

# Stanford CS193p

Developing Applications for iOS  
Winter 2017



CS193p  
Winter 2017

# Today

## ⌚ Table View

Way to display large data sets

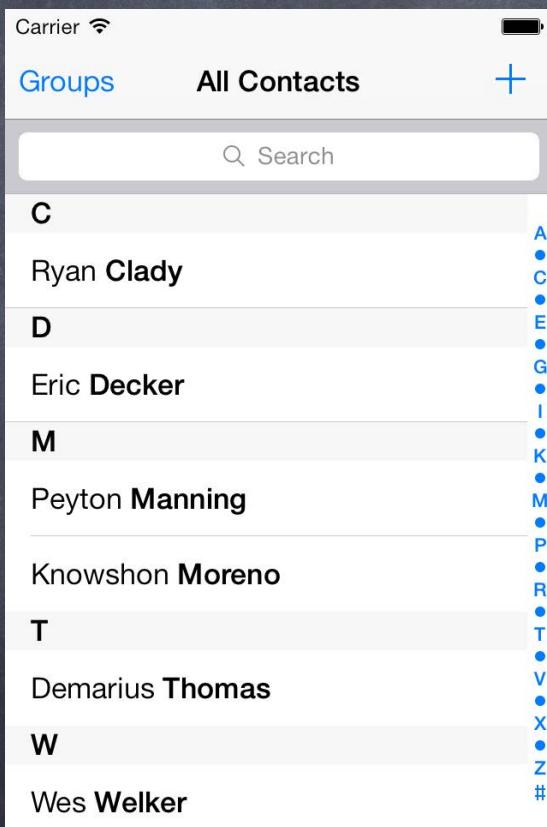
Demo: Twitter Client



CS193p  
Winter 2017

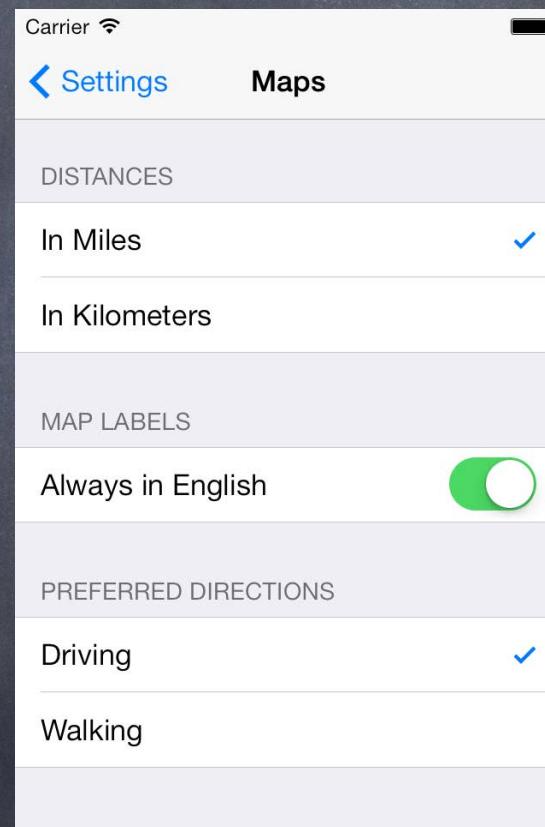
# UITableView

UITableViewStyle.plain



Dynamic (List)  
& Plain  
(ungrouped)

.grouped

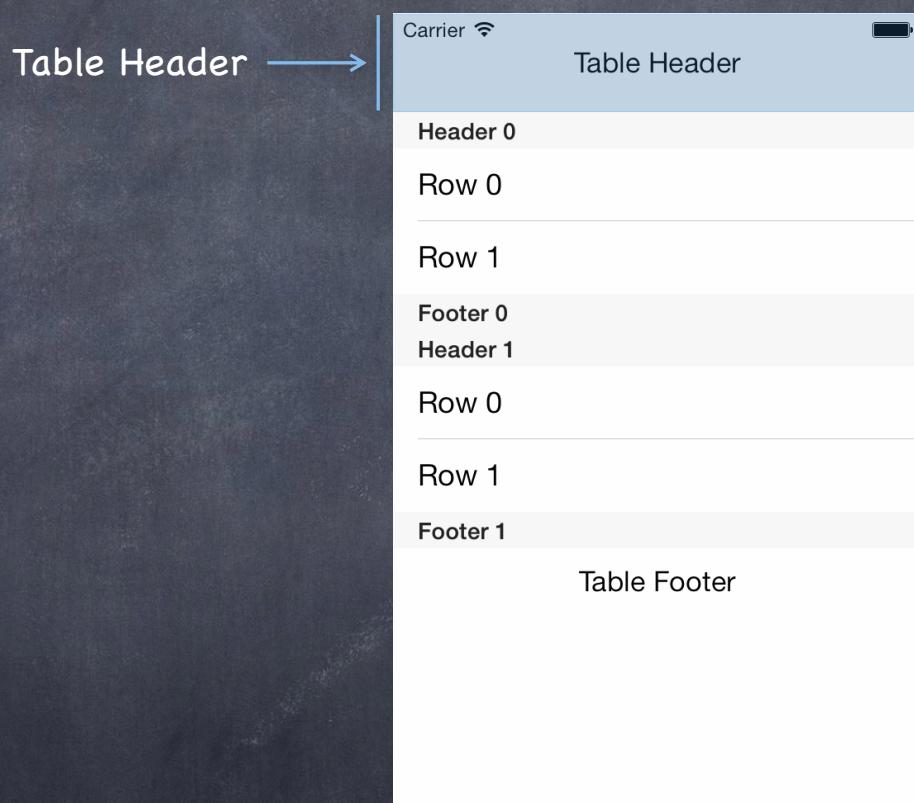


Static  
& Grouped



# UITableView

## Plain Style

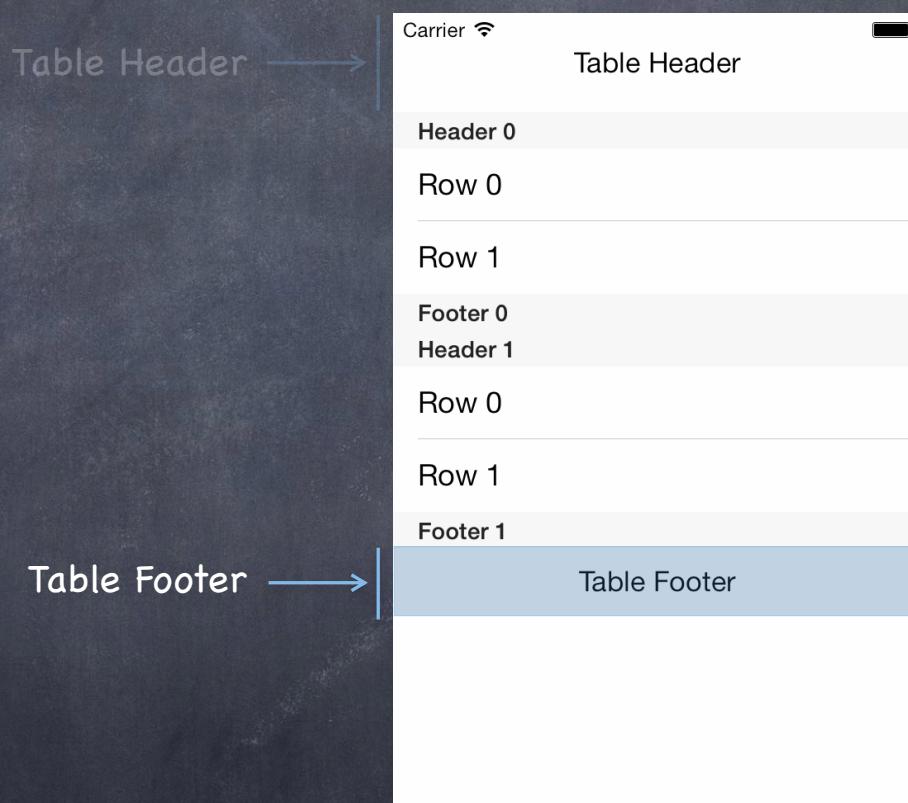


```
var tableHeaderView: UIView
```



# UITableView

## Plain Style

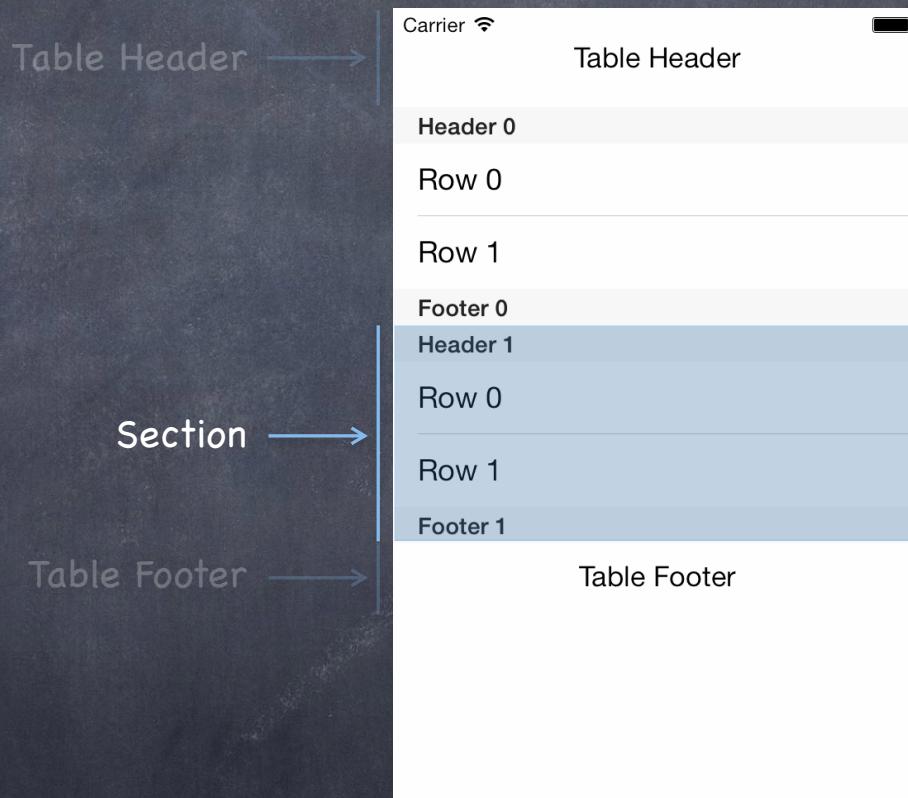


```
var tableFooterView: UIView
```



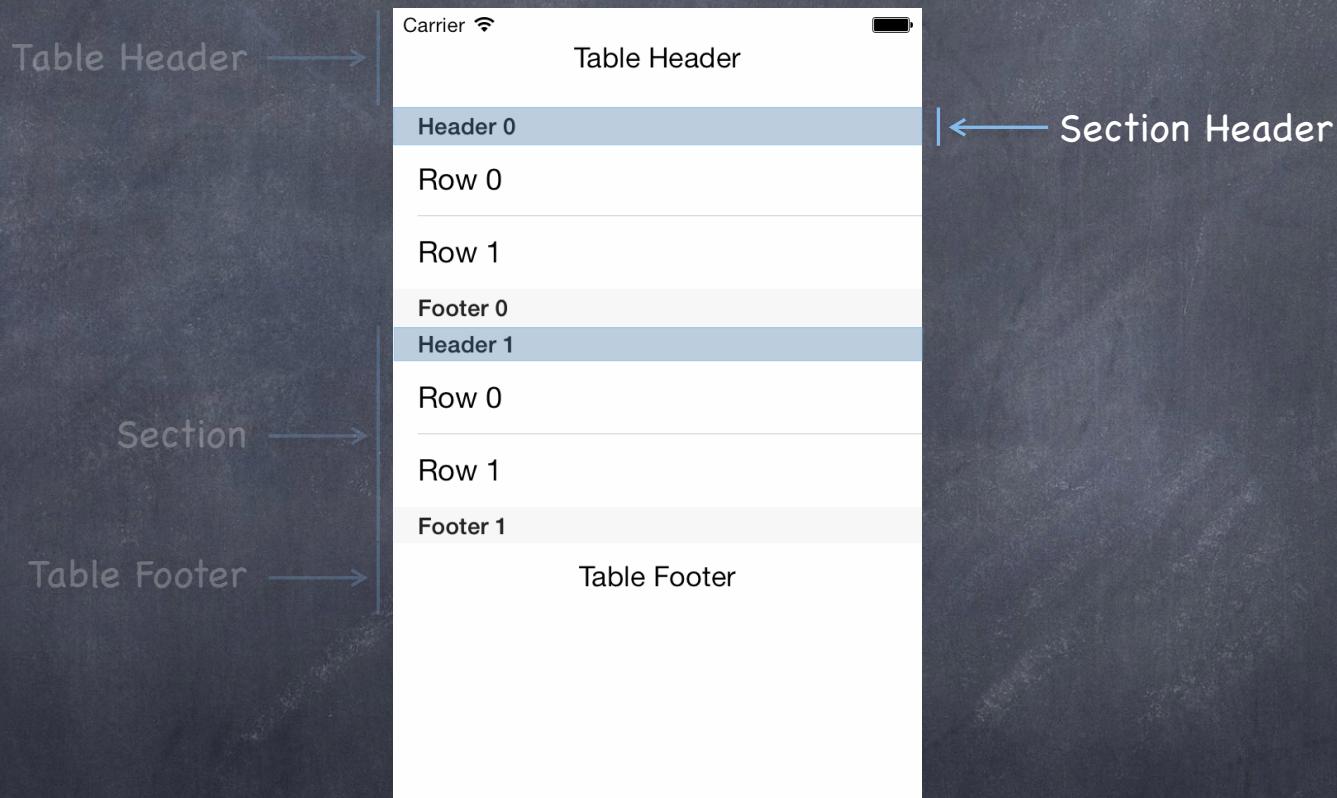
# UITableView

## Plain Style



# UITableView

## Plain Style



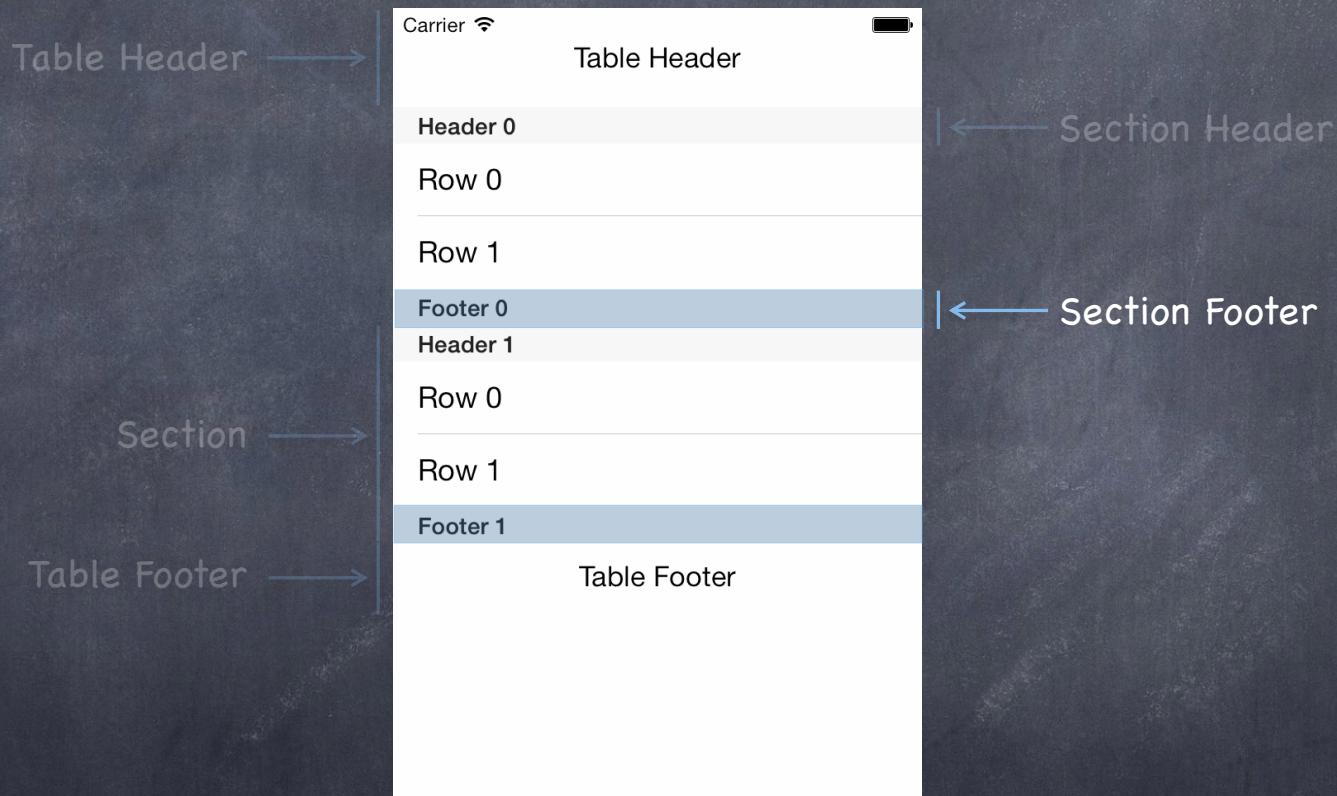
UITableViewDataSource's `tableView(tableView, titleForHeaderInSection: Int)`



