #warning Incomplete implementation,
        return 0
    }

    override func tableView(_ tableView: UITableView) {
        // #warning Incomplete implementation,
        return 0
    }
    */
}
```

Identity and Type

Name: MyTableViewController.swift

Type: Default - Swift Source

Location: Relative to Group

Full Path: /Users/raduionescu/FMI/Teaching/iOSLab/Apps/Swift/TableViewTest/TableViewTest/MyTableViewController.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

TableViewTest

Text Settings

Text Encoding: Unicode (UTF-8)

Line Endings: Default - macOS / Unix (LF)

Label: static text.

Button: Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control: Displays multiple segments, each of which functions as a discrete button.

Text Field: Displays editable text and sends an action message to a target object when Return is tapped.

Slider: Displays a continuous range of values and allows the selection of a single value.

Switch: Displays an element showing the boolean state of a value.

Creating Table View MVCs

Move the entry point arrow to the table view controller, and let's run the application.

Tap the detail disclosure accessory and notice the log printed on the console.

```
class MyTableViewController: UITableViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // Uncomment the following line to preserve selected state during
        // transitions. This will only work if the visualizer supports
        // preserving selected state, available from iOS 13.0+.
        // if #available(iOS 13.0, *) {
        //     self.tableView.allowsSelection = false
        // }

        // Uncomment the following line to display a rightBarButtonItem in the
        // right sidebar
        // self.navigationItem.rightBarButtonItem = self.editButtonItem

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    // MARK: - Target Actions

    @IBAction func makeThingsHappenNow() {
        print("We make things happen")
    }