CS193p  
Winter 2017

# UITableView

## Plain Style

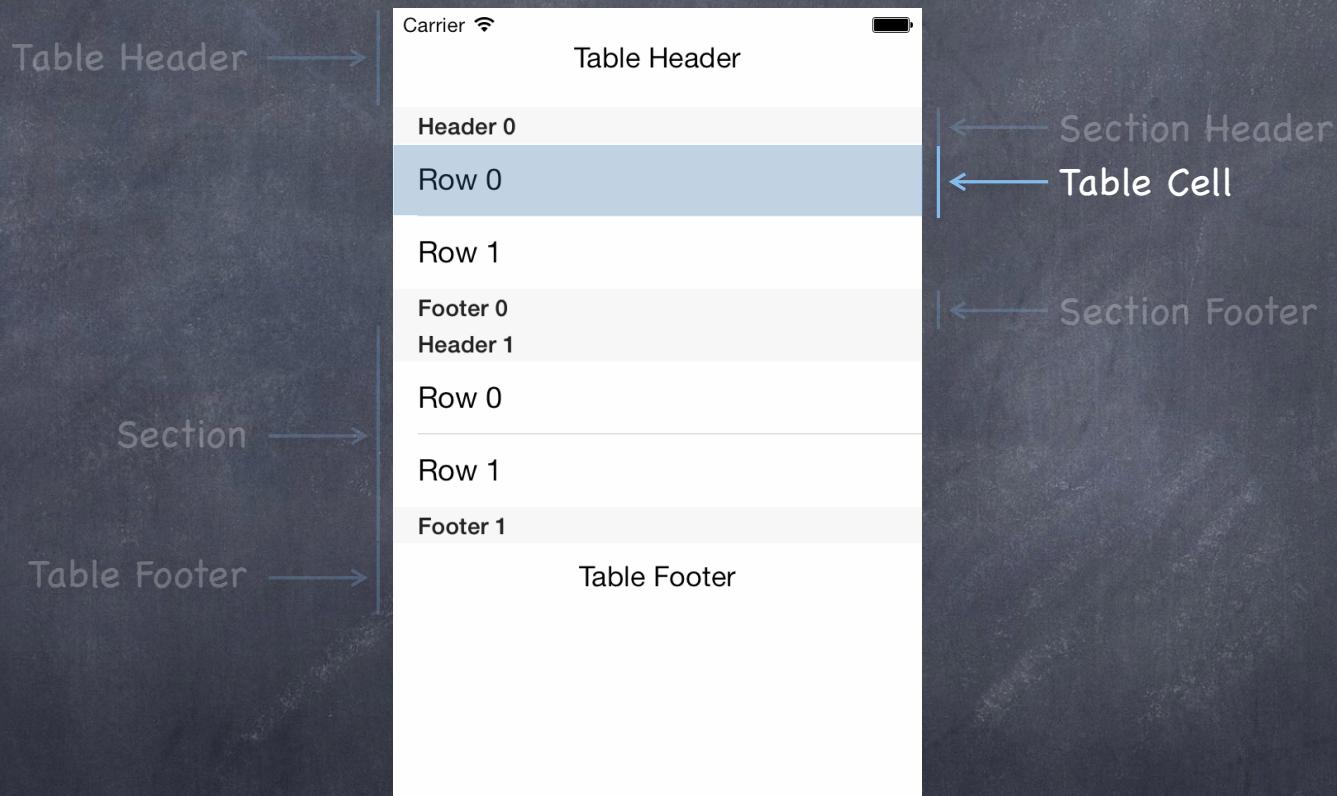


`UITableViewDataSource's tableView(UITableView, titleForFooterInSection: Int)`



# UITableView

## Plain Style

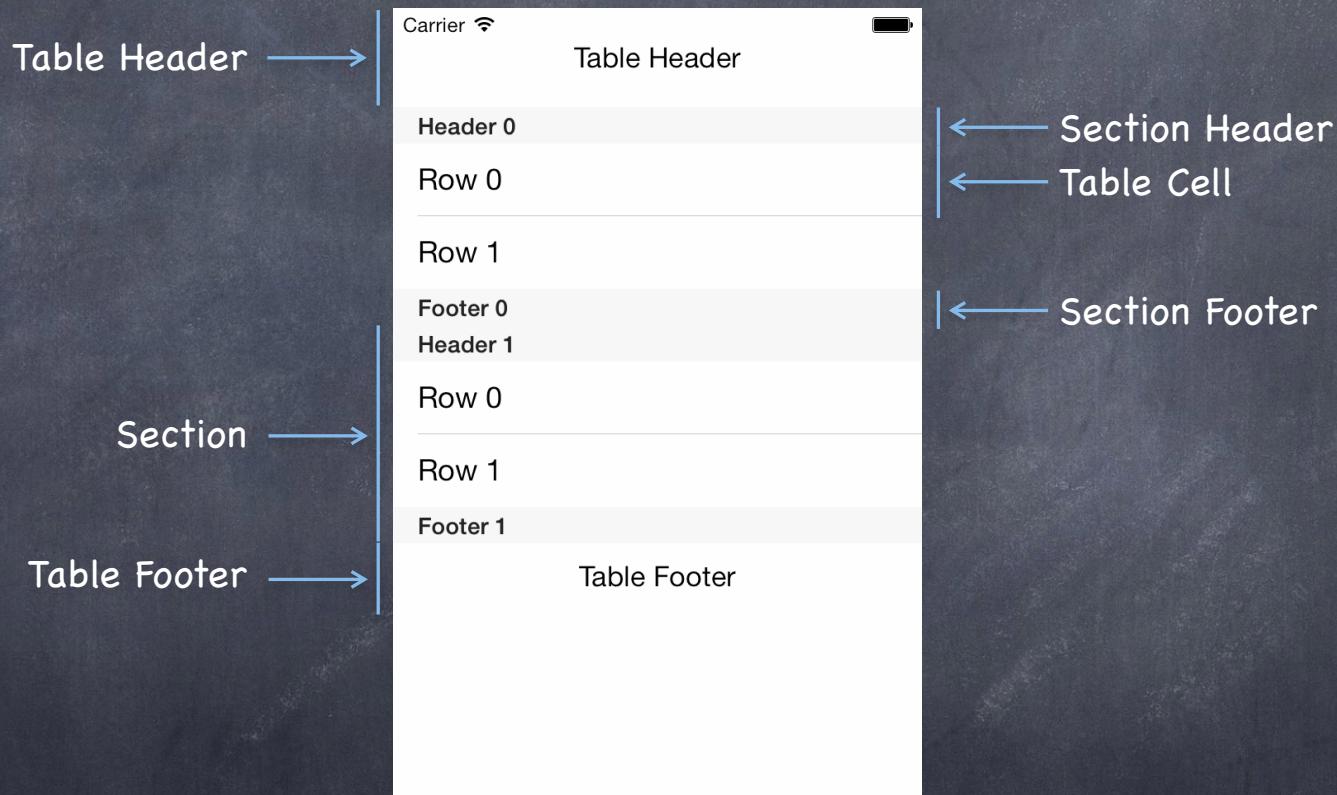


UITableViewDataSource's `tableView(UITableView, cellForRowAt indexPath:)`



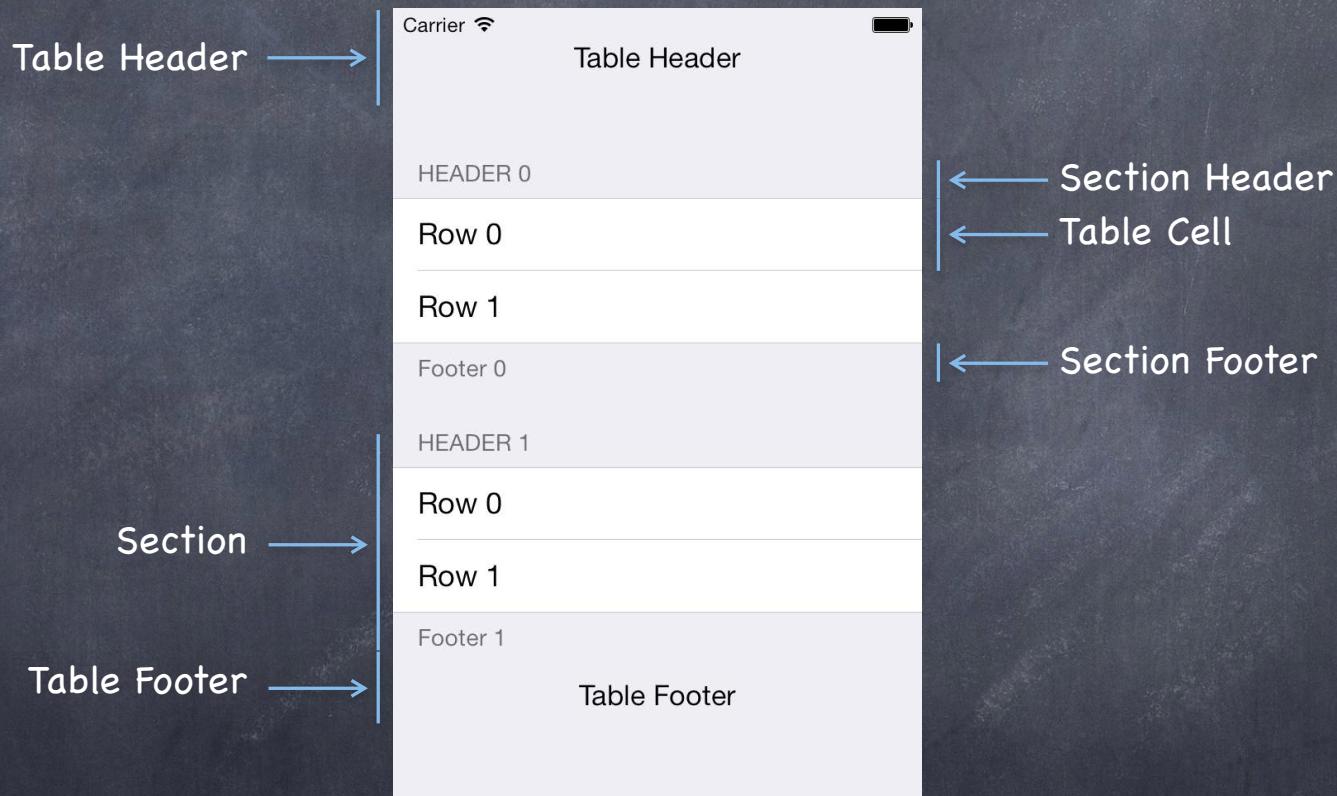
# UITableView

## Plain Style

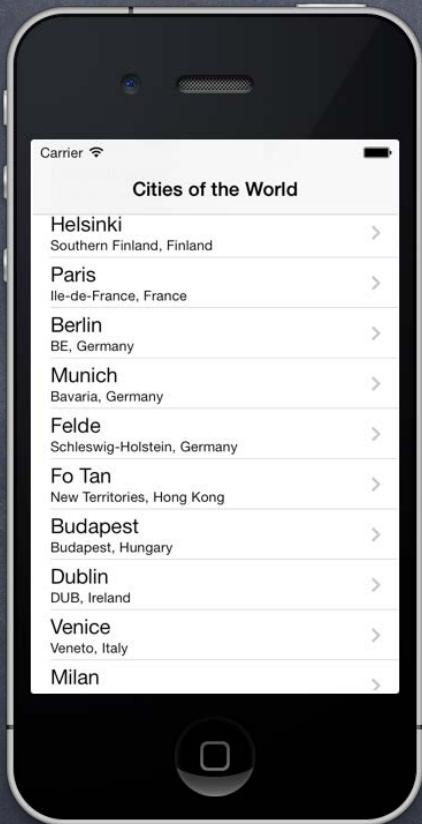


# UITableView

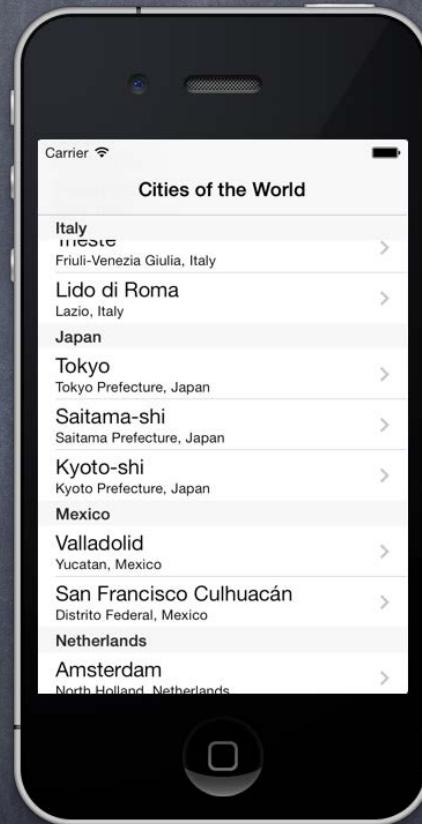
## Grouped Style



# Sections or Not



No Sections

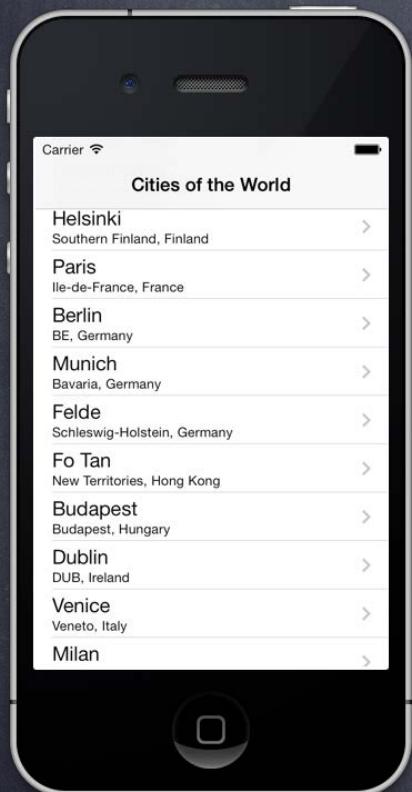


Sections



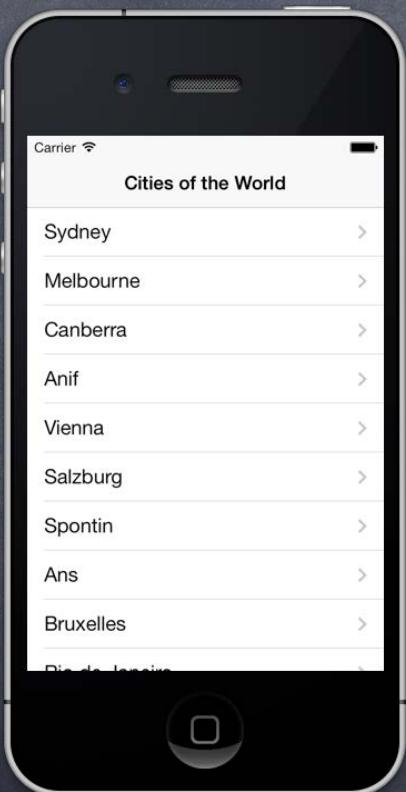
CS193p  
Winter 2017

# Cell Type



Subtitle

UITableViewCellStyle.subtitle



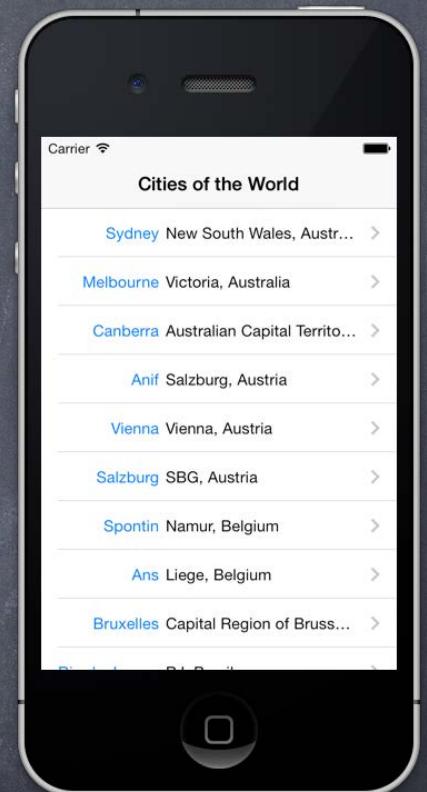
Basic

.default



Right Detail

.value1



Left Detail

.value2



CS193p

Winter 2017

The class `UITableViewController` provides a convenient packaging of a `UITableView` in an MVC.

It's mostly useful when the `UITableView` is going to fill all of `self.view` (in fact `self.view` in a `UITableViewController` is the `UITableView`).

You can add one to your storyboard simply by dragging it from here.

A screenshot of the Xcode interface showing a storyboard. At the top, the title bar says "TVCExample: Ready". The storyboard contains a single view controller placeholder with a blue rounded rectangle. A callout bubble points to this placeholder with the text: "You can add one to your storyboard simply by dragging it from here.". To the right of the storyboard, there is a sidebar titled "No Selection" containing icons for various view controllers: Navigation Controller, Table View Controller, Collection View Controller, and Tab Bar Controller. The bottom of the screen shows the Xcode toolbar with icons for file operations, a search field, and zoom controls. The status bar at the very bottom indicates "View as: iPhone 7 (wC hR)" and "75%".

Controller: (subclass of) UITableViewController  
Controller's **view** property: the UITableView

The screenshot shows the Xcode interface with a storyboard file open. The storyboard contains a single UITableViewController. The table view has a prototype cell section labeled "Prototype Cells". A blue callout bubble points from the explanatory text above to this table view. The right side of the screen shows the Attributes Inspector, which displays various settings for the controller, including "Simulated Metrics" (Size Inferred, Status Bar Inferred, Top Bar Inferred, Bottom Bar Inferred), "Table View Controller" (Selection checked, Clear on Appearance, Refreshing Disabled), and "View Controller" (Title empty, Is Initial View Controller unchecked). Below these are icons and descriptions for Navigation Controller, Table View Controller, Collection View Controller, and Tab Bar Controller. The bottom of the screen shows the Xcode toolbar with zoom controls and other standard icons.

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Custom Class  
Class: UITableViewControl...  
Module: None

Identity  
Storyboard ID:   
Restoration ID:   
 Use Storyboard ID

User Defined Runtime Attributes  
Key Path | Type | Value

+ - Document  
Label: Xcode Specific Label  
Object ID: ab2-9w-rly  
Lock: Inherited - (Nothing)

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

CS193p Winter 2017

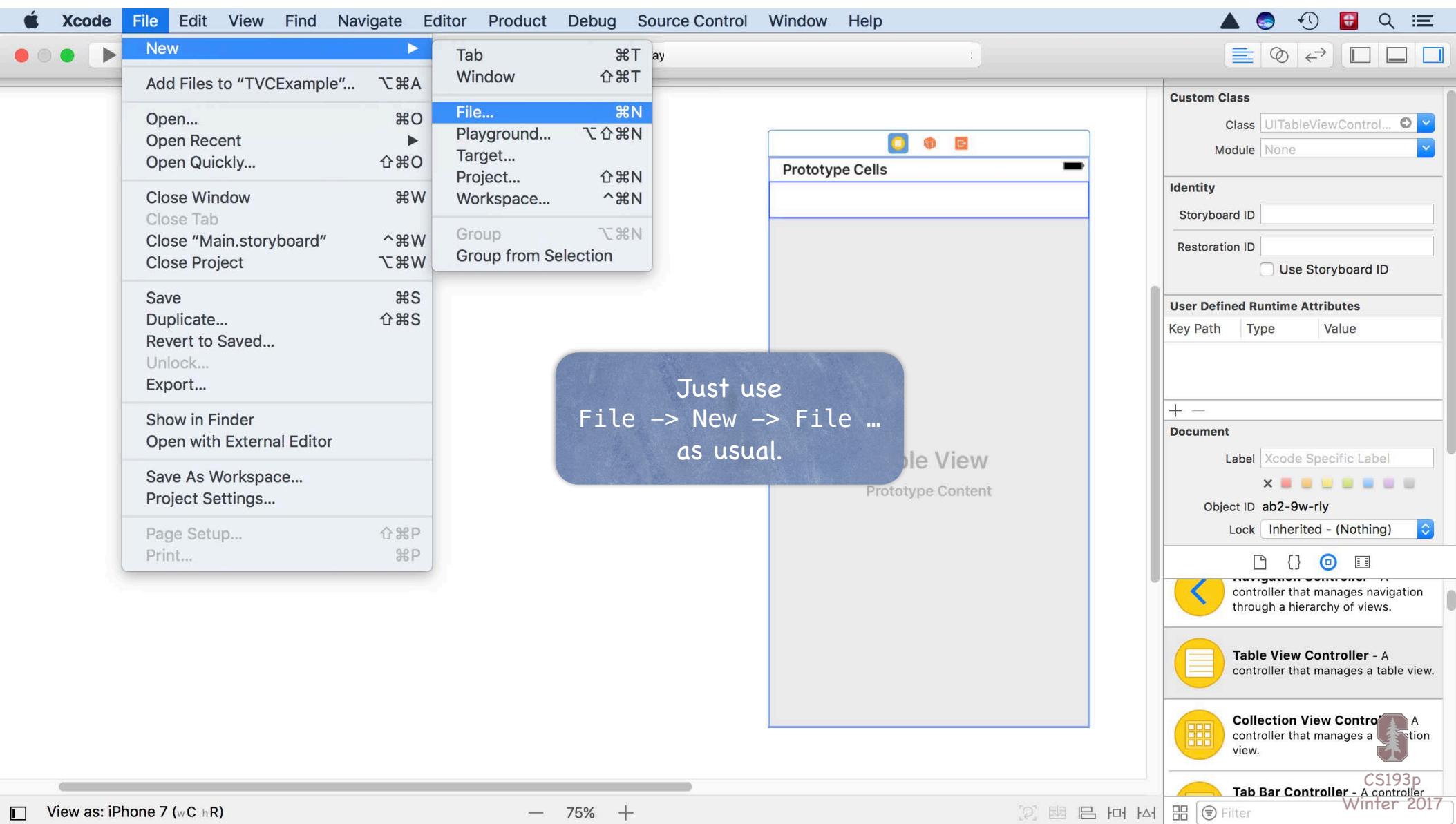
Prototype Cells

Prototype Content

Table View

Like any other View Controller, you'll want to set its class in the Identity Inspector.