    // MARK: - Table view data source

    /*
     * override func numberOfSections(in tableView: UITableView) -> Int {
     *     // #warning Incomplete implementation, return 0
     *     return 0
     * }
     */
}
```

We make things happen
We make things happen

Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.

The screenshot shows the Xcode IDE with a storyboard open. The storyboard contains a 'View Controller' and a 'Table View'. The 'Table View' is currently in 'Static Cells' mode. A callout box points to the 'Table View' with the text: 'Switch to a Dynamic Table with Prototype Cells.' The 'Table View' properties panel on the right shows the 'Dynamic Prototypes' option selected. The 'Table View' is currently in 'Static Cells' mode. The 'Table View' properties panel on the right shows the 'Dynamic Prototypes' option selected. The 'Table View' is currently in 'Static Cells' mode. The 'Table View' properties panel on the right shows the 'Dynamic Prototypes' option selected.

Switch to a Dynamic Table with Prototype Cells.

Table View

- Dynamic Prototypes
- Static Cells

Style: Plain

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Editing

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

Scroll Indicators: Shows Horizontal Indicator, Shows Vertical Indicator

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controllers.

Page View Controller - Presents a series of views that are displayed one at a time.

View as: iPhone SE (w C h R)

100%

Device Orientation

Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.

The screenshot shows the Xcode IDE with a storyboard open. The storyboard contains a 'View Controller' and a 'Table View'. The 'Table View' is labeled 'Table View' and 'Prototype Content'. A callout box points to the 'Table View' with the text: 'Now click on the Prototype to edit it. All cells in this table will be like this Prototype (though we'll set the contents to be different in code).' The right-hand side of the interface shows the 'Table View Cell' inspector with various settings like Style, Identifier, Selection, Accessory, Editing Acc., Focus Style, Indentation, Separator, Content Mode, Semantic, Tag, Interaction, and Multiple Touch. The bottom of the interface shows the 'View as: iPhone SE (w C h R)' and a zoom level of 100%.

Now click on the Prototype to edit it.
All cells in this table will be like this Prototype (though we'll set the contents to be different in code).

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0 Level 10 Width

☒ Indent While Editing

☐ Shows Re-order Controls

Separator: Default Insets

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

Interaction: ☒ User Interaction Enabled

☐ Multiple Touch

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controle...

Page View Controller - Presents a

View as: iPhone SE (w C h R)

100%

Device Orientation

Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.

The screenshot shows the Xcode IDE with a storyboard open. On the left, a 'View Controller' is connected to a 'Table View'. The 'Table View' is currently in 'Prototype Cells' mode, showing a single cell with the text 'Table View Prototype Content'. A blue callout box points to the 'Table View' with the text: 'Now click on the Prototype to edit it. All cells in this table will be like this Prototype (though we'll set the contents to be different in code).' Another blue callout box points to the 'Table View Cell' in the Attributes Inspector on the right, with the text: 'You should see Table View Cell properties appear in the Attributes Inspector.' The Attributes Inspector shows various settings for the 'Table View Cell', including Style (Custom), Identifier (Reuse Identifier), Selection (Default), Accessory (None), Editing Acc. (None), Focus Style (Default), Indentation (0 Level, 10 Width), Indent While Editing (checked), Shows Re-order Controls (unchecked), Separator (Default Insets), Content Mode (Scale To Fill), Semantic (Unspecified), Tag (0), Interaction (User Interaction Enabled checked, Multiple Touch unchecked), and a list of other view controllers at the bottom.

View Controller

Prototype Cells

Table View
Prototype Content

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0 Level, 10 Width

☒ Indent While Editing

☐ Shows Re-order Controls

Separator: Default Insets

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

☒ User Interaction Enabled

☐ Multiple Touch

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controle...

Page View Controller - Presents a

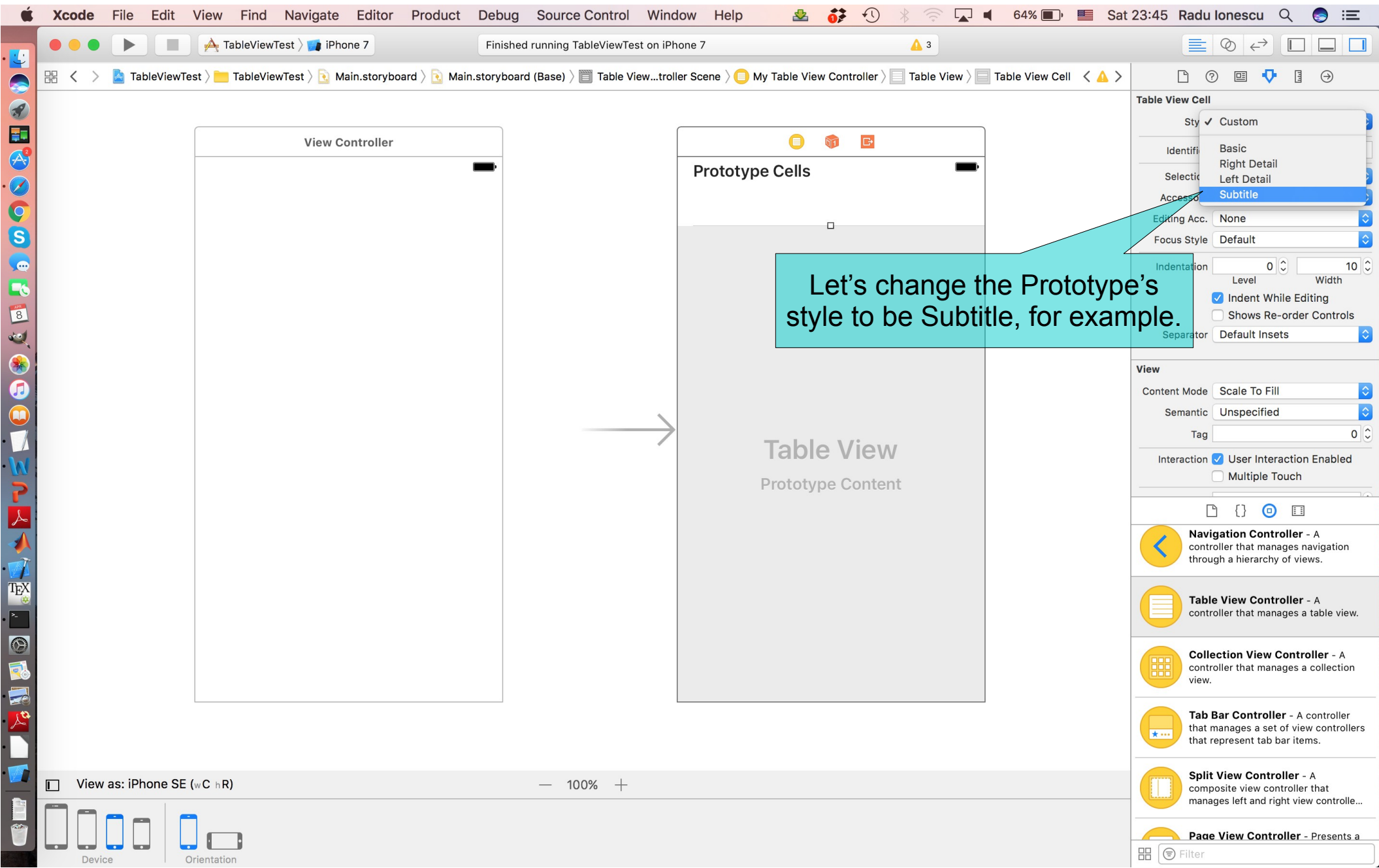
View as: iPhone SE (w C h R)

100%

Device Orientation

Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.



Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.

The screenshot shows the Xcode IDE with a storyboard open. The storyboard contains a 'View Controller' and a 'Table View'. The 'Table View' is currently in 'Prototype Cells' mode, showing a single cell with 'Title' and 'Subtitle' labels. A blue callout box points to the 'Reuse Identifier' field in the 'Table View Cell' properties panel, which is set to 'Reuse Identifier'. The callout text reads: 'The reuse identifier is a very important field! It is the name that we will reference in our code to identify this prototype (more on this in a moment)'. The properties panel on the right shows various settings for the 'Table View Cell', including Style (Subtitle), Image, Identifier (Reuse Identifier), Selection (Default), Accessory (None), Editing Acc. (None), Focus Style (Default), Indentation (0), Content Mode (Scale To Fill), Semantic (Unspecified), Tag (0), and Interaction (User Interaction Enabled). The bottom of the screen shows the 'View as: iPhone SE (w C h R)' and a zoom level of 100%.

View Controller

Prototype Cells

Title

Subtitle

Table View

Prototype Content

The reuse identifier is a very important field!
It is the name that we will reference in
our code to identify this prototype
(more on this in a moment).

Table View Cell

Style Subtitle

Image Image

Identifier Reuse Identifier

Selection Default

Accessory None

Editing Acc. None

Focus Style Default

Indentation 0

Content Mode Scale To Fill

Semantic Unspecified

Tag 0

Interaction ☒ User Interaction Enabled

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controle...

Page View Controller - Presents a

View as: iPhone SE (w C h R)

100%

Device Orientation

Creating Table View MVCs

All of the above examples were “static” cells (setup in the storyboard). If you switch to dynamic mode, then the cell you edit is a “prototype” for all cells in the list.

The screenshot shows the Xcode IDE with a storyboard open. On the left, a 'View Controller' is visible. In the center, a 'Table View' is shown with a 'Prototype Content' area. An arrow points from the 'View Controller' to the 'Table View'. On the right, a 'Table View Cell' is selected, and its properties are displayed in the right-hand pane. The 'Table View Cell' properties include: Style: Subtitle, Image: Image, Identifier: My Table View Cell, Selection: Default, Accessory: None, Editing Acc.: None, Focus Style: Default, Indentation: 0, Level: 0, Width: 10, Separator: Default, and Tag: 0. A text box is overlaid on the 'Table View Cell' properties, stating: 'Pick a name that is meaningful. “My Table View Cell” would probably not be that great. Something like “Photo Description” (if this were a list of photos) would be better.'

View Controller

Prototype Cells

Title

Subtitle

Table View

Prototype Content

Pick a name that is meaningful. “My Table View Cell” would probably not be that great. Something like “Photo Description” (if this were a list of photos) would be better.

Table View Cell

Style: Subtitle

Image: Image

Identifier: My Table View Cell

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0

Level: 0

Width: 10

Separator: Default

Tag: 0

Interaction: ☒ User Interaction Enabled

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controllers.

Page View Controller - Presents a

View as: iPhone SE (w C h R)

100%

Device

Orientation

UITableView Protocols

How do we connect to all this stuff in our code?

- A `UITableView` has two important properties: its `delegate` and its `dataSource`.
- The `delegate` is used to control how the table is displayed.
- The `dataSource` provides the data that is displayed inside the cells.
- Your `UITableViewController` is automatically set as the `UITableView`'s `delegate` and `dataSource`.
- Your `UITableViewController` subclass will also have a property that points to the `UITableView`:

```
var tableView: UITableView! { get set }
```


UITableView Protocols


- To be “dynamic”, we need to be the `UITableView`’s `dataSource`.
- Three important methods in this protocol:
 1. How many sections in the table?
 2. How many rows in each section?
 3. Give me a `UIView` to use to draw each cell at a given row in a given section.
- Let’s cover the last one first.

UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

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



In a static table, you do not need to implement this method (though you can if you want to ignore what's in the storyboard).

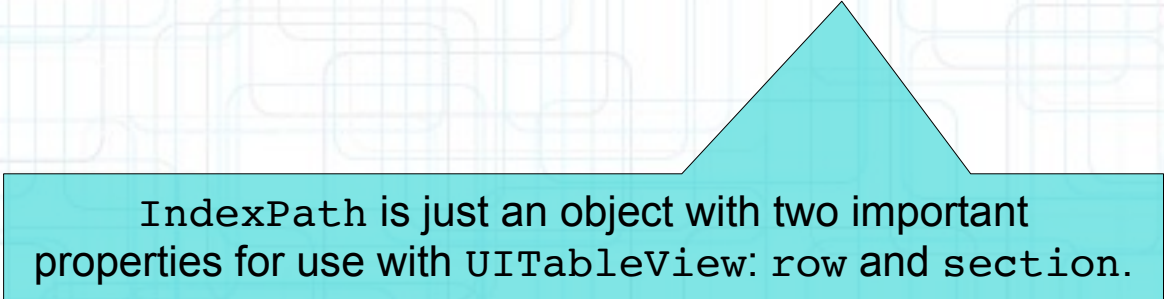
```
    return cell  
}
```

UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

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



IndexPath is just an object with two important properties for use with UITableView: row and section.