View as: iPhone 7 (wC hR) 75% Filter

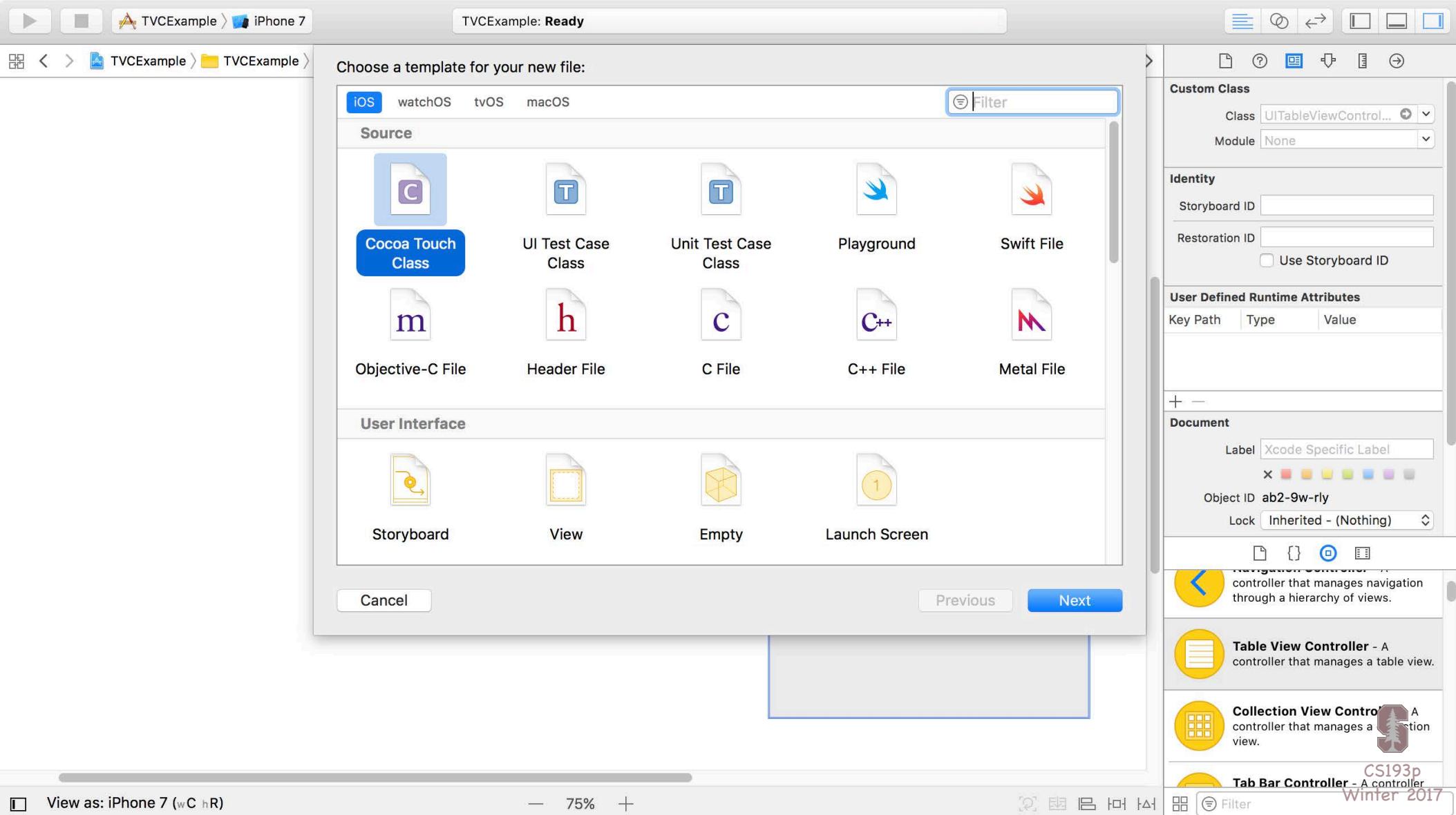


View as: iPhone 7 (wC hR)

- 75% +

Filter

CS193p  
Winter 2017



TVCEExample > iPhone 7

TVCEExample: Ready

Choose options for your new file:

Class: UITableViewController

Subclass of: UITableViewController

Language: UITableViewController

Cancel Previous Next

Custom Class

Class: UITableViewController

Module: None

Identity

Storyboard ID: [ ]

Restoration ID: [ ]

Use Storyboard ID

User Defined Runtime Attributes

Label: Xcode Specific Label

X Object ID: ab2-9w-rly

Lock: Inherited - (Nothing)

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.

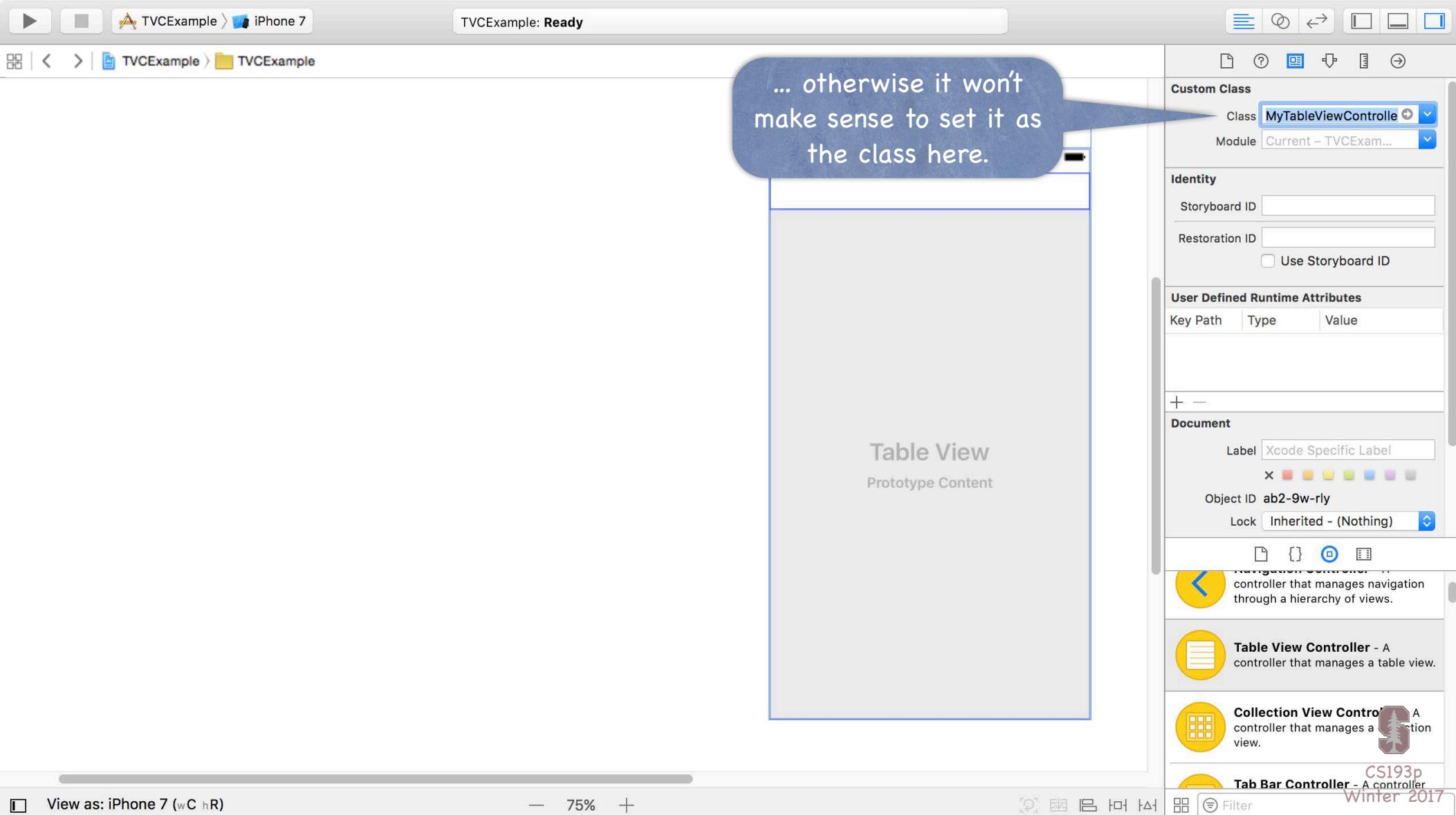
Tab Bar Controller - A controller

CS193p Winter 2017

Make sure you set the superclass to UITableViewController

View as: iPhone 7 (wC hR) — 75% +

... otherwise it won't  
make sense to set it as  
the class here.



Your UITableViewController subclass will also serve as the UITableView's dataSource and delegate (more on this in a moment).

You can see that if you right-click the Controller here.

Table View  
Prototype Content

Custom Class  
Class MyTableViewController  
Module Current – TVCEExam...

User Defined Runtime Attributes

Key Path	Type	Value

+ - Document

Label Xcode Specific Label  
Object ID ab2-9w-rly  
Lock Inherited - (Nothing)

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

CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter

TVCEExample > iPhone 7

TVCEExample: Ready

Custom Class  
Class MyTableViewController  
Module Current – TVCEExam...

Identity

User Defined Runtime Attributes

Key Path Type Value

Document  
Label Xcode Specific Label  
Object ID ab2-9w-rly  
Lock Inherited - (Nothing)

Navigation Controller 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

CS193p Winter 2017

View as: iPhone 7 (wC hR) — 75% +

You can see that if you right-click the Controller here.

dataSource and delegate properties

If you use UITableView without UITableViewController, you'll have to wire these up yourself.

You can edit attributes of the UITableView by inspecting it.

Remember that you can shift-right-click (or ctrl-shift-left-click) on things to pick exactly what you want from what is under the mouse.

This makes it easier to pick the Controller, the table view, a cell in a table view, or even a view inside a cell in the table view.

### Table View

Prototype Content



#### Table View

Content	Dynamic Prototypes
Prototype Cells	1
Style	Plain
Separator	Default
+ Separator	Default
Separator Inset	Default
Selection	Single Selection
Editing	No Selection During Ed...

#### Section Index

Display Limit	0
+ Text	Default
+ Background	Default
+ Tracking	Default

#### ScrollView

Style	Default
Shows Horizontal Indicators	
+ Text	Default

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

CS193p

Winter 2017

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View

Content: Dynamic Prototypes

Prototype Cells:

- Style: Plain (selected)
- Grouped

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

+ Text: Default

+ Background: Default

+ Tracking: Default

ScrollView

Style: Default

Scroll Indicators: Shows Horizontal 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

View as: iPhone 7 (wC hR) 75% +

CS193p Winter 2017

One important attribute is the Plain vs. Grouped style ...

Table View  
Prototype Content

The screenshot shows the Xcode interface with a storyboard open. On the left, there's a table view with three prototype cells. On the right, the 'Table View' section of the Attributes Inspector is visible. A blue callout bubble points from the text 'One important attribute is the Plain vs. Grouped style ...' to the 'Style' dropdown menu, which has 'Plain' selected. Other options like 'Grouped' and 'Default' are also shown. The storyboard preview on the left shows the table view with three empty rows. The bottom status bar indicates 'View as: iPhone 7 (wC hR)' and '75%'. The bottom right corner of the screen shows the text 'CS193p Winter 2017'.

TVCEExample > iPhone 7 TVCEExample: Ready

Table View

Content: Dynamic Prototypes

Prototype Cells: 1

Style: Plain ✓ Grouped

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

+ Text: Default

+ Background: Default

+ Tracking: Default

ScrollView

Style: Default

Shows Horizontal 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

CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter

The screenshot shows the Xcode interface with a storyboard open. The storyboard contains a single table view with the title "Prototype Cells". Inside the table view, there is a section labeled "Table View" with the subtitle "Prototype Content". On the right side of the screen, the "Attributes Inspector" is visible, displaying various properties for the table view. Under the "Table View" section, the "Content" dropdown is set to "Dynamic Prototypes". The "Style" dropdown is currently set to "Plain" (which is checked), but "Grouped" is also listed. Other visible properties include "Separator" (set to "Default"), "Separator Inset" (set to "Default"), "Selection" (set to "Single Selection"), and "Editing" (set to "No Selection During Ed..."). Below the table view properties, there are sections for "Section Index" (with a display limit of 0), "Text" (set to "Default"), "Background" (set to "Default"), and "Tracking" (set to "Default"). Further down, the "ScrollView" section is shown with its style set to "Default" and the "Shows Horizontal Indicator" checkbox checked. At the bottom of the screen, there are several icons for navigating between storyboard scenes and a "Filter" button. The status bar at the bottom indicates "View as: iPhone 7 (wC hR)" and a zoom level of "75%".

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Grouped

Table View  
Prototype Content

PROTOTYPE CELLS

Table View

Content Dynamic Prototypes

Prototype Cells 1

Style Grouped

Separator Default

Separator Inset Default

Selection Single Selection

Editing No Selection During Ed...

Section Index

Display Limit 0

+ Text Default

+ Background Default

+ Tracking Default

ScrollView

Style Default

Scroll Indicators Shows Horizontal Indicators

Navigation Controller 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

CS193p Winter 2017

View as: iPhone 7 (wC hR) — 75% +

Filter

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Grouped

Another important attribute is Dynamic versus Static ...

Table View  
Prototype Content

PROTOTYPE CELLS

Table View

Dynamic Prototypes

Static Cells

Style: Grouped

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

ScrollView

Style: Default

Show Horizontal Indicators

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

CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Grouped

Another important attribute is Dynamic versus Static ...

Table View  
Prototype Content

Table View

Content ✓ Dynamic Prototypes  
Static Cells

Style Grouped  
Separator Default  
Separator Inset Default  
Selection Single Selection  
Editing No Selection During Ed...

Section Index

Display Limit 0

Text Default

Background Default

Tracking Default

ScrollView

Style Default

Shows Horizontal 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

CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter

The image shows a screenshot of the Xcode interface during a lecture. On the left, a storyboard scene is displayed with a table view containing three prototype cells. A blue callout bubble points to the word 'Grouped' in the Attributes Inspector, which is highlighted. Another blue callout bubble points to the 'Dynamic Prototypes' checkbox in the same inspector, which is also highlighted. The storyboard scene is titled 'Table View' and has 'Prototype Content'. The bottom right corner of the screen features the Stanford University logo and the text 'CS193p Winter 2017'.

TVCEExample > iPhone 7

TVCExample: Ready

TVCEExample > TVCEExample

Table View

Content: Static Cells

Sections: 1

Style: Grouped

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

ScrollView

Style: Default

Show Horizontal Indicat

Navigation Controller - 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

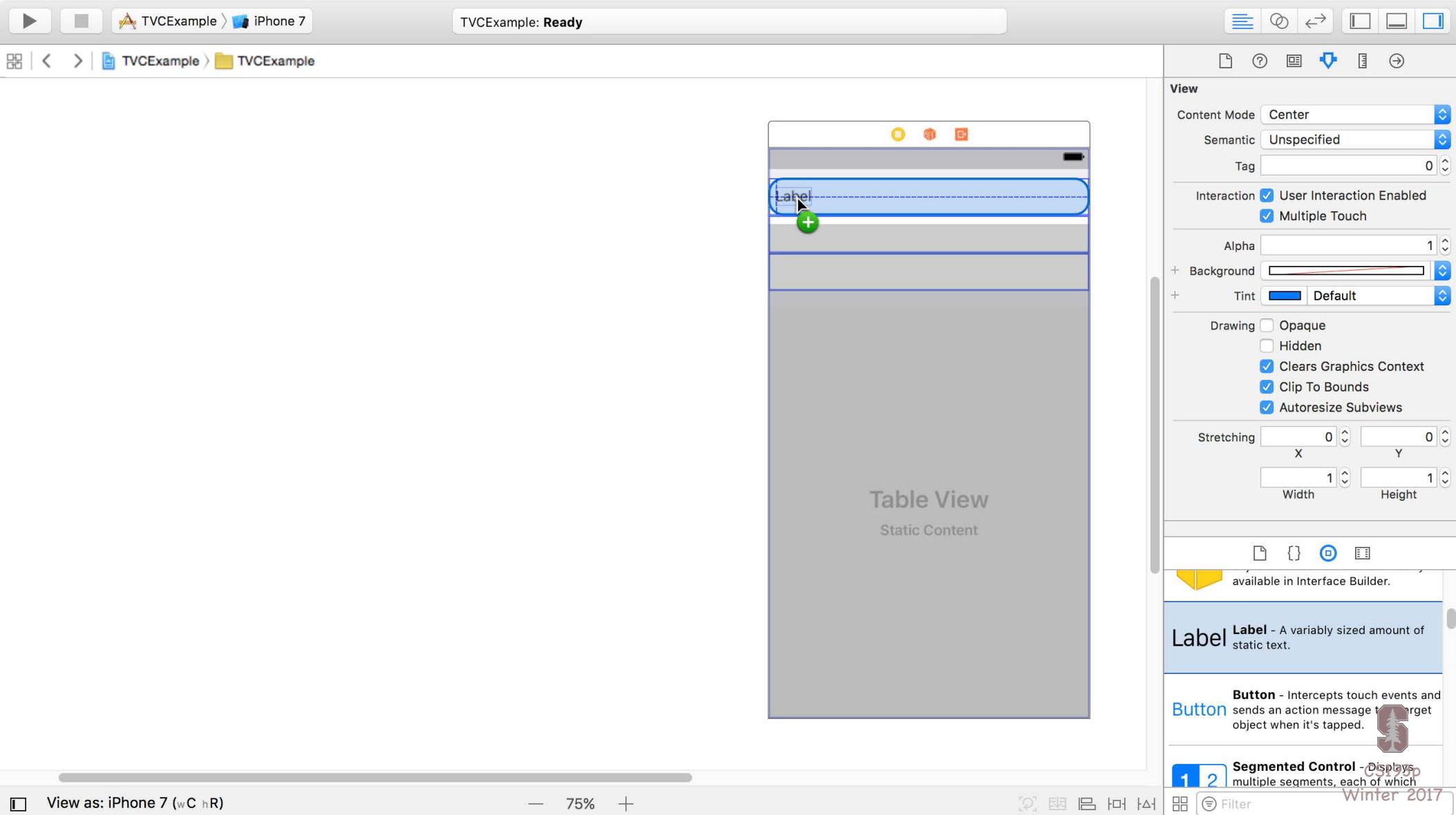
CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter

**Grouped**

"Static" means that these cells are set up in the storyboard only. You can edit them however you want including dragging buttons, etc., into them (and wiring up outlets to the Controller).

The screenshot shows the Xcode interface with a storyboard open. A Table View component is selected, displaying five static cells. A blue callout bubble points to the top cell with the word "Grouped". Another blue callout bubble contains the text: "'Static' means that these cells are set up in the storyboard only. You can edit them however you want including dragging buttons, etc., into them (and wiring up outlets to the Controller)." On the right side of the screen, the Attributes Inspector is open, showing settings for the Table View such as Content (Static Cells), Sections (1), and Style (Grouped). Below the Attributes Inspector, there are sections for Section Index and ScrollView, along with descriptions of other controller types like Navigation Controller, Table View Controller, Collection View Controller, and Tab Bar Controller. The bottom of the screen shows the Xcode toolbar with various icons and the current view configuration as 'iPhone 7 (wC hR)' at 75% zoom.



TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Label

Text Plain Label

Color Default

Font System 17.0

Alignment

Lines 1

Behavior  Enabled  Highlighted

Baseline Align Baselines

Line Break Truncate Tail

Autoshrink Fixed Font Size

Tighten Letter Spacing

Highlighted Default

Shadow Default

Shadow Offset 0 -1

Width Height

View

available in Interface Builder.

Label Label - A variably sized amount of static text.