```
    return cell  
}
```


UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

```
override func tableView(_ tableView: UITableView,  
    cellForRowAt indexPath: IndexPath) -> UITableViewCell  
{  
    // get a cell to use (instance of UITableViewCell)  
  
    // set properties on the cell to prepare it to display  
  
    return cell  
}
```

UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

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

This MUST match what is in your storyboard if you want to use the prototype you defined there!

UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

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

The cells in the table are actually reused. When one goes off-screen, it gets put into a “reuse pool”. The next time a cell is needed, one is grabbed from the reuse pool if available. If none is available, one will be put into the reuse pool if there’s a prototype in the storyboard. Otherwise the dequeue method might return `nil` or generate an `Error`.

UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

```
override func tableView(_ tableView: UITableView,  
    cellForRowAt indexPath: IndexPath) -> UITableViewCell  
{  
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell",  
                                           for: indexPath)  
  
    // set properties on the cell to prepare it to display  
  
    return cell  
}
```


UITableViewDataSource

How do we control what is drawn in each cell in a dynamic table?

- Each row is drawn by its own instance of `UITableViewCell`.
- Here is the `UITableViewDataSource` method to get that cell for a given row in a given section.

```
override func tableView(_ tableView: UITableView,  
    cellForRowAt indexPath: IndexPath) -> UITableViewCell  
{  
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell",  
                                           for: indexPath)  
  
    cell.textLabel?.text = self.data(forRow: indexPath.row,  
                                     inSection: indexPath.section)  
  
    return cell  
}
```

There are obviously other things you can do in the cell besides setting its text (detail text, image, accessory, etc).

UITableViewDataSource

How does a dynamic table know how many rows are there?

- And how many sections, too, of course?

```
func numberOfSections(in tableView: UITableView) -> Int
```

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

- Number of sections is 1 by default. In other words, if you don't implement `numberOfSections(in:)`, it will be 1.
- No default for number of rows in a section.
- This is a required method in this protocol (as is `tableView(:cellForRowAt:)`).

What about a static table?

- Do not implement these `dataSource` methods for a static table.
- `UITableViewController` will take care of that for you.

UITableViewDataSource

There are a number of other methods in this protocol

- But we're not going to cover all of them.
- They are mostly about getting the headers and footers for sections.
- And about dealing with editing the table (moving/deleting/inserting rows).

UITableViewDataSource

There are a number of other methods in this protocol

- Let us continue with our demo and see, for example, how can we delete rows from a Table View.
- We implement the following method to return `true` to allow editing.