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

Segmented Control Segmented Control - Displays multiple segments, each of which

1 2 Filter Winter 2017

Table View Static Content

— 75% +

View as: iPhone 7 (wC hR)

A screenshot of the Xcode Interface Builder interface. The main canvas shows a UITableView with the title "Table View" and subtitle "Static Content". Inside the table view, there is a single label with the text "Label". The right side of the screen features the standard Xcode property inspector for the selected "Label" object. The "Label" section in the inspector includes settings for text (Plain), color (Default), font (System 17.0), alignment, lines (1), behavior (Enabled checked, Highlighted unchecked), baseline, line break, autoshrink, and shadow. Below the property inspector, there are sections for "View" (with a note that it's available in Interface Builder) and two collapsed sections for "Label" and "Button". At the bottom of the interface, there are zoom controls (— 75% +) and a "View as" dropdown set to "iPhone 7 (wC hR)".

TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Label

Text Plain Feature Enabled

Color Default

Font System 17.0

Alignment

Lines 1

Behavior  Enabled  Highlighted

Baseline Align Baselines

Line Break Truncate Tail

Autoshrink Fixed Font Size

Tighten Letter Spacing

Highlighted Default

Shadow Default

Shadow Offset 0 -1

Width Height

View

available in Interface Builder.

Label Label - A variably sized amount of static text.

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

Segmented Control Segmented Control - Displays multiple segments, each of which

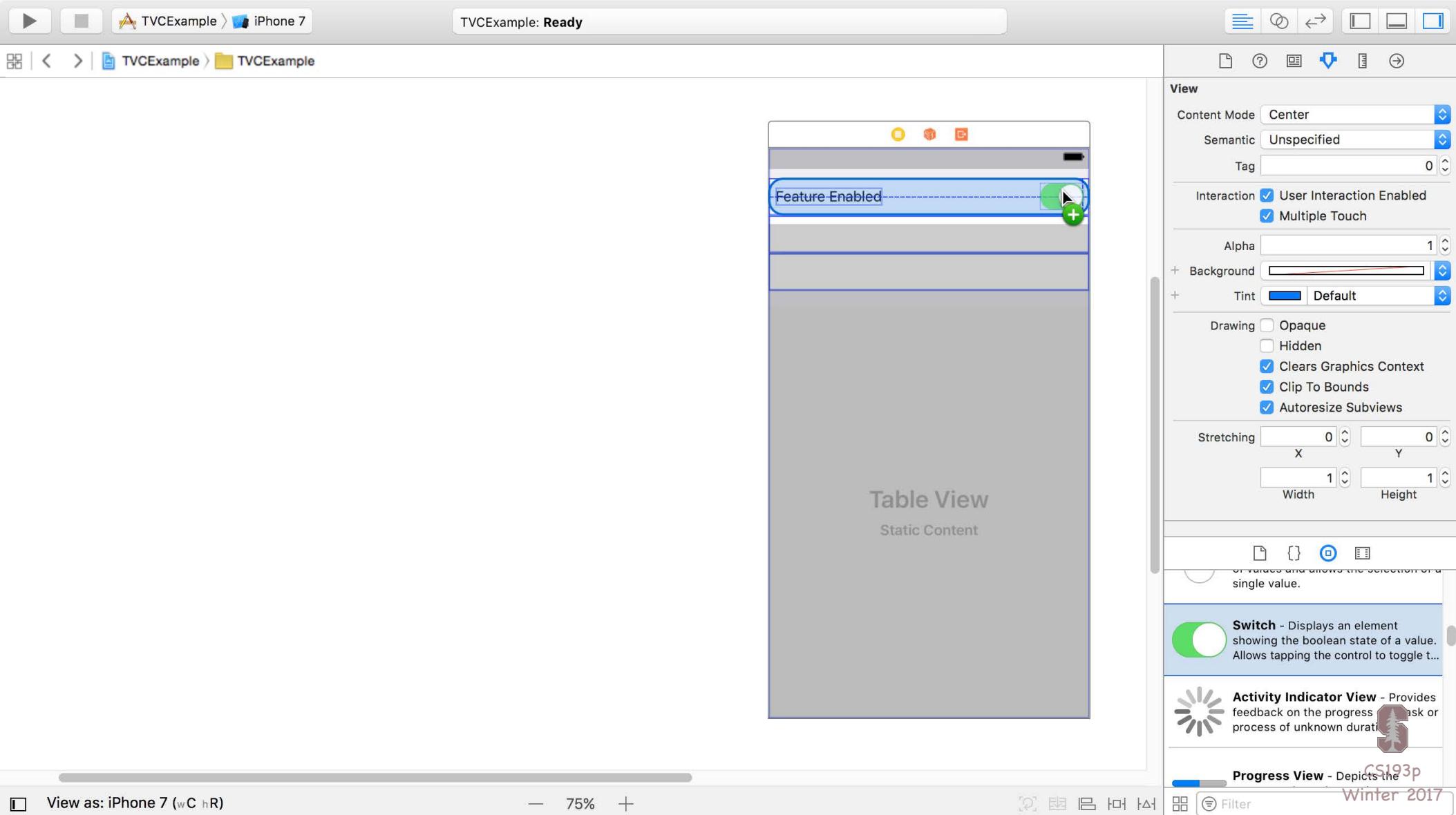
1 2 Filter Winter 2017

Table View Static Content

— 75% +

View as: iPhone 7 (wC hR)





TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Switch

Value: On

+ On Tint: Default

+ Thumb Tint: Default

+ On Image: On Image

+ Off Image: Off Image

Control

Alignment:

Horizontal

Vertical

State:

Selected

Enabled

Highlighted

View

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

Table View  
Static Content

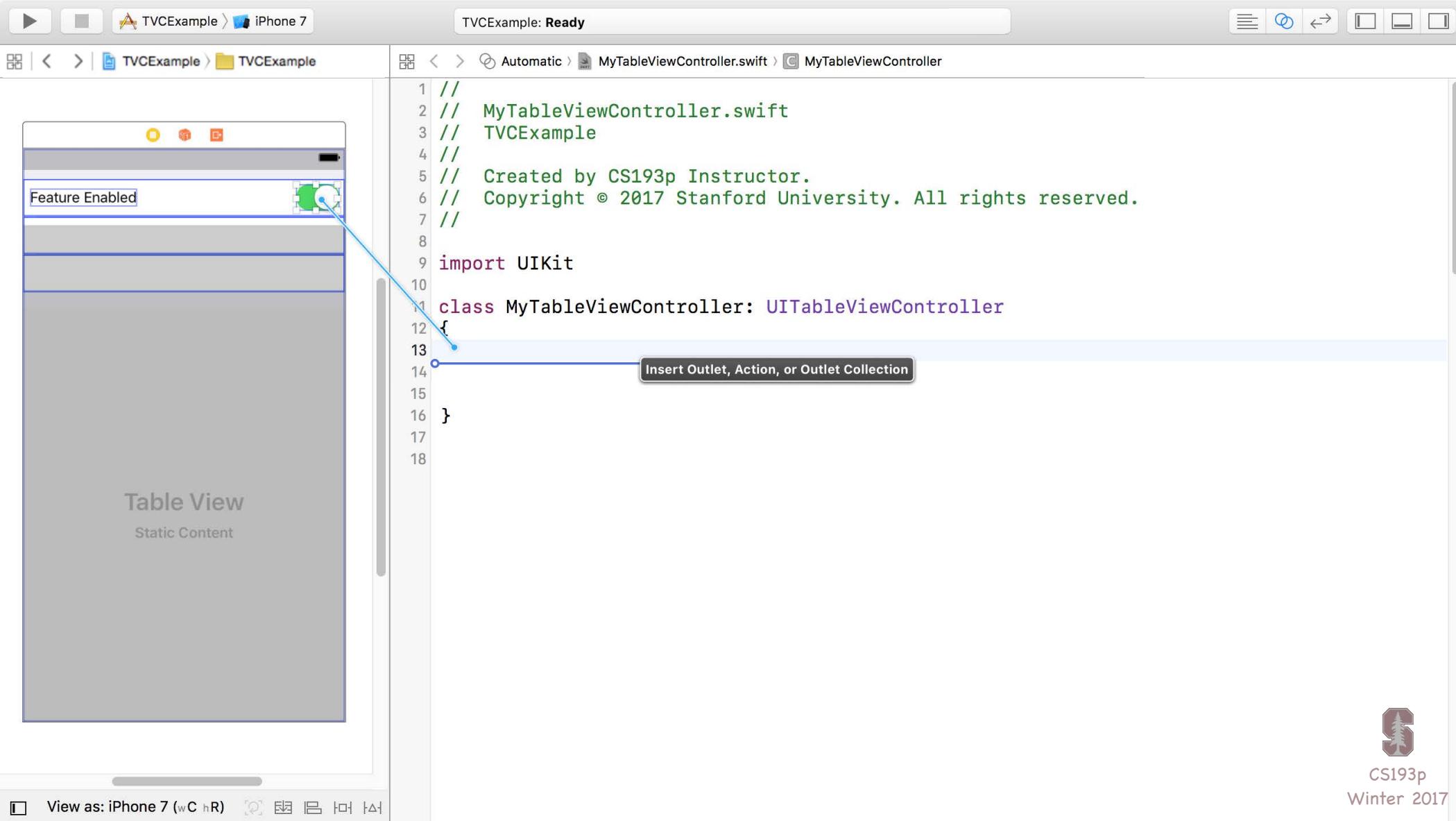
Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle the value.

Activity Indicator View - Provides feedback on the progress of a task or process of unknown duration.

Progress View - Depicts the progress of a task.

CS193p Winter 2017

View as: iPhone 7 (wC hR) 75% Filter



CS193p  
Winter 2017

TVCEExample > iPhone 7

TVCExample: Ready

TVCEExample > TVCEExample

Automatic > MyTableViewController.swift > MyTableViewController

MyTableViewController.swift

```
1 // MyTableViewController.swift
2 // TVCEExample
3 //
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13 }
14
15
16
17
18 }
```

Feature Enabled

Connection: Outlet

Object: My Table View Contr...

Name: featureEnabledSwitch

Type: UISwitch

Storage: Weak

Cancel Connect

Table View

Static Content

View as: iPhone 7 (wC hR)



CS193p

Winter 2017

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with "TVCEExample" selected.
- Editor:** Displays the "MyTableViewController.swift" file content. The code is as follows:

```
1 // MyTableViewController.swift
2 // TVCEexample
3 /**
4 * Created by CS193p Instructor.
5 * Copyright © 2017 Stanford University. All rights reserved.
6 */
7
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController {
12
13     @IBOutlet weak var featureEnabledSwitch: UISwitch!
14
15 }
16
17
18
```

The storyboard preview on the left shows a table view with a single row containing the text "Feature Enabled" and a green switch icon.

**Bottom Status Bar:** Shows "View as: iPhone 7 (wC hR)".



TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Switch

- Value: On
- + On Tint: Default
- + Thumb Tint: Default
- + On Image: On Image
- + Off Image: Off Image

Control

- Alignment:
  - Horizontal
  - Vertical
- State:
  - Selected
  - Enabled
  - Highlighted

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
  - Clip To Bounds
  - Autoresizes Subviews

Table View  
Static Content

Feature Enabled

Let's clear this UI out and look at another way to do table views.

View as: iPhone 7 (wC hR) — 75% +

Winter 2017

A different way to populate the UI of your table view is by dynamically providing the data at runtime.

The screenshot shows a Xcode interface with a storyboard file named "TVCEExample". The storyboard contains a single table view with five static cells. The right panel displays the "Table View" settings. Under "Dynamic Prototypes", the "Content" dropdown is set to "Static Cells". Other settings include "Style: Grouped", "Separator: Default", "Separator Inset: Default", "Selection: Single Selection", and "Editing: No Selection During Ed...". The "Section Index" section has a "Display Limit" of 0. The "ScrollView" section has "Shows Horizontal Indicator" and "Shows Vertical Indicator" checked, and "Bounces" checked. The "Touch" section has "Bounces Zoom" checked. The bottom right corner of the Xcode window shows the text "CS193p Winter 2017".

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View

Dynamic Prototypes

Content: Static Cells

Sections: 1

Style: Grouped

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

ScrollView

Style: Default

Scroll Indicators: Shows Horizontal Indicator, Shows Vertical Indicator

Scrolling: Scrolling Enabled

Bounce: Bounces, Bounce Vertically

Zoom: 1, Min

Touch: Bounces Zoom, Delays Content Touches

View as: iPhone 7 (wC hR)

75%

CS193p Winter 2017

A different way to populate the UI of your table view is by dynamically providing the data at runtime.

The screenshot shows the Xcode interface with a storyboard project named "TVCExample". The storyboard scene is titled "TVCExample: Ready". On the right, the Attributes Inspector displays settings for a "Table View" component, including "Content: Dynamic Prototypes", "Style: Grouped", and "Separator: Default". The storyboard itself shows a "Table View" with the title "Prototype Content" and "Prototype Cells". A blue callout bubble originates from the text "A different way to populate the UI of your table view is by dynamically providing the data at runtime." and points towards the "Table View" component in the storyboard. The bottom right corner of the screen features a watermark for "CS193p Winter 2017".

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View

Content: Dynamic Prototypes

Prototype Cells: Plain

Style: **Grouped**

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

+ Text: Default

+ Background: Default

+ Tracking: Default

ScrollView

Style: Default

Scroll Indicat...  Shows Horizontal Indicator  
 Shows Vertical Indicator

Scrolling  Scrolling Enabled  
 Paging Enabled  
 Direction Lock Enabled

Bounce  Bounces  
 Bounce Horizontally  
 Bounce Vertically

Zoom: 1

Touch  Bounces Zoom  
 Delays Content Touches

CS193p Winter 2017

... which we almost always use in Plain style.

Table View

Prototype Content

PROTOTYPE CELLS



View as: iPhone 7 (wC hR)

— 75% +

CS193p Winter 2017

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View

Content: Dynamic Prototypes

Prototype Cells: Plain

Style: Grouped

Separator: Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

+ Text: Default

+ Background: Default

+ Tracking: Default

ScrollView

Style: Default

Scroll Indicat...: Shows Horizontal Indicator  
Shows Vertical Indicator

Scrolling: Scrolling Enabled  
Paging Enabled  
Direction Lock Enabled

Bounce: Bounces  
Bounce Horizontally  
Bounce Vertically

Zoom: 1 Min 1

Touch: Bounces Zoom  
Delays Content Touches

CS193p Winter 2017

View as: iPhone 7 (wC hR)

— 75% +

PROTOTYPE CELLS

... which we almost always use in Plain style.

Table View

Prototype Content

The screenshot shows the Xcode interface with a storyboard project named 'TVCEExample'. A callout bubble originates from the 'Plain' style option in the 'Table View' section of the Attributes Inspector, pointing towards a text annotation in the storyboard preview area. The text annotation reads: "... which we almost always use in Plain style." The storyboard preview displays a table view with four prototype cells, each labeled 'PROTOTYPE CELLS'. The Attributes Inspector on the right shows various settings for the table view, including 'Content: Dynamic Prototypes', 'Prototype Cells: Plain', and 'Style: Grouped'. The 'Plain' style is highlighted with a blue selection bar. Other visible settings include separator styles, selection, and editing options. The bottom right corner of the screen features a watermark for 'CS193p Winter 2017'.

▶ ⌂ A TVCEExample > iPhone 7 TVCEExample: Ready

Table View

Content Dynamic Prototypes

Prototype Cells 3

Style Plain

Separator Default

Separator Inset Default

Selection Single Selection

Editing No Selection During Ed...

Section Index

Display Limit 0

+ Text Default

+ Background Default

+ Tracking Default

ScrollView

Style Default

Scroll Indicat...  Shows Horizontal Indicator  
 Shows Vertical Indicator

Scrolling  Scrolling Enabled  
 Paging Enabled  
 Direction Lock Enabled

Bounce  Bounces  
 Bounce Horizontally  
 Bounce Vertically

Zoom 1 Min 1

Touch  Bounces Zoom  
 Delays Content Touches

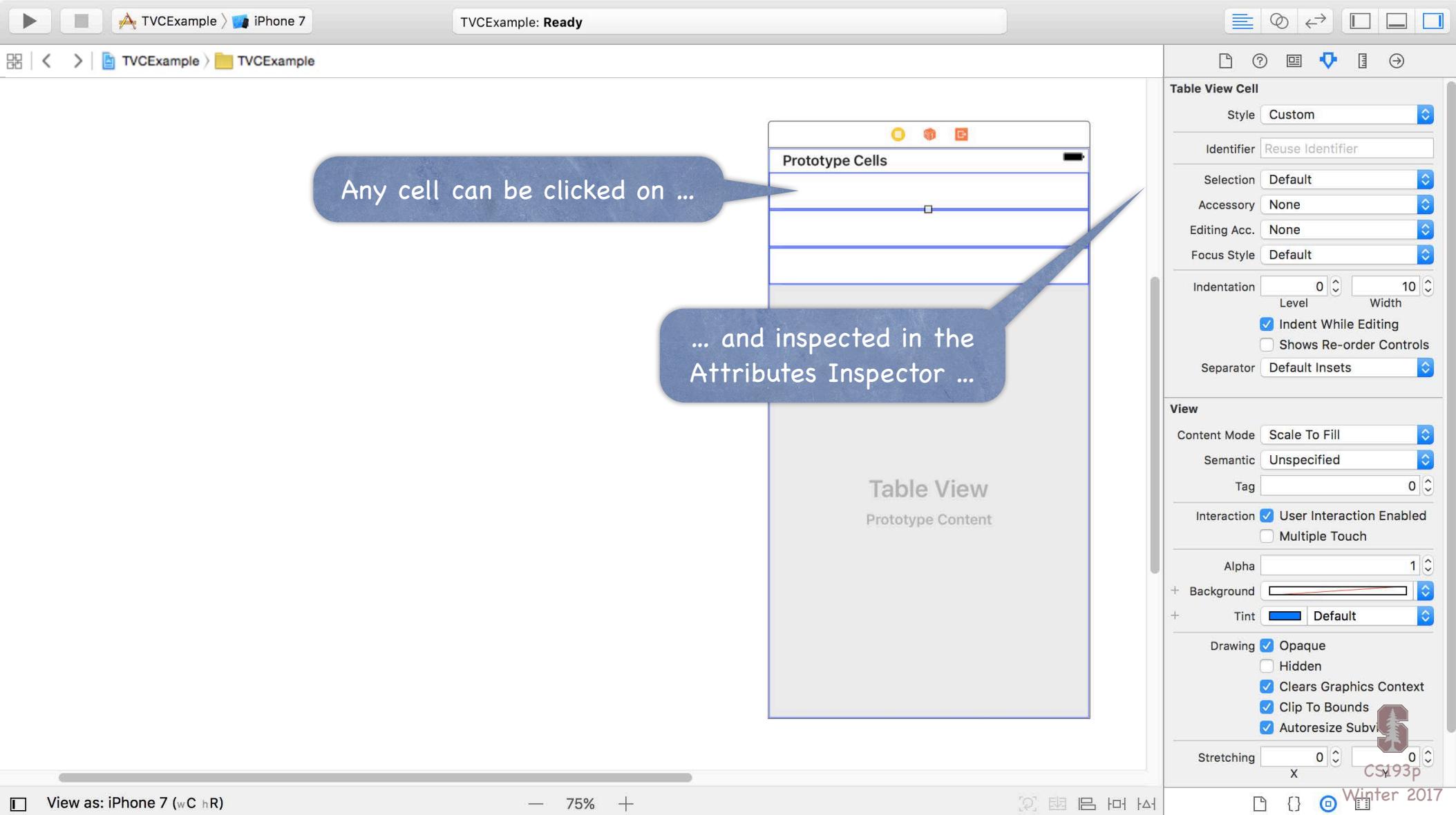
CS193p Winter 2017

Table View Prototype Content

These cells are now templates which will be repeated for however many rows are needed to display the data in MVC's Model.

View as: iPhone 7 (wC hR) — 75% +





TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

Style: Custom

Identifier: Basic

Selection: Right Detail

Accessory: Left Detail

Accessory Style: Subtitle

Editing Access: None

Focus Style: Default

Indentation: Level 0, Width 10

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 Swatch]

+ Tint: [Color Swatch] Default

Drawing

Opaque

Hidden

Clears Graphics Context

Clip To Bounds

Autoresizes Subviews

Stretching: X 0, Y 0

CS193p Winter 2017

Prototype Cells

Prototype Content

Here's where you can set the style of the cell.