```
override func tableView(_ tableView: UITableView,  
    canEditRowAt indexPath: IndexPath) -> Bool  
{  
    /* Return false if you do not want the specified item  
     * to be editable. */  
    return true  
}
```


UITableViewDataSource

There are a number of other methods in this protocol

- Let us continue with our demo and see, for example, how can we delete rows from a Table View.
- We delete the row from the Table View and also from the Model by implementing this method:

```
override func tableView(_ tableView: UITableView,  
    commit editingStyle: UITableViewCellEditingStyle,  
    forRowAt indexPath: IndexPath)  
{  
    if editingStyle == .delete  
    {  
        // Delete the row from the data source  
        let deletedData = self.deleteData(atRow:indexPath.row,  
                                           inSection: indexPath.section)  
        print("We removed : \(deletedData)")  
  
        tableView.deleteRows(at: [indexPath], with: .fade)  
    }  
}
```

UITableViewDelegate

- All of the above was the `UITableView`'s `dataSource`.

But `UITableView` has another protocol-driven delegate called its `delegate`.

- The `delegate` controls how the `UITableView` is displayed.

Not what it displays (that's the `dataSource`'s job).

- It is common for `dataSource` and `delegate` to be the same object.

Usually the Controller of the MVC in which the `UITableView` is part of the (or is the entire) View.

- The `delegate` also lets you observe what the table view is doing.

Especially responding to when the user selects a row.

We often will use segues when this happens, but we can also track it directly.

Table View “Target/Action”

UITableViewDelegate method sent when row is selected

- This is sort of like table view “target/action”.
- You might use this to update a detail view in a split view if master is a table view

```
override func tableView(_ tableView: UITableView,
                        didSelectRowAt indexPath: IndexPath)
{
    /* Go do something based on information about our
     * data structure corresponding to indexPath.row
     * in indexPath.section */

    // We usually want to deselect the row once you are done
    tableView.deselectRow(at: indexPath, animated: true)
}
```

Table View “Target/Action”

Lots and lots of other delegate methods

- `will/did` methods for both selecting and deselecting rows.
- Providing `UIView` objects to draw section headers and footers.
- Handling editing rows (moving them around with touch gestures).
- `willBegin/didEnd` notifications for editing.
- Copying/pasting rows.

Table View Segues

You can segue when a row is touched, just like from a button. Segues will call `prepare(for:sender:)` with the chosen `UITableViewCell` as sender.

The screenshot shows the Xcode interface with a storyboard open. The storyboard contains two main elements: a 'Table View' on the left and a 'View Controller' on the right. The 'Table View' has a 'Prototype Cells' section with a single cell containing 'Title' and 'Subtitle'. A blue arrow points from this cell to the 'View Controller'. A large cyan arrow points from the 'Table View' to the 'View Controller'. A text box with a right-pointing arrow contains the text: 'CTRL+drag from your cell to another View Controller to create a segue in Storyboard.' The right sidebar shows the 'Table View Cell' properties, including 'Style: Subtitle', 'Image: Image', 'Identifier: CityCell', and 'Selection: Default'. The 'View' section shows 'Content Mode: Scale To Fill', 'Semantic: Unspecified', and 'Tag: 0'. The bottom status bar shows 'View as: iPhone SE (w C h R)' and '100%' zoom.

CTRL+drag from your cell to another View Controller to create a segue in Storyboard.

Table View Segues

You can segue when a row is touched, just like from a button. Segues will call `prepare(for:sender:)` with the chosen `UITableViewCell` as sender.