View as: iPhone 7 (wC hR) 75%

TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Subtitle cell style

The screenshot shows the Xcode interface with a storyboard file open. A callout bubble points from the text "Subtitle cell style" to a specific cell in a table view prototype. The table view is titled "Table View" and has "Prototype Content". The selected cell is labeled "Title" and "Subtitle". The right-hand panel displays the "Table View Cell" settings for this cell, including:

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

The "View" section includes:

- Content Mode: Scale To Fill
- Semantic: Unspecified
- Tag: 0
- User Interaction Enabled
- Multiple Touch
- Alpha: 1
- + Background: Red color swatch
- + Tint: Blue color swatch

The "Drawing" section includes:

- Opaque
- Hidden
- Clears Graphics
- Clip To Bounds
- AutoresizesSubviews

At the bottom, there are zoom controls (75%), a toolbar, and a status bar indicating "View as: iPhone 7 (wC hR)".

CS193p Winter 2017

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

Style: Subtitle

Image: Image

Identifier: Reuse Identifier

Selection: Default

Accessory:  None

Editing Accessory:  Detail Disclosure

Focus Style: Checkmark

Indentation: Detail

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 Swatch]

+ Tint: [Color Swatch] Default

Drawing:  Opaque

Hidden

Clears Graphics

Clip To Bounds

Autoresizes Subviews

Stretching: 0

CS193p Winter 2017

You can also set a symbol to appear on the right of the cell.

Prototype Cells

Title  
Subtitle

Table View

Prototype Content

View as: iPhone 7 (wC hR) — 75% +

The screenshot shows the Xcode interface with the storyboard editor open. A callout bubble points from the text "You can also set a symbol to appear on the right of the cell." towards the prototype cell in the table view. The storyboard contains a single table view with one prototype cell. The cell has a subtitle and a disclosure indicator icon on the right. The right-hand sidebar displays the "Table View Cell" settings for this cell, including options for accessory types like "Detail Disclosure". A large blue callout bubble originates from the text "You can also set a symbol to appear on the right of the cell." and points towards the disclosure indicator icon in the prototype cell.

The screenshot shows the Xcode interface with a storyboard file open. The main area displays a Table View containing a single Prototype Cell. The cell has a blue header labeled "Title" and a white body labeled "Subtitle". A blue callout bubble points from the right side of the cell towards the Attributes Inspector on the right, which is titled "Table View Cell". The Attributes Inspector shows the following settings:

- Style:** Subtitle
- Image:** Image
- Identifier:** Reuse Identifier
- Selection:** Default
- Accesso:**  None
- Editing Ac:** Disclosure Indicator set to Detail Disclosure
- Focus Sty:** Checkmark
- Indentatio:** Detail
- Level:** 0
- Width:** 0
- Indent While Editing
- Shows Re-order Controls
- Default Insets
- Content Mode
- Content Mode:** Scale To Fill
- Semantic:** Unspecified
- Tag:** 0
- Interaction:**  User Interaction Enabled  
 Multiple Touch
- Alpha:** 1
- Background:** A color swatch.
- Tint:** Blue
- Drawing:**  Opaque  
 Hidden  
 Clears Graphics  
 Clip To Bounds  
 Autoresizing Subviews
- Stretching:** 0

A second blue callout bubble points from the right side of the screen towards the bottom right corner, containing the text: "This one's sort of special ...".

At the bottom left, the text "View as: iPhone 7 (wC hR)" is visible. At the bottom right, there is a watermark for "CS193p Winter 2017" with a small Stanford University logo.

► □ A TVCEExample > iPhone 7 TVCEExample: Ready

Table View Cell

Style Subtitle  
Image Image  
Identifier Reuse Identifier  
Selection Default  
Accessory Detail Disclosure  
Editing Acc. None  
Focus Style Default  
Indentation Level 0 Width 10  
 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   
Tint   
Drawing  Opaque  
 Hidden  
 Clears Graphics  
 Clip To Bounds  
 AutoresizeSubviews  
Stretching 0

CS193p Winter 2017

Prototype Cells

Title Subtitle

We'll talk about this Detail Disclosure button in a bit.

Table View  
Prototype Content

View as: iPhone 7 (wC hR) — 75% +

TVCEExample > iPhone 7 TVCEExample: Ready

Table View Cell

Style: Subtitle  
Image: Image  
Identifier: Reuse Identifier  
Selection: Disclosure Indicator  
Accesso: ✓ Detail Disclosure  
Editing Ac: Checkmark  
Focus Style: Detail  
Indentation: Level 0 Width 10  
 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:   
Tint: Default  
Drawing: ✓ Opaque  
 Hidden  
 Clears Graphics  
 Clip To Bounds  
 Autoresizing Subviews  
Stretching: 0

CS193p Winter 2017

Prototype Cells

Title  
Subtitle

Table View  
Prototype Content

View as: iPhone 7 (wC hR) — 75% +

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Prototype Cells

Title  
Subtitle

One of the cell styles you can choose is Custom.

Table View

Prototype Content

Custom

Basic  
Right Detail  
Left Detail  
Style ✓ Subtitle  
Image Image

Identifier Reuse Identifier

Selection Default

Accessory None

Editing Acc. None

Focus Style Default

Indentation Level 0 Width 10  
 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

+ Tint Default

Drawing  Opaque  
 Hidden  
 Clears Graphics  
 Clip To Bounds  
 AutoresizesSubviews

Stretching 0

CS193p Winter 2017

View as: iPhone 7 (wC hR) — 75% +

The screenshot shows the Xcode interface with a storyboard open. A prototype table view cell is selected, displaying a list of items with blue horizontal separators. The right side of the screen features the Attribute Inspector for the selected cell. A large blue callout bubble points from the bottom of the cell towards the text "Table View Prototype Content". Another callout bubble points from the left side of the cell towards the text "Like the cells in a static table view, Custom style cells can have UI built inside them." A third callout bubble points from the bottom right of the cell towards the text "The UI inside these cells is going to get replicated for each row (because this is a Dynamic table and thus these are ‘prototype’ cells.)".

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: Level 0, Width 10

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 Swatch]

+ Tint: [Color Swatch] Default

Drawing:  Opaque  
 Hidden  
 Clears Graphics Context  
 Clip To Bounds  
 Autoresizes Subviews

Stretching: 0 0 0 0

CS193p Winter 2017

View as: iPhone 7 (wC hR) — 75% +

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

Style: Custom

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: Level 0, Width 10

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 Swatch]

+ Tint: [Color Swatch] Default

Drawing:  Opaque  
 Hidden  
 Clears Graphics Context  
 Clip To Bounds  
 Autoresizes Subviews

Stretching: X 0, Y 0

You can change their size.

Table View  
Prototype Content

W: 375.0 H: 125.0 Prototype Cells

View as: iPhone 7 (wC hR) 75% +

CS193p Winter 2017

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

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

objects and controllers not directly available in Interface Builder.

Label

Label - A variably sized amount of static text.

Button

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

Segmented Control

Segmented Control - Displays

Filter

Winter 2017

Prototype Cells

Title

Description

Photo

Table View

Prototype Content

And you can drag UI elements into them.

It is important to set proper autolayout constraints if you want your Custom cells to adjust their height automatically to their content.

View as: iPhone 7 (wC hR) 75% +

You can't wire up outlets from your UITableViewController to these UI elements though.

Why not? Because there could be 100's or 1000's of these rows, each with a copy of this UI!

So, instead, we wire the outlets up to the UIView that is containing these UI elements.

The kind of UIView that is containing these elements is a UITableViewCell.

Just like with a Controller, we have to subclass UITableViewCell in order to have outlets in it.

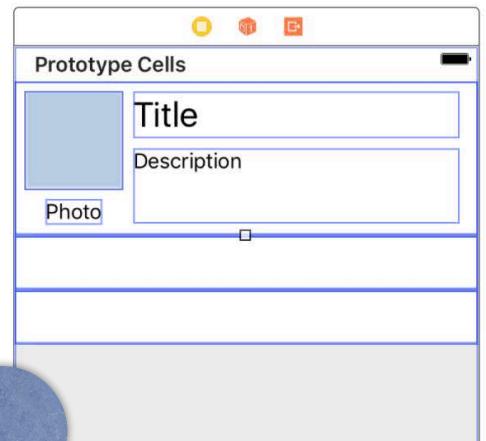


Table View  
Prototype Content

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

Content Mode: Scale To Fill

Semantic: Unspecified

Tag: 0

User Interaction Enabled

Multiple Touch

Alpha: 1

Background: [Color Swatch]

Tint: [Color Swatch] Default

Drawing:  Opaque

Hidden

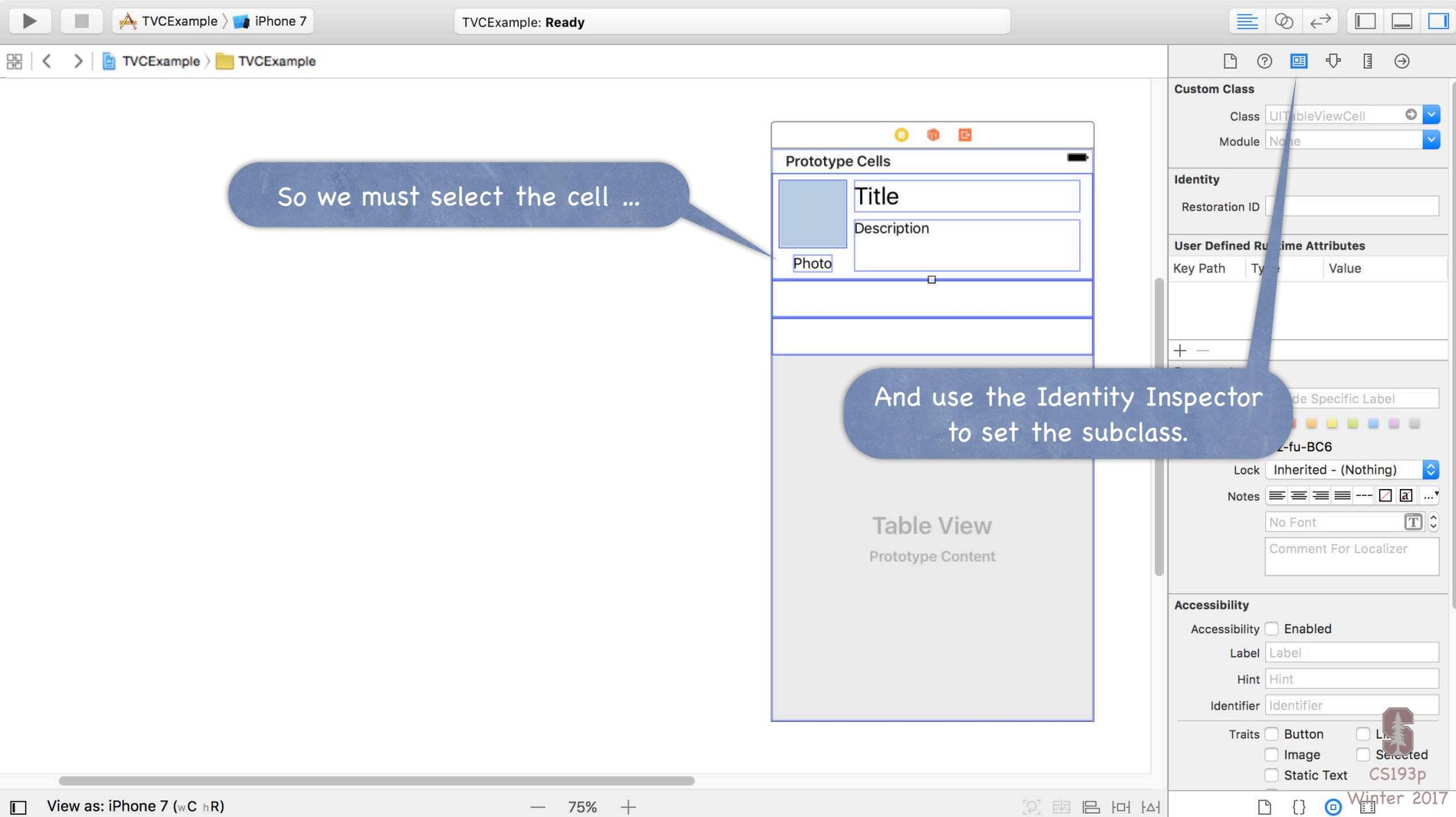
Clears Graphics Context

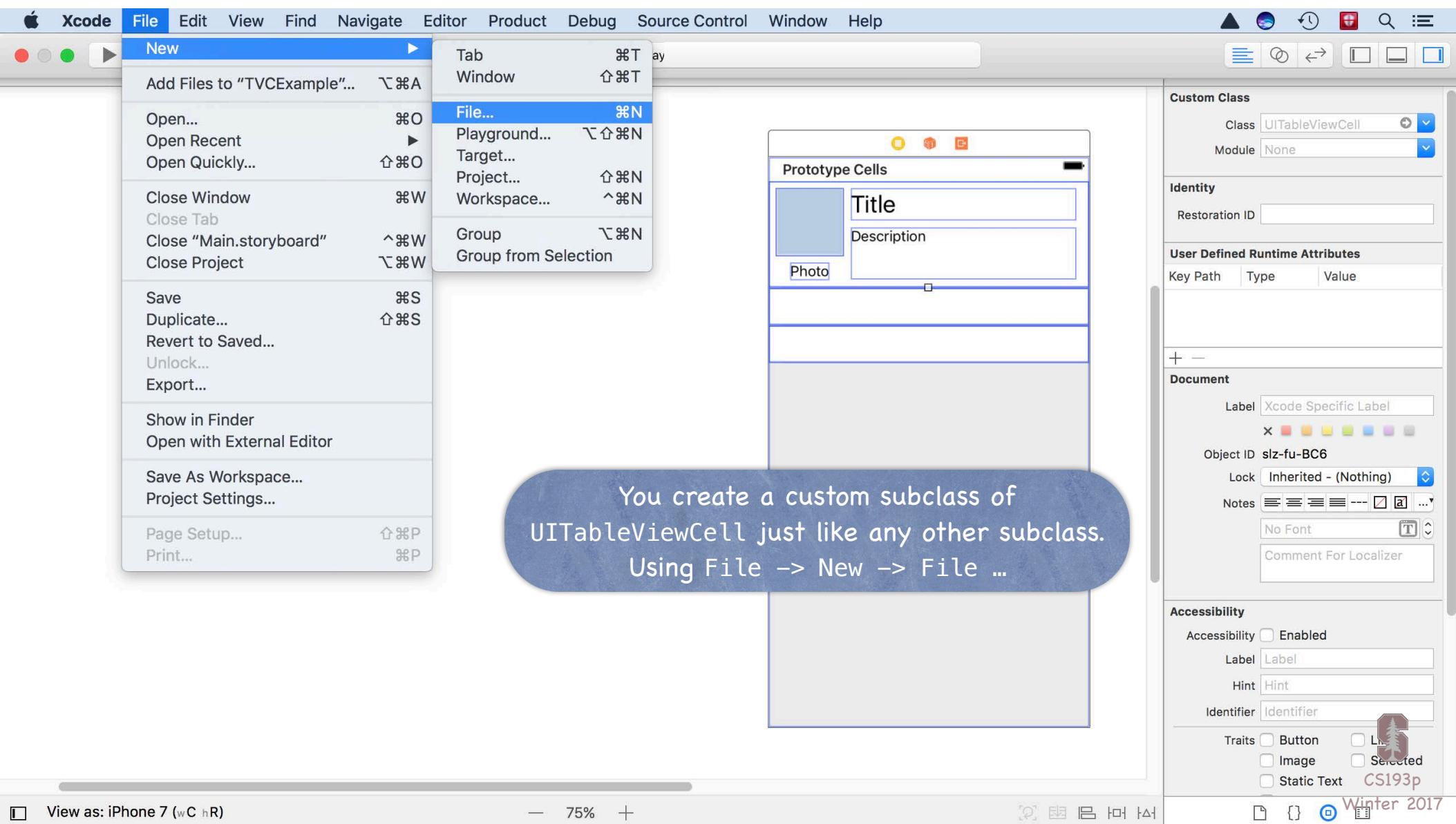
Clip To Bounds

Autoresizes Subviews

Stretching: 0 0 0 0

CS193p Winter 2017





Choose options for your new file:

Class: UITableViewcontroller

Subclass of: UITableViewcontroller  
UITableViewController  
**UITableViewCell**  
UICollectionViewcontroller  
UICollectionViewCell  
UICollectionViewReusableView

Language: Swift

Cancel Previous Next

Custom Class

Class: UITableViewCell

Module: None

Identity

Restoration ID:

User Defined Runtime Attributes

Key Path Type Value

Document

Label: Xcode Specific Label

Object ID: slz-fu-BC6

Lock: Inherited - (Nothing)

Notes: No Font

Comment For Localizer

Accessibility

Enabled:

Label: Label

Hint: Hint

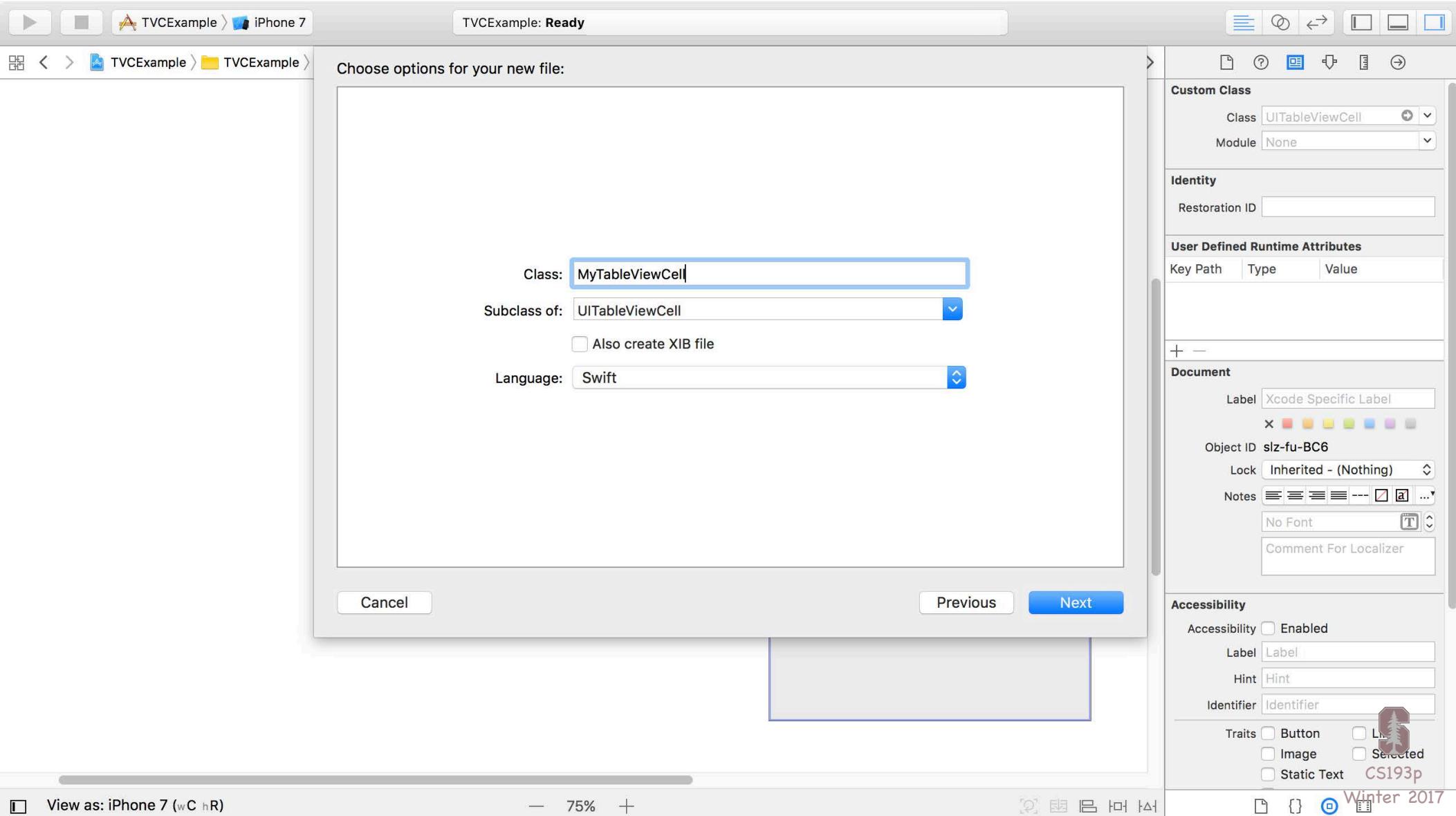
Identifier: Identifier

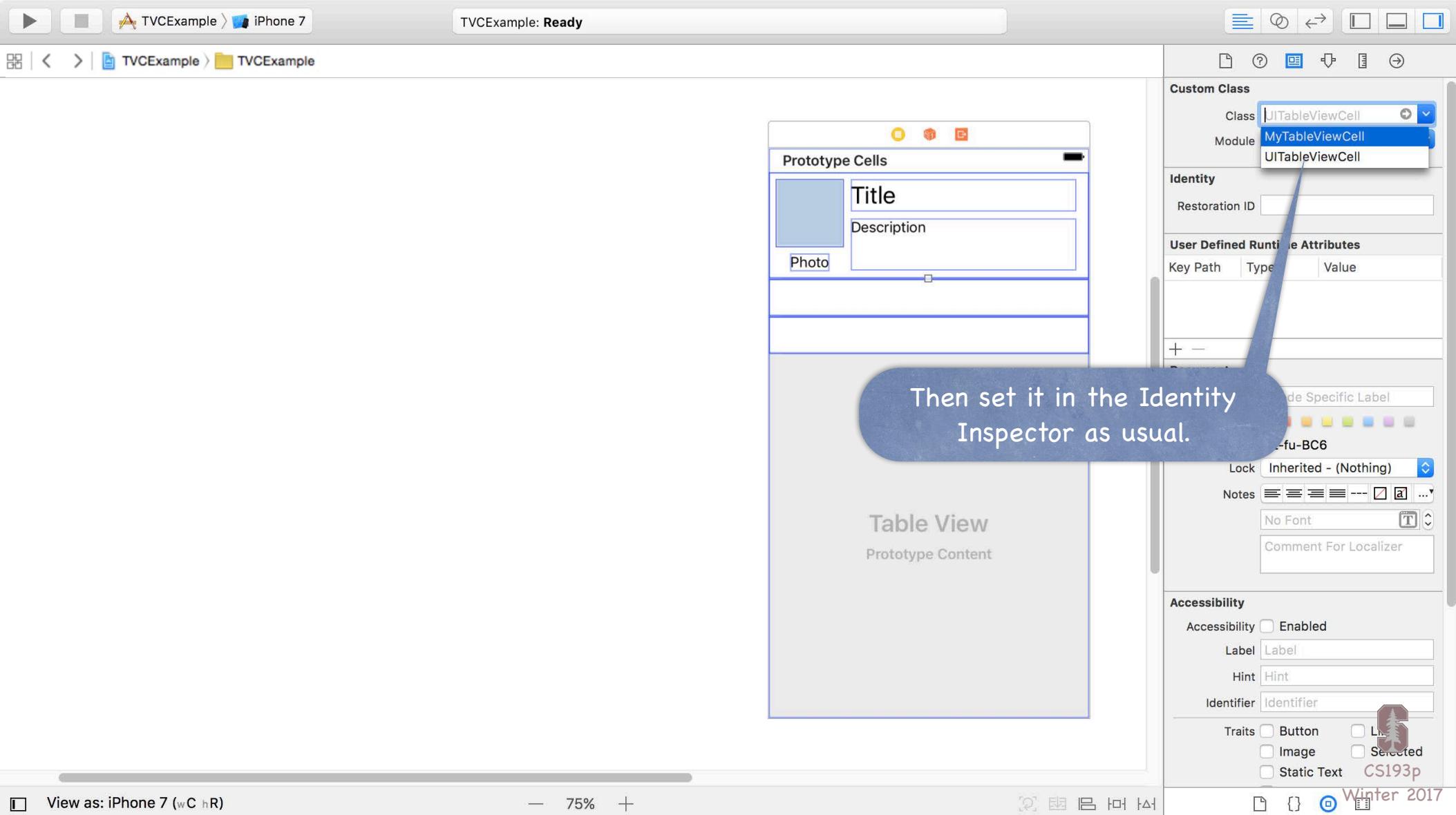
Traits:  Button  Link  
 Image  Selected  
 Static Text  CS193p

View as: iPhone 7 (wC hR) 75% +

Winter 2017

Choose UITableViewCell as the class to subclass off of.

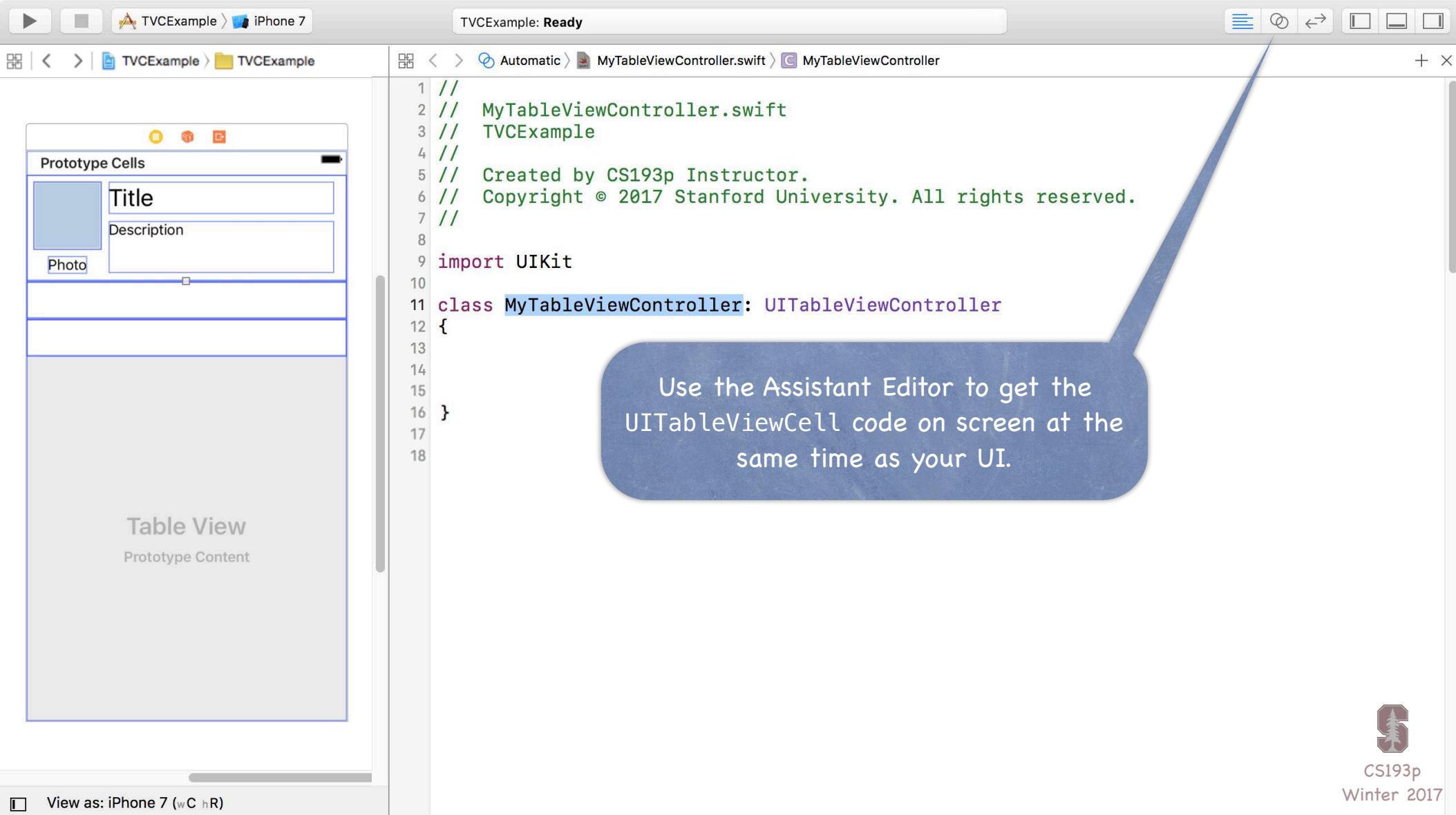




Now you can wire up outlets and actions to the UI elements.

Static cell UI elements outlets are wired to the UITableViewController.  
Dynamic cell UI elements outlets are wired to the UITableViewCell containing them.

The screenshot shows the Xcode interface with a storyboard file open. The storyboard is titled "TVCExample: Ready" and is part of the project "TVCExample". The storyboard contains a single prototype cell. This cell has three main components: a photo placeholder labeled "Photo", a title label labeled "Title", and a description label labeled "Description". A blue callout bubble originates from the "Photo" label and points towards the "Photo" placeholder in the cell's layout. The right side of the Xcode window displays the "Identity Inspector" panel, which shows the custom class is set to "MyTableViewCell". Other tabs in the inspector include "Custom Class", "User Defined Runtime Attributes", "Document", and "Accessibility". The "Document" tab shows the object ID as "slz-fu-BC6", a lock status of "Inherited - (Nothing)", and a note about "Font" and "Text Content For Localizer". The bottom of the screen shows the Xcode toolbar with various icons and the text "View as: iPhone 7 (wC hR)" and "75%".

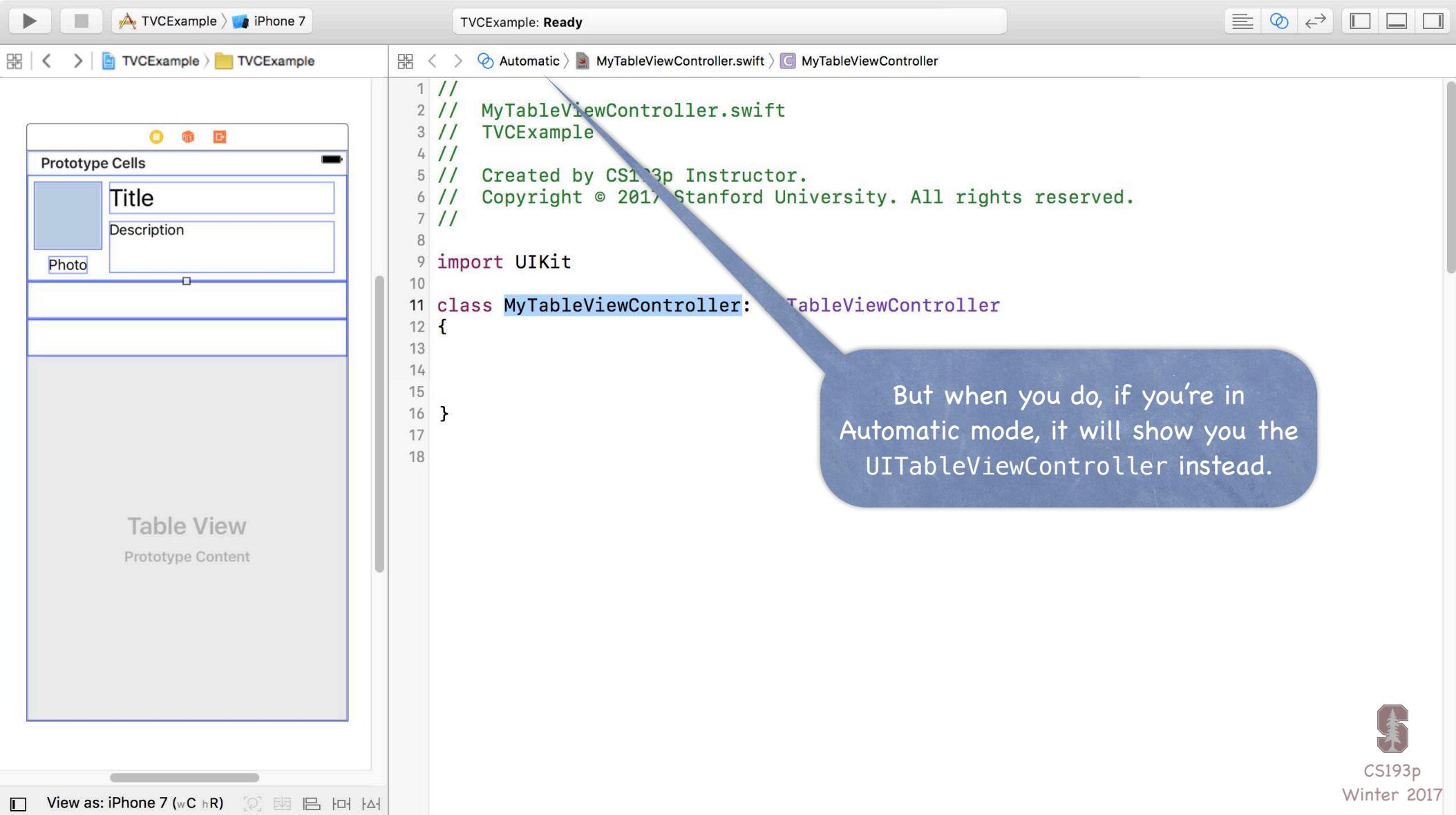


A screenshot of the Xcode interface demonstrating the Assistant Editor feature. On the left, the Storyboard editor shows a table view with a prototype cell containing a photo placeholder, a title label, and a description label. On the right, the code editor shows the Swift file `MyTableViewController.swift` with the following content:

```
1 // MyTableViewController.swift
2 // TVCExample
3 /**
4 * Created by CS193p Instructor.
5 * Copyright © 2017 Stanford University. All rights reserved.
6 */
7
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15 }
16
17
18 }
```

A blue callout bubble points from the text "UITableViewController code on screen at the same time as your UI." to the code editor area.





The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard titled "TVCEExample" for an "iPhone 7" device. It contains a table view with a prototype cell having three sections: "Photo", "Title", and "Description". The right pane shows the "TVCEExample: Ready" workspace with the file "MyTableViewController.swift" open. The code is as follows:

```
// MyTableViewController.swift
// TVCEExample
//
// Created by CS193p Instructor.
// Copyright © 2017 Stanford University. All rights reserved.

import UIKit

class MyTableViewController: UITableViewController
```

A blue callout bubble originates from the word "UITableViewController" in the code and points towards the storyboard preview on the left.

**But when you do, if you're in Automatic mode, it will show you the UITableViewController instead.**



A screenshot of the Xcode IDE. On the left, the Storyboard editor shows a prototype cell with three sections: 'Photo', 'Title', and 'Description'. Below it is a large 'Table View' placeholder. On the right, the Swift code editor displays the following code:

```
1 // Manual
2 // Automatic (1)
3 // Top Level Objects (1)
4 // Localizations
5 // Notification Payloads
6 // Preview (1)

7
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15
16 }
```

The 'Automatic (1)' item in the file inspector is highlighted with a blue arrow pointing from a callout bubble. The callout bubble contains the text: "So mouse down on Automatic ...".



The screenshot shows the Xcode interface with two main panes. On the left is the Storyboard editor, displaying a prototype table view cell with sections for 'Photo', 'Title', and 'Description'. Below it is a preview area labeled 'Table View' and 'Prototype Content'. On the right is the Swift code editor, showing the following code:

```
1 // Manual
2
3
4
5
6
7
8
9 import UIKit
10
11 class MyTableViewController: UITableViewController
12 {
13
14
15
16
17
18 }
```

A blue callout bubble originates from the code editor and points towards the storyboard, containing the text "... switch to Manual ...".



CS193p

Winter 2017

The screenshot shows the Xcode interface with a storyboard and a Swift code editor side-by-side.

**Storyboard View:** On the left, the storyboard editor displays a table view with a prototype cell containing three sections: "Title", "Description", and "Photo".

**File Navigator:** In the center, the file navigator shows the project structure under "TVCEExample":

- Manual
- Automatic (1)
- Top Level Objects (1)
- Localizations
- Notification Payloads
- Preview (1)

**Code Editor View:** On the right, the code editor shows the `MyTableViewCell.swift` file:1 import UIKit  
2  
3 class MyTableViewCell: UITableViewCell  
4 {  
5  
6 }  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18

A callout bubble originates from the storyboard preview area and points to the line `class MyTableViewCell: UITableViewCell` in the code editor, suggesting to choose `UITableViewCell` instead of `MyTableViewCell`.

**Callout Text:** ... and choose your UITableViewCells subclass instead.



TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Prototype Cells

Title

Description

Photo

Table View

Prototype Content

```
1 // MyTableViewCell.swift
2 // TVCEexample
3 //
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewCell: UITableViewCell
12 {
13
14
15 }
16
17
18 }
```

Now you're ready to ctrl-drag!



The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard titled 'TVCEExample' for an 'iPhone 7' device. It contains a 'Table View' with a single prototype cell. This cell has three components: a 'Photo' image view at the bottom, a 'Title' label above it, and a 'Description' label above that. A blue line with arrows indicates a connection being drawn from the 'Photo' view in the storyboard to the code in the right pane. The right pane shows the 'MyTableViewCell.swift' file under the 'TVCEExample' project. The code is as follows:

```
1 // MyTableViewCell.swift
2 // TVCEExample
3 //
4 // Created by CS193p Instructor.
5 // Copyright © 2017 Stanford University. All rights reserved.
6 //
7
8 import UIKit
9
10 class MyTableViewCell: UITableViewCell
11 {
12 }
13
14 
```

A callout bubble with the text 'Insert Outlet or Outlet Collection' points to the line '14' in the code, where the cursor is currently located.



The screenshot shows the Xcode interface with the storyboard and code editor side-by-side.

**Storyboard:** On the left, the storyboard displays a prototype cell with three components: "Photo" (a placeholder image), "Title" (a text label), and "Description" (a text label). A connection inspector dialog is open, showing the following settings:

- Connection: Outlet
- Object: My Table View Cell
- Name: photolImageView
- Type: UIImageView
- Storage: Weak

**Code Editor:** On the right, the code editor displays the `MyTableViewCell.swift` file:

```
1 // MyTableViewCell.swift
2 // TVCExample
3 //
4 // Created by CS193p Instructor.
5 // Copyright © 2017 Stanford University. All rights reserved.
6 //
7
8 import UIKit
9
10 class MyTableViewCell: UITableViewCell
11 {
12 }
```

**Bottom Bar:** At the bottom, the interface includes a "View as: iPhone 7 (wC hR)" dropdown and several small icons for navigating between different views.



The screenshot shows the Xcode interface with two main panes. The left pane displays a storyboard for an iPhone 7 device, featuring a table view with a prototype cell containing a title label, a description label, and a photo image view. The right pane shows the corresponding Swift code for the `MyTableViewCell` class.

```
1 // MyTableViewCell.swift
2 // TVCExample
3 //
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewCell: UITableViewCell
12 {
13     @IBOutlet weak var photoImageView: UIImageView!
14
15 }
16
17
18 }
```

A blue callout bubble points from the `photoImageView` outlet declaration in the code back to the `Photo Image View` in the storyboard's prototype cell.

This outlet is not in the Controller!  
It's in your UITableViewCell subclass.  
Every row in the table will have its own  
photoImageView.



A UITableCell subclass has to have some public API that gives it the information it needs to load up its outlet views.