The screenshot shows the Xcode IDE with a storyboard open. On the left, a 'Table View' is shown with 'Prototype Cells'. A 'CityCell' is selected, and a segue menu is open, showing options like 'Push', 'Modal', and 'Custom'. A blue callout box points to the 'Push' option with the text: 'Chose your segue type. In a Navigation Controller, you might want to create a Push segue.'

Table View Cell

- Style: Subtitle
- Image: Image
- Identifier: CityCell
- Selection: Default
- Accessory: None
- Editing Acc.: None
- Focus Style: Default
- Indentation: 0 Level, 10 Width
- ☒ Indent While Editing
- ☐ Shows Re-order Controls
- Separator: Default Insets

View

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- Interaction: ☒ User Interaction Enabled

Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller that manages a set of view controllers that represent tab bar items.

Split View Controller - A composite view controller that manages left and right view controllers.

Page View Controller - Presents a

Table View Segues

You can segue when a row is touched, just like from a button. Segues will call `prepare(for:sender:)` with the chosen `UITableViewCell` as sender.

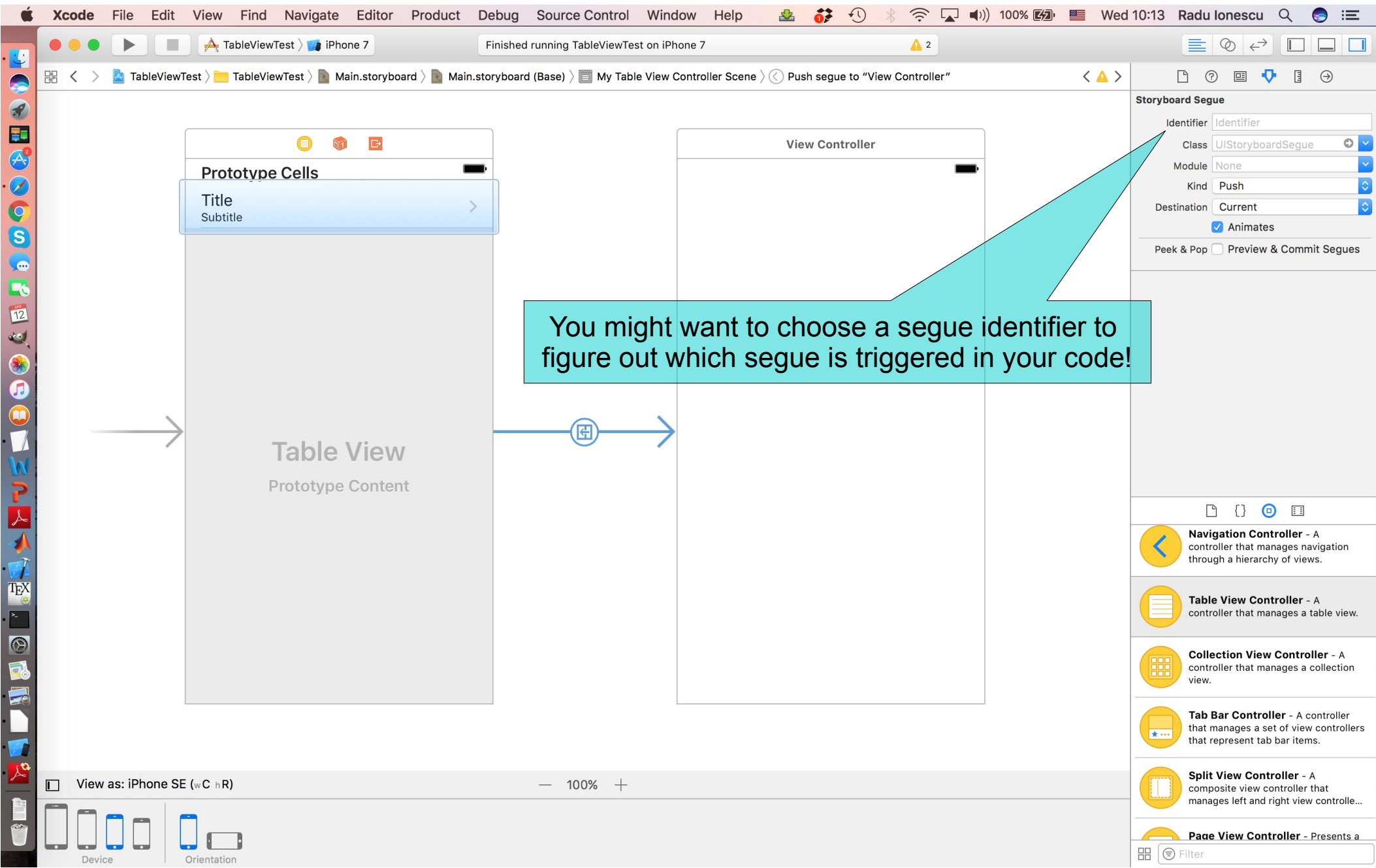


Table View Segues

You might also want to embed your Table View Controller in a Navigation Controller.

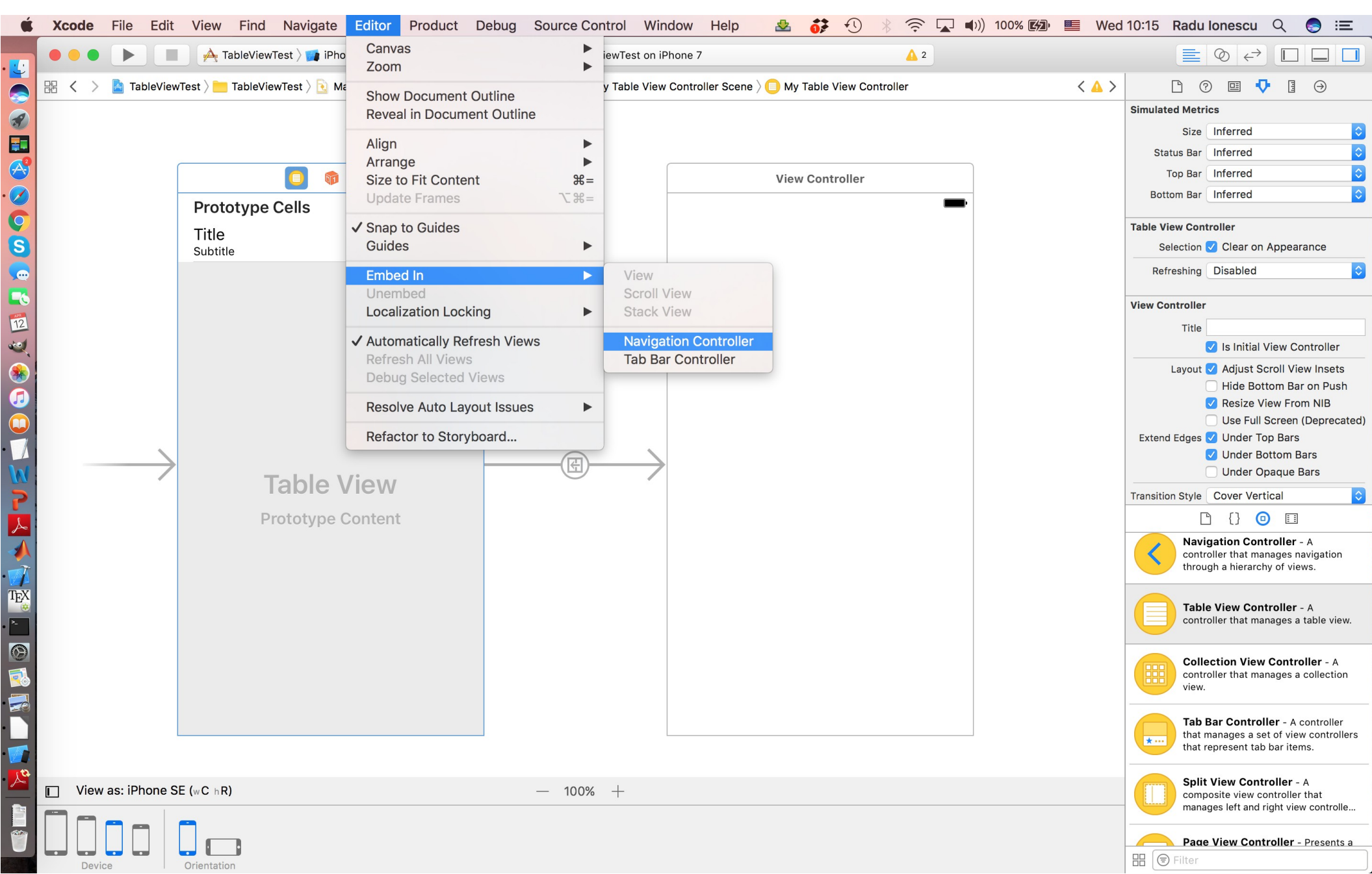


Table View Segues

You might also want to embed your Table View Controller in a Navigation Controller.

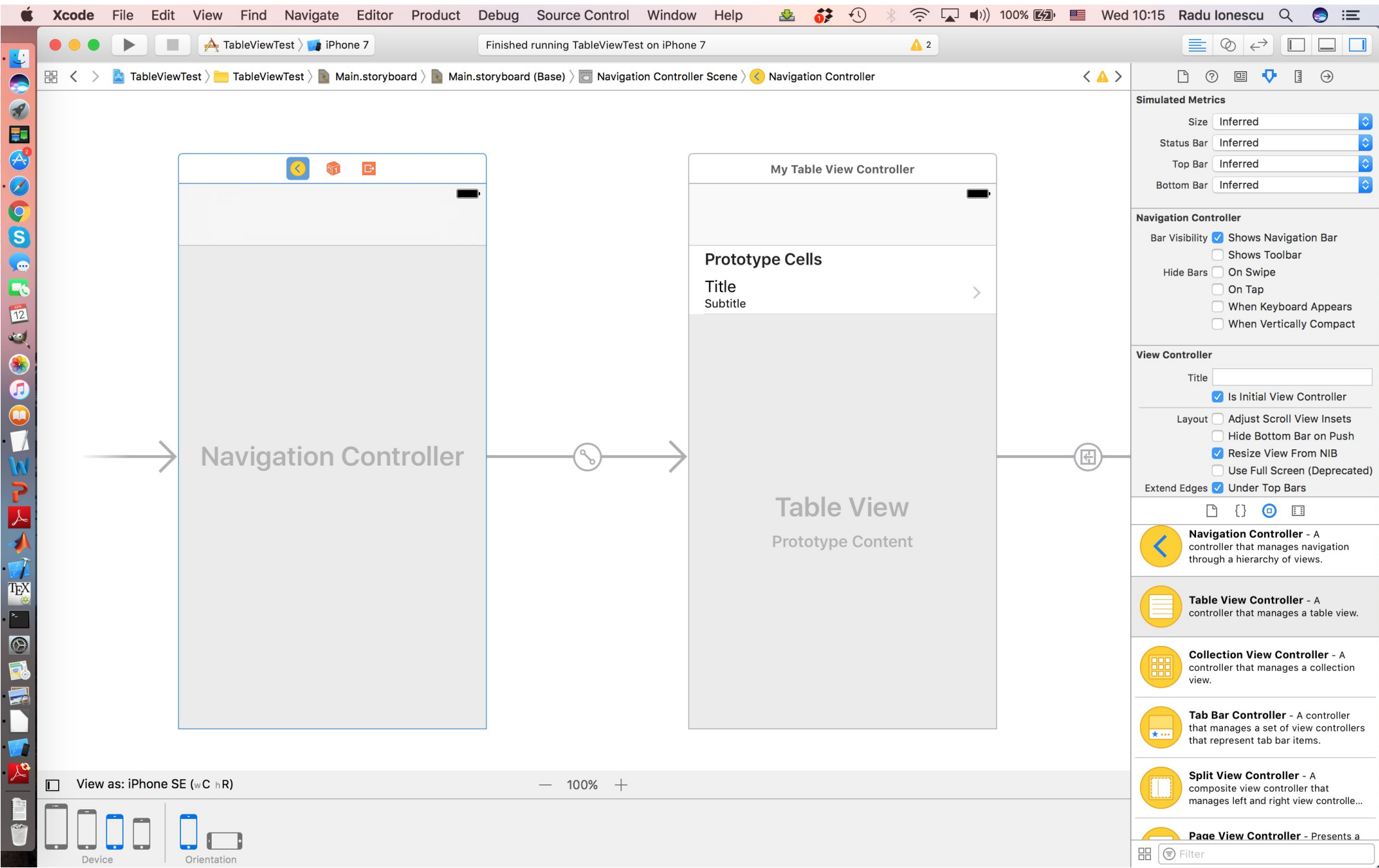


Table View Segues

- You can tailor whatever data the MVC needs to whichever cell was selected.
- This works whether dynamic or static.