We'll see where you set this var in code in a moment.

```
1 // MyTableViewCell.swift
2 // TVCEExample
3 /**
4 * Created by CS193p Instructor.
5 * Copyright © 2017 Stanford University. All rights reserved.
6 */
7
8 import UIKit
9
10 class MyTableViewCell: UITableViewCell {
11     @IBOutlet weak var photoImageView: UIImageView!
12
13     var infoShownByThisCell: Type { didSet { updateUI() } }
14
15 }
```



# UITableView Protocols

## ⌚ How to connect all this stuff up in code?

Connections to code are made using the UITableView's **dataSource** and **delegate**

The **delegate** is used to control how the table is displayed (it's look and feel)

The **dataSource** provides the data that is displayed inside the cells

UITableViewController automatically sets itself as the UITableView's delegate & dataSource

Your UITableViewController subclass will also have a property pointing to the UITableView ...

```
var tableView: UITableView // self.view in UITableViewController
```

## ⌚ When do we need to implement the dataSource?

Whenever the data in the table is dynamic (i.e. not static cells)

There are three important methods in this protocol ...

How many sections in the table?

How many rows in each section?

Give me a view to use to draw each cell at a given row in a given section.

Let's cover the last one first (since the first two are very straightforward) ...



# Customizing Each Row

## • Providing a UIView to draw each row ...

It has to be a **UITableViewCell** (which is a subclass of **UIView**) or subclass thereof

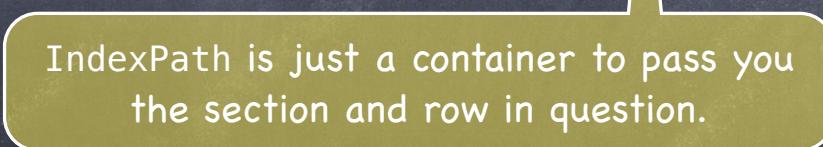
Don't worry, if you have 10,000 rows, only the visible ones will have a **UITableViewCell**

But this means that **UITableViewCells** are **reused** as rows appear and disappear

This has ramifications for **multithreaded** situations, so be careful in that scenario

The **UITableView** will ask its **UITableViewDataSource** for the **UITableViewCell** for a row ...

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



IndexPath is just a container to pass you  
the section and row in question.

```
}
```



# Customizing Each Row

- Providing a UIView to draw each row ...

It has to be a **UITableViewCell** (which is a subclass of **UIView**) or subclass thereof

Don't worry, if you have 10,000 rows, only the visible ones will have a **UITableViewCell**

But this means that **UITableViewCells** are **reused** as rows appear and disappear

This has ramifications for **multithreaded** situations, so be careful in that scenario

The **UITableView** will ask its **UITableViewDataSource** for the **UITableViewCell** for a row ...

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    // myInternalDataStructure is conceptual here: it doesn't have to be an Array of Arrays
}
```



# Customizing Each Row

## • Providing a UIView to draw each row ...

It has to be a `UITableViewCell` (which is a subclass of `UIView`) or subclass thereof

Don't worry, if you have 10,000 rows, only the visible ones will have a `UITableViewCell`

But this means that `UITableViewCell`s are **reused** as rows appear and disappear

This has ramifications for **multithreaded** situations, so be careful in that scenario

The `UITableView` will ask its `UITableViewDataSource` for the `UITableViewCell` for a row ...

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]

    let cell = . . . // create a UITableViewCell and load it up with data

    return cell
}
```



CS193p

Winter 2017

# Customizing Each Row

## • Providing a UIView to draw each row ...

It has to be a `UITableViewCell` (which is a subclass of `UIView`) or subclass thereof

Don't worry, if you have 10,000 rows, only the visible ones will have a `UITableViewCell`

But this means that `UITableViewCell`s are **reused** as rows appear and disappear

This has ramifications for **multithreaded** situations, so be careful in that scenario

The `UITableView` will ask its `UITableViewDataSource` for the `UITableViewCell` for a row ...

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]

    let cell = . . . // create a UITableViewCell and load it up with data

    return cell
}
```



CS193p

Winter 2017

TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

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

func tableView(\_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
 let data = myInternalDataStructure[indexPath.section][indexPath.row]  
}

View as: iPhone 7 (wC hR) 75% CS193p Winter 2017

TVCEExample > iPhone 7 TVCEExample: Ready

TVCEExample > TVCEExample

Table View Cell

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

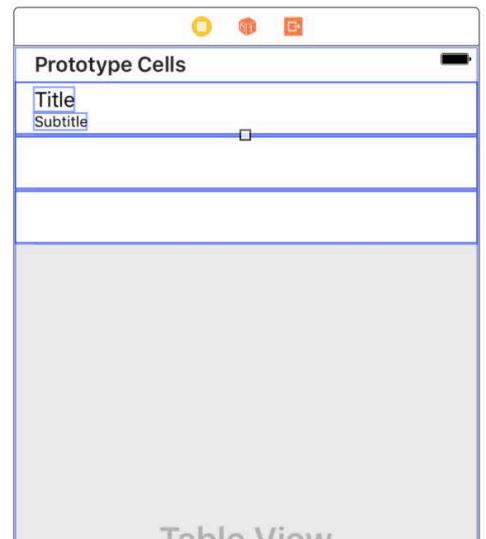
View

Content Mode Scale To Fill  
Semantic Unspecified  
Tag 0  
 User Interaction Enabled  
 Multiple Touch  
Alpha 1  
Background  
Tint Default  
Drawing  Opaque  
 Hidden  
 Clears Graphics  
 Clip To Bounds  
 AutoresizesSubviews  
Stretching 0

CS193p Winter 2017

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
}
```

View as: iPhone 7 (wC hR) — 75% +



This method gets a UITableViewCell for us either by reusing one that has gone off screen or by making a copy of one of our prototypes in the storyboard.

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCell", for: indexPath)
}
```

The storyboard shows a table view with a prototype cell. The prototype cell has two sections: 'Title' and 'Subtitle'. The right side of the interface shows the 'Table View Cell' settings for the selected prototype cell. The 'Identifier' is set to 'MyCell'. Other settings include 'Style: Subtitle', 'Image: Image', 'Selection: Default', 'Accessory: None', 'Editing Acc.: None', 'Focus Style: Default', 'Indentation: 0 Level, 10 Width', 'Shows Re-order Controls: checked', 'Separator: Default Insets', 'Content Mode: Scale To Fill', 'Semantic: Unspecified', 'Tag: 0', 'User Interaction Enabled: checked', 'Multiple Touch: unchecked', 'Alpha: 1', 'Background: red gradient', 'Tint: blue', 'Drawing: Opaque (checked)', 'Hidden: unchecked', 'Clears Graphics: Next (unchecked)', 'Clip To Bounds: checked', 'AutoresizeSubviews: checked', and 'Stretching: 0'. The bottom right corner of the storyboard area has a watermark 'CS193p Winter 2017'.

This method gets a UITableViewCell for us either by reusing one that has gone off screen or by making a copy of one of our prototypes in the storyboard.

This String tells iOS which prototype to copy or reuse.

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCell", for: indexPath)
}
```

The storyboard shows a table view with a prototype cell labeled "MyCell". The identifier field in the Attributes Inspector is set to "MyCell".

The code editor shows the implementation of the `tableView(_:cellForRowAt:)` method. It dequeues a reusable cell from the table view using the identifier "MyCell".

The screenshot shows the Xcode interface with a storyboard file open. On the left, a preview window shows an iPhone 7 device with the title "TVCEExample: Ready". Below it, the storyboard canvas displays a table view with a prototype cell containing two labels: "Title" and "Subtitle". A callout bubble from the storyboard points to the right-hand sidebar, which is titled "Table View Cell". The "Identifier" field in the sidebar is set to "MyCell". A second callout bubble originates from the storyboard's table view and points to the associated Swift code in the bottom-left corner.

For a non-Custom cell ...

... the dequeued thing will be a generic UITableViewCell. You can look up its API to see what sort of configuration options are available for it.

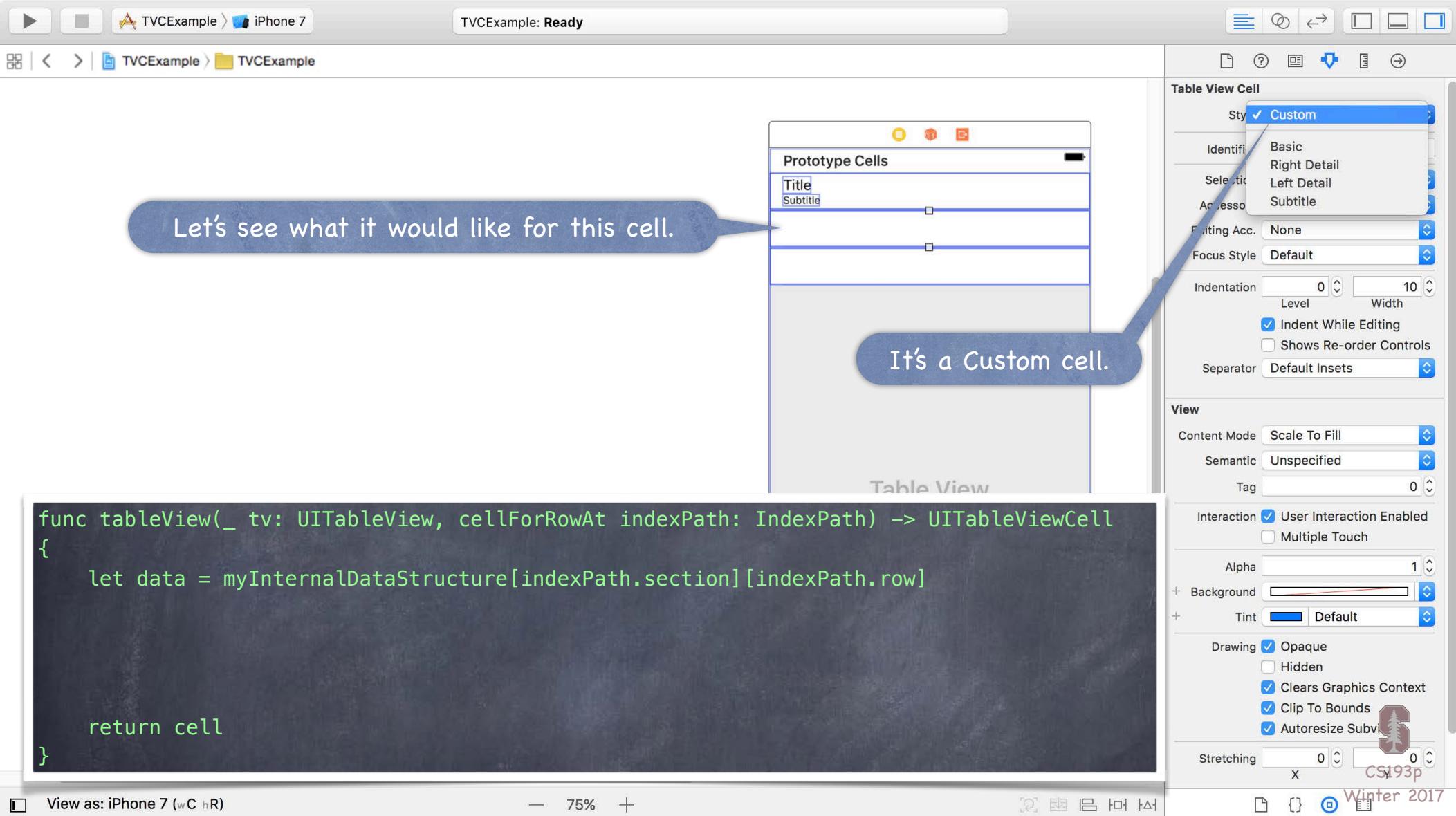
```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCell", for: indexPath)

    dequeued.textLabel?.text = data.importantInfo
    dequeued.detailTextLabel?.text = data.lessImportantInfo
    return cell
}
```

Let's see what it would like for this cell.

It's a Custom cell.

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let cell = UITableViewCell(style: .custom, reuseIdentifier: "Cell")
    cell.textLabel?.text = data.name
    cell.detailTextLabel?.text = data.address
    return cell
}
```



Let's see what it would like for this cell.

The screenshot shows the Xcode interface with a storyboard, a preview, and a code editor. In the storyboard, a table view contains two prototype cells. The first cell has a title and subtitle. The second cell is selected and highlighted with a blue border. A callout bubble points from the text "Let's see what it would like for this cell." towards the second cell. On the right, the "Table View Cell" inspector is open, showing settings for the selected cell, including Identifier "MyCustomCell". Below the storyboard, the code for the UITableViewDataSource protocol is visible:

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCustomCell", for: indexPath)

    return cell
}
```

The code uses the identifier "MyCustomCell" to dequeue the cell. The storyboard preview shows the table view with two cells, and the second cell is highlighted with a blue border. The Xcode status bar at the bottom indicates "View as: iPhone 7 (wC hR)" and "75%".



CS193p

Winter 2017

TVCEExample > iPhone 7 TVCEExample: Ready

Identity Inspector

Let's see what it would like for this cell.

... the dequeued thing will be your subclass of UITableViewCell.  
You will use its public API to configure it  
(i.e. that public API will set the values of its outlets, etc.).

The screenshot shows the Xcode interface with a storyboard file open. A UITableView is selected, and its prototype cell is visible in the preview area. The Identity Inspector on the right side of the screen is open, showing the following details for the selected cell:

- Custom Class:** MyTableViewCell
- Module:** Current – TVCEExam...
- Identity:** Restoration ID: (empty)
- User Defined Runtime Attributes:** Key Path, Type, Value: (empty)
- Document:** Label: Xcode Specific Label, Object ID: 8no-Rw-blt, Lock: Inherited - (Nothing), Notes: No Font, Comment For Localizer: (empty)
- Accessibility:** Enabled: (unchecked), Label: Label, Hint: Hint, Identifier: Identifier, Traits: (unchecked) Button, (unchecked) Image, (unchecked) Static Text, (unchecked) L, (unchecked) Selected, (unchecked) Selected Text

Below the storyboard, a portion of the code for the UITableViewDataSource implementation is shown:

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let data = myInternalDataStructure[indexPath.section][indexPath.row]  
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCustomCell", for: indexPath)  
    if let cell = dequeued as? MyTableViewCell {  
        cell.infoShownByThisCell = data.theDataTheCellNeedsToDisplayItsCustomLabelsEtc  
    }  
    return cell  
}
```

At the bottom of the interface, there are buttons for View as: iPhone 7 (wC hR), zoom controls (75%), and a footer bar with icons for search, file, and other Xcode functions.

TVCEExample > iPhone 7

TVCEExample: Ready

TVCEExample > TVCEExample

Prototype Cells

Title

Description

Photo Image View

```
1 // MyTableViewCell.swift
2 // TVCEexample
3 //
4 //
5 // Created by CS193p Instructor.
6 // Copyright © 2017 Stanford University. All rights reserved.
7 //
8
9 import UIKit
10
11 class MyTableViewCell: UITableViewCell
12 {
13     @IBOutlet weak var photoImageView: UIImageView!
14
15     var infoShownByThisCell: Type { didSet { updateUI() } }
16 }
17
18
```

```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: "MyCustomCell", for: indexPath)
    if let cell = dequeued as? MyTableViewCell {
        cell.infoShownByThisCell = data.theDataTheCellNeedsToDisplayItsCustomLabelEtc
    }
    return cell
}
```

View as: iPhone 7 (wC hR)



CS193p

Winter 2017

# UITableViewDataSource

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

And how many sections, too, of course?

Via these UITableViewDataSource protocol methods ...

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

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

## • Number of sections is 1 by default

In other words, if you don't implement numberOfSectionsInTableView, it will be 1

## • No default for numberOfRowsInSection

This is a required method in this protocol (as is cellForRowAt)

## • What about a static table?

Do not implement these dataSource methods for a static table

UITableViewController will take care of that for you

You edit the data directly in the storyboard



# UITableViewDataSource

## ⌚ Summary

Loading your table view with data is simple ...

1. set the table view's dataSource to your Controller (automatic with UITableViewController)
2. implement `numberOfSections` and `numberOfRowsInSection`
3. implement `cellForRowAtIndex` to return loaded-up UITableViewCells

## ⌚ Section titles are also considered part of the table's "data"

So you return this information via UITableViewDataSource methods ...