```
override func prepare(for segue: UIStoryboardSegue,  
                      sender: Any?)  
{  
    if sender is UITableViewCell  
    {  
        let cell = sender as! UITableViewCell  
        let indexPath = self.tableView.indexPath(for: cell)  
  
        /* Prepare segue.destinationController to display  
         * based on information about my data structure  
         * corresponding to indexPath.row  
         * in indexPath.section */  
    }  
}
```


UITableView

What if your Model changes?

- You can:

```
func reloadData()
```

- Causes the table view to call `numberOfSections(in:)` and `tableView(numberOfRowsInSection:)` all over again and then `tableView(cellForRowAt:)` on each visible cell.
- Relatively heavy weight obviously, but if your entire data structure changes, that's what you need.
- If only part of your Model changes, there are lighter-weight reloaders, for example:

```
func reloadRows(at indexPaths: [IndexPath],  
                with animation: UITableViewRowAnimation)
```

UITableView

There are dozens of other methods in `UITableView`

- Setting headers and footers for the entire table.
- Controlling the look (separator style and color, default row height, etc).
- Getting cell information (cell for index path, index path for cell, visible cells, etc). Scrolling to a row.
- Selection management (allows multiple selection, getting the selected row, etc).
- Moving, inserting and deleting rows, etc.

Next Time

View Controller Lifecycle and UIKit:

- View Controller Lifecycle
- Image View
- Web View
- Scroll View