```
func tableView(UITableView, titleForHeaderInSection: Int) -> String
```

If a String is not sufficient, the UITableView's delegate can provide a UIView

## ⌚ There are a number of other methods in this protocol

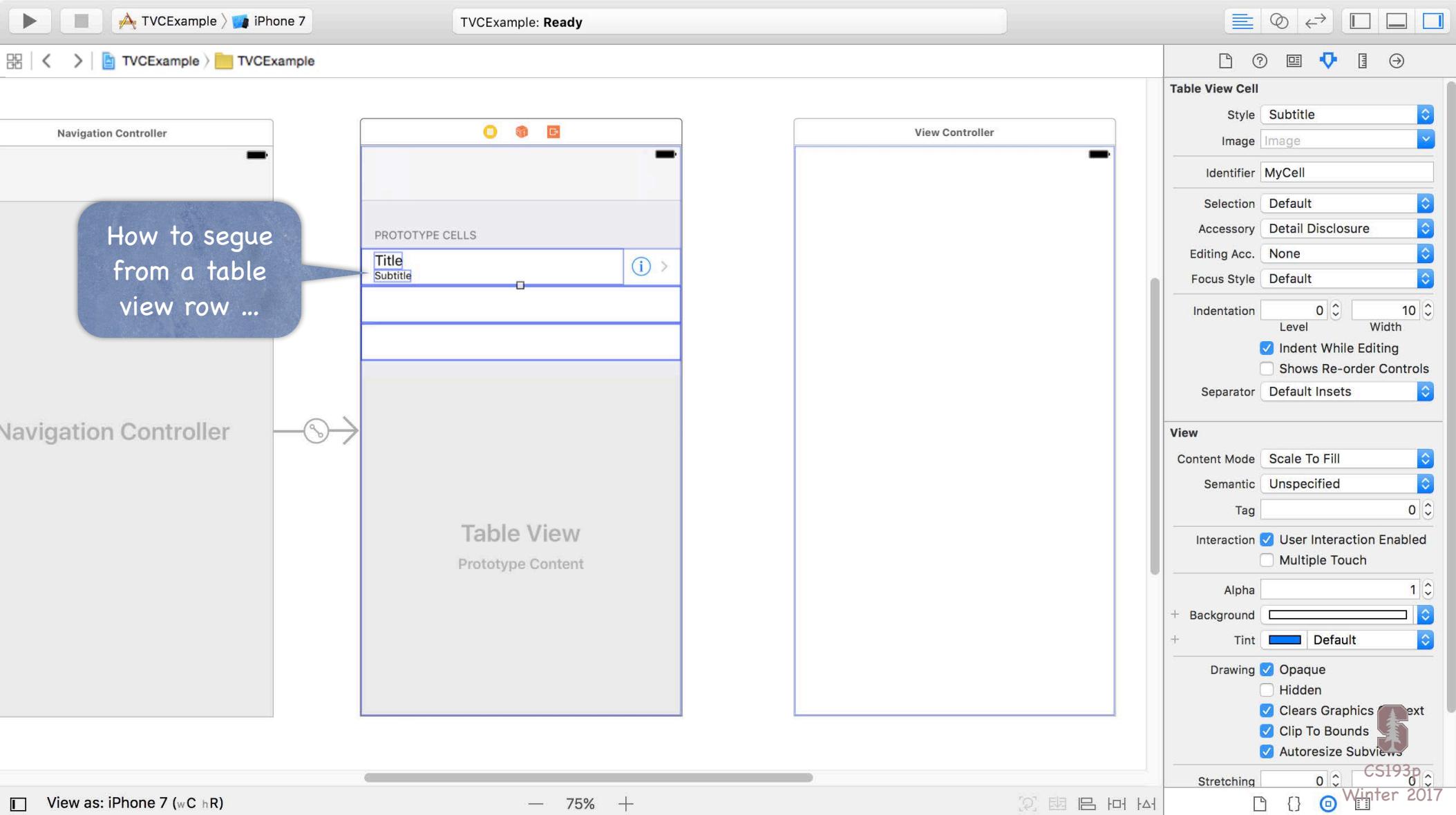
But we're not going to cover them in lecture

They are mostly about dealing with editing the table by deleting/moving/inserting rows

That's because when rows are deleted, inserted or moved, it would likely modify the Model

(and we're talking about the UITableViewDataSource protocol here)





TVCEExample > iPhone 7

TVCEExample: Ready

Navigation Controller

Table View  
Prototype Content

View Controller

PROTOTYPE CELLS

Title Subtitle

Note that this row has a Detail Disclosure Accessory.

We can segue from the row and/or from the Detail Disclosure Accessory.

Table View Cell

Style Subtitle

Image Image

Identifier MyCell

Accessory  Detail Disclosure

Editing Accessory Checkmark

Focus Style Default

Indentation Level 0 Width 10

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

Drawing  Opaque

Hidden

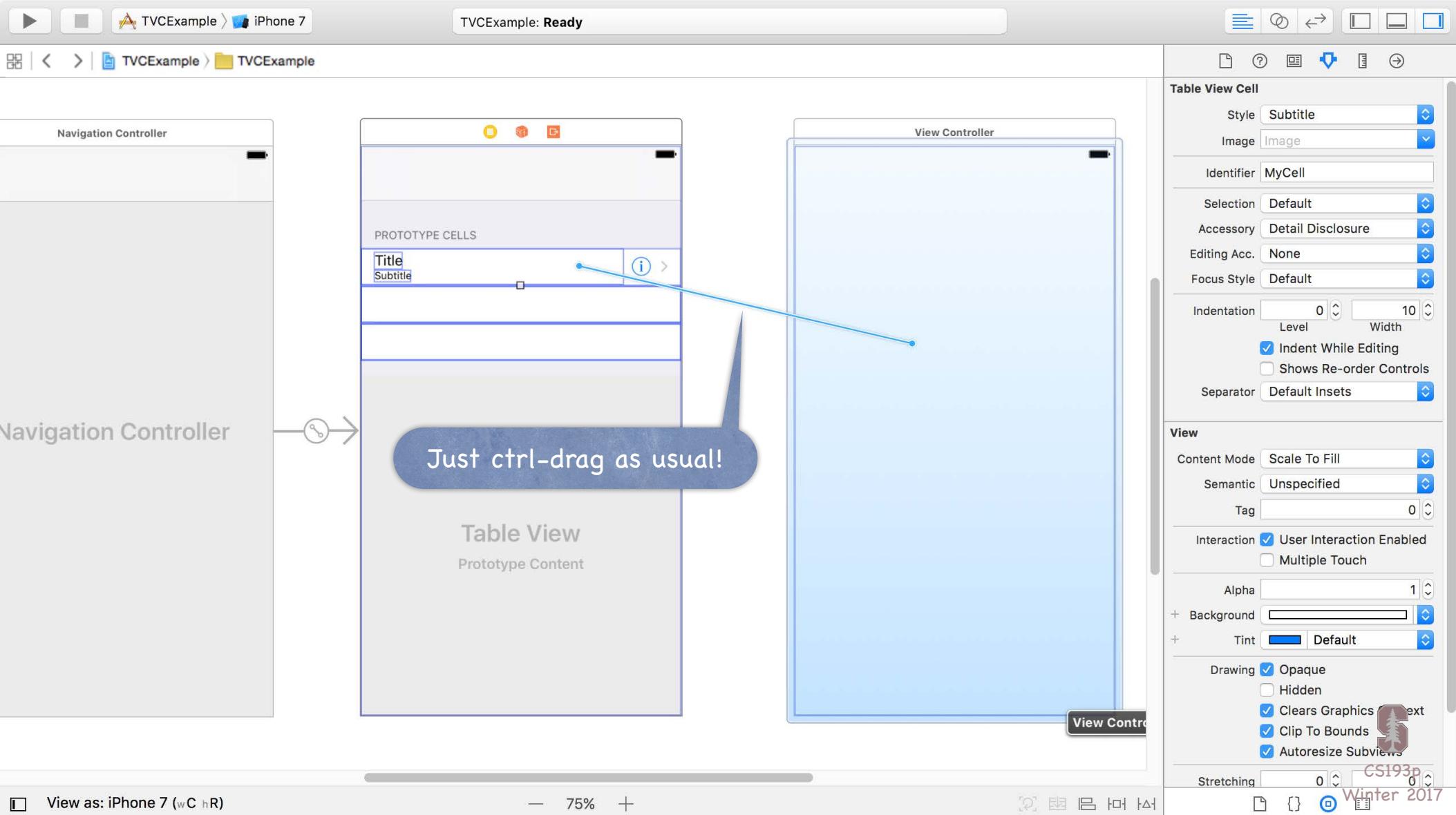
Clears Graphics

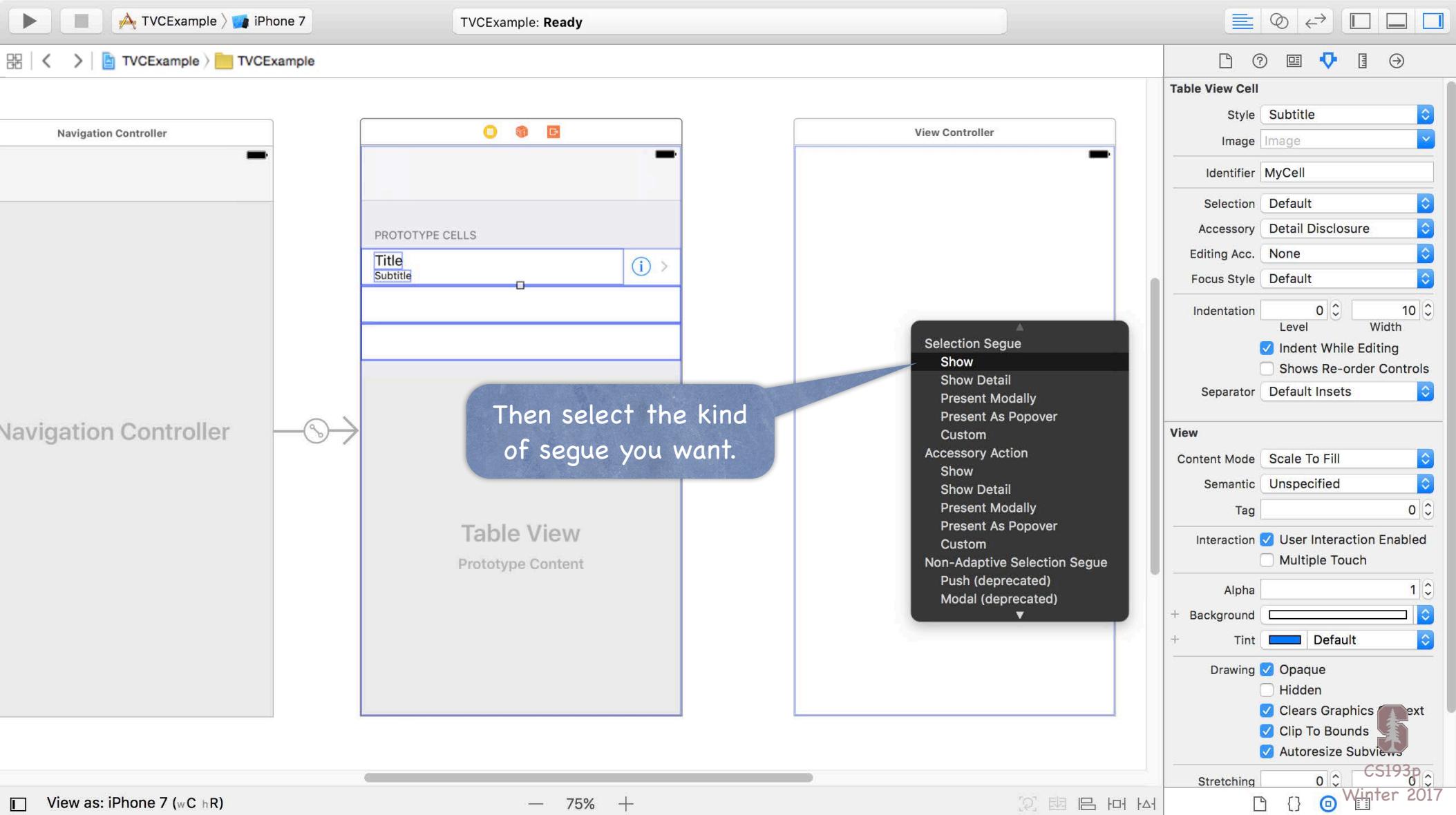
Clip To Bounds

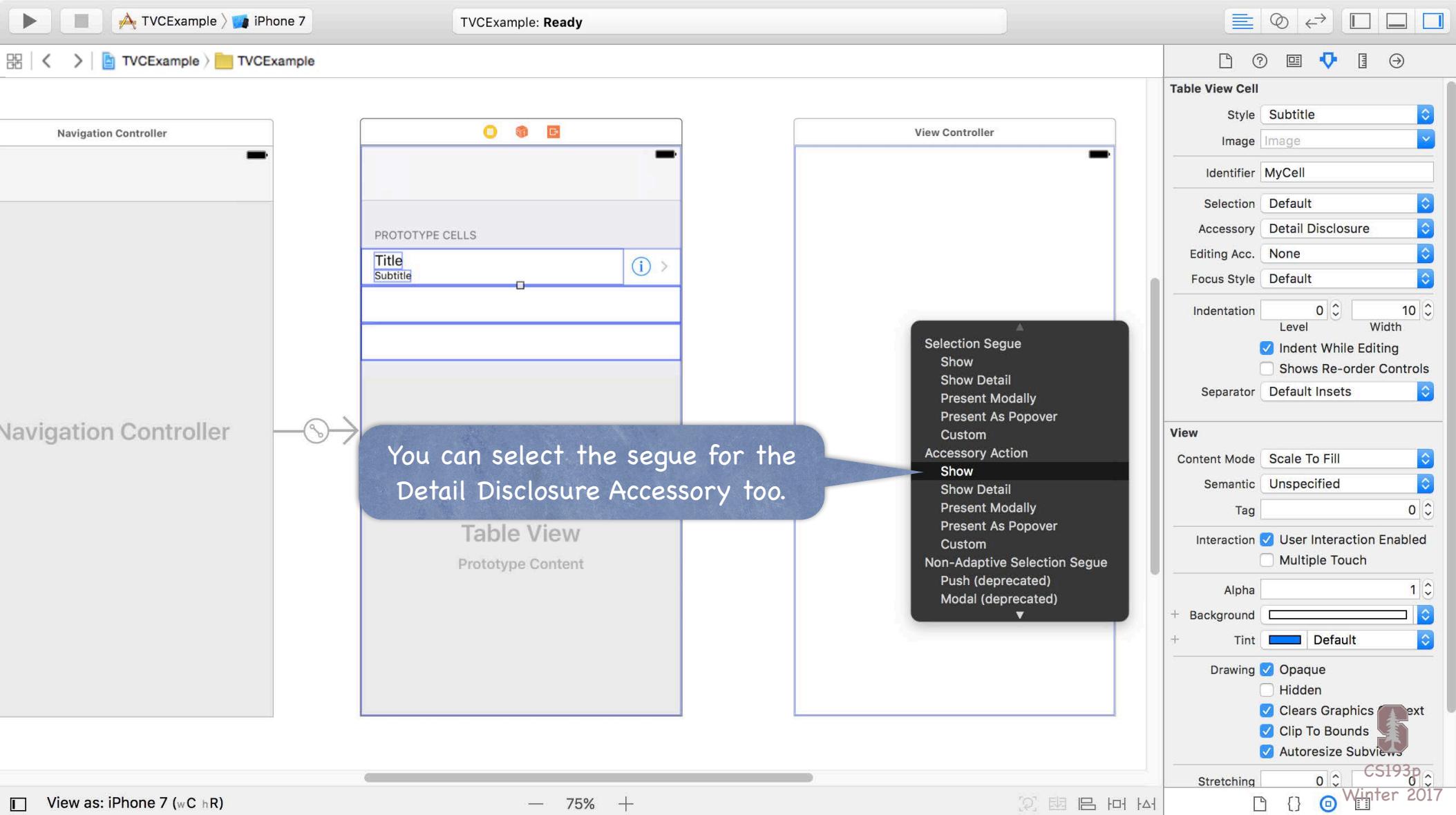
AutoresizesSubviews

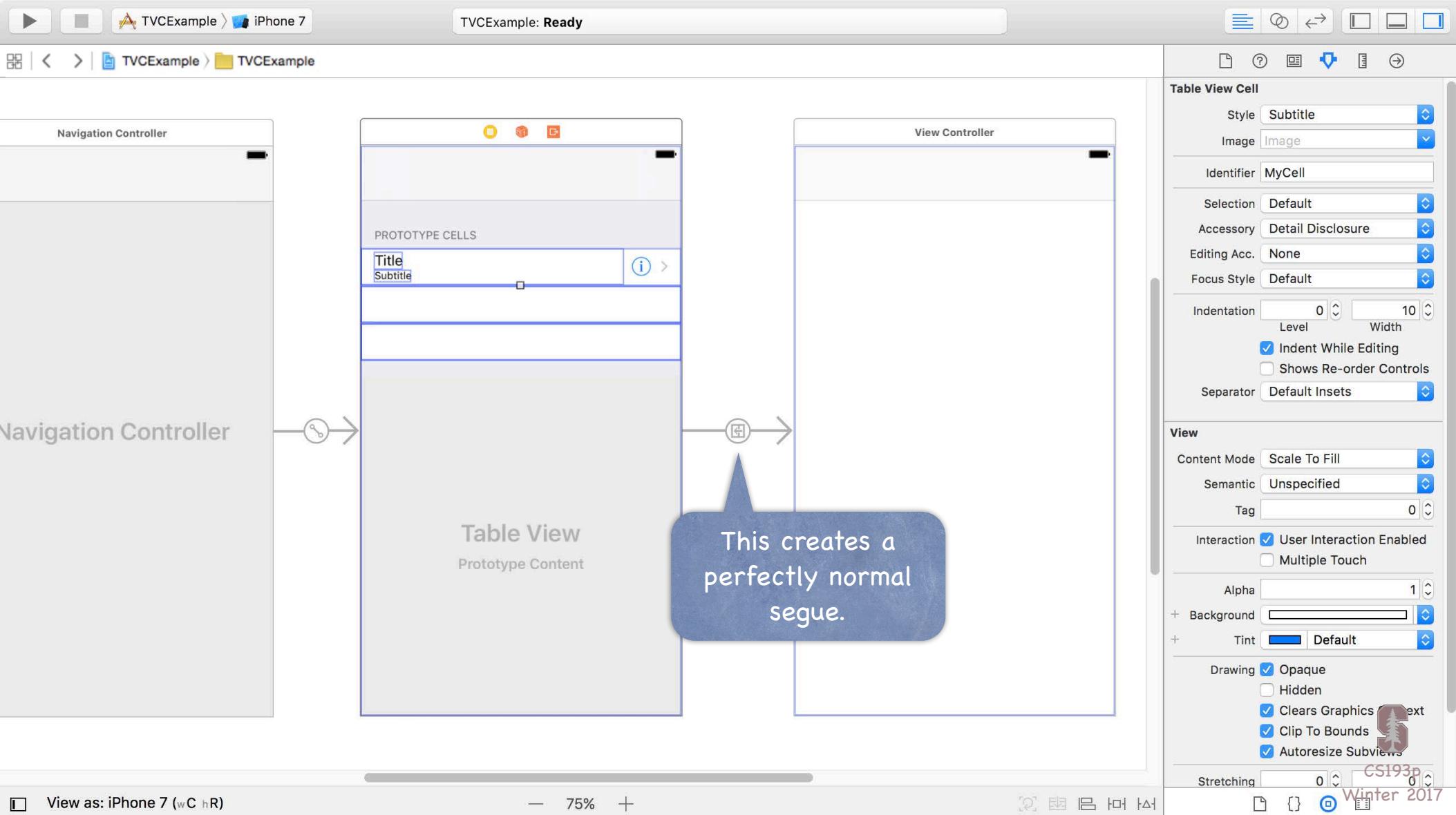
Stretching 0

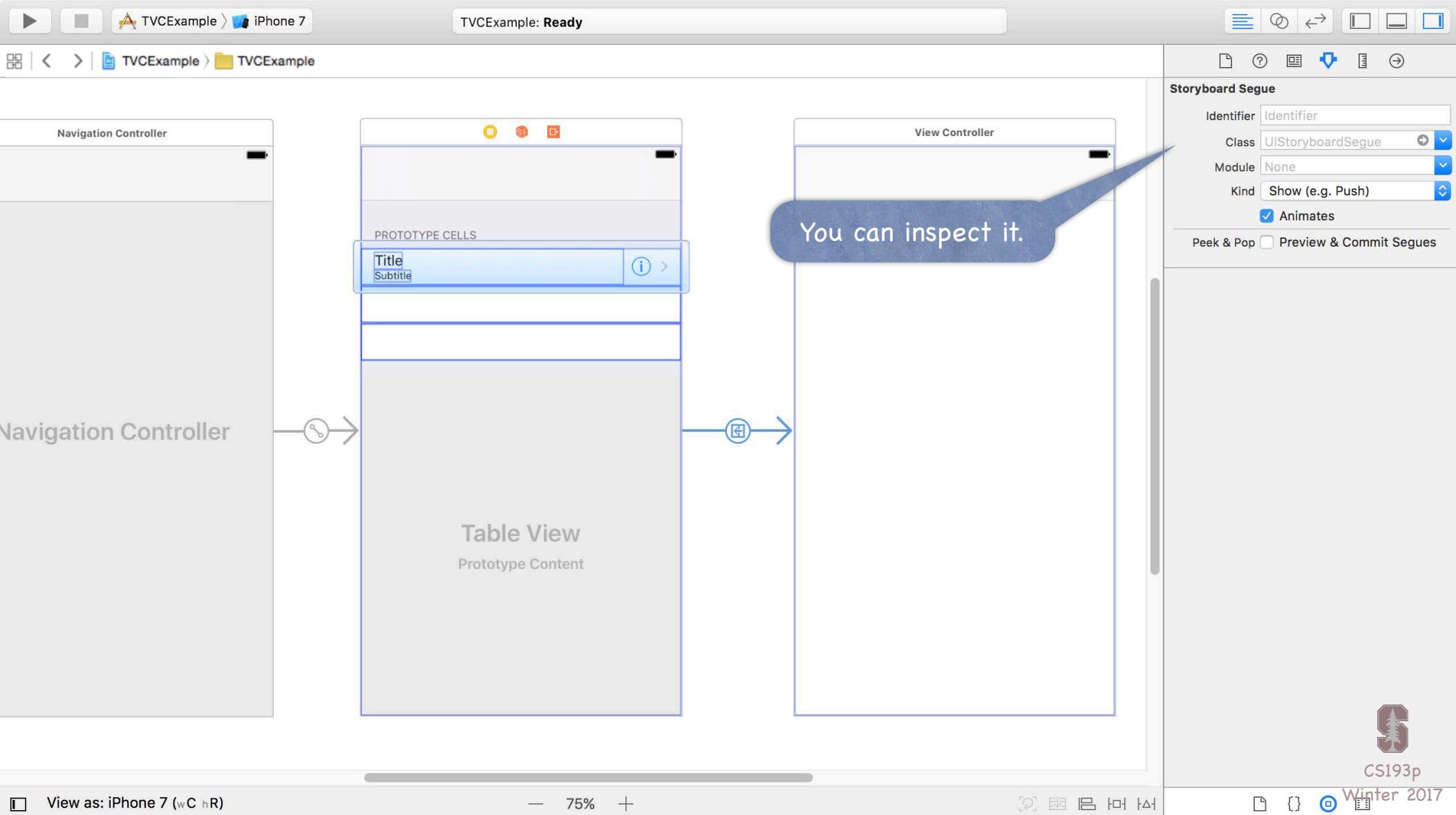
CS193p Winter 2017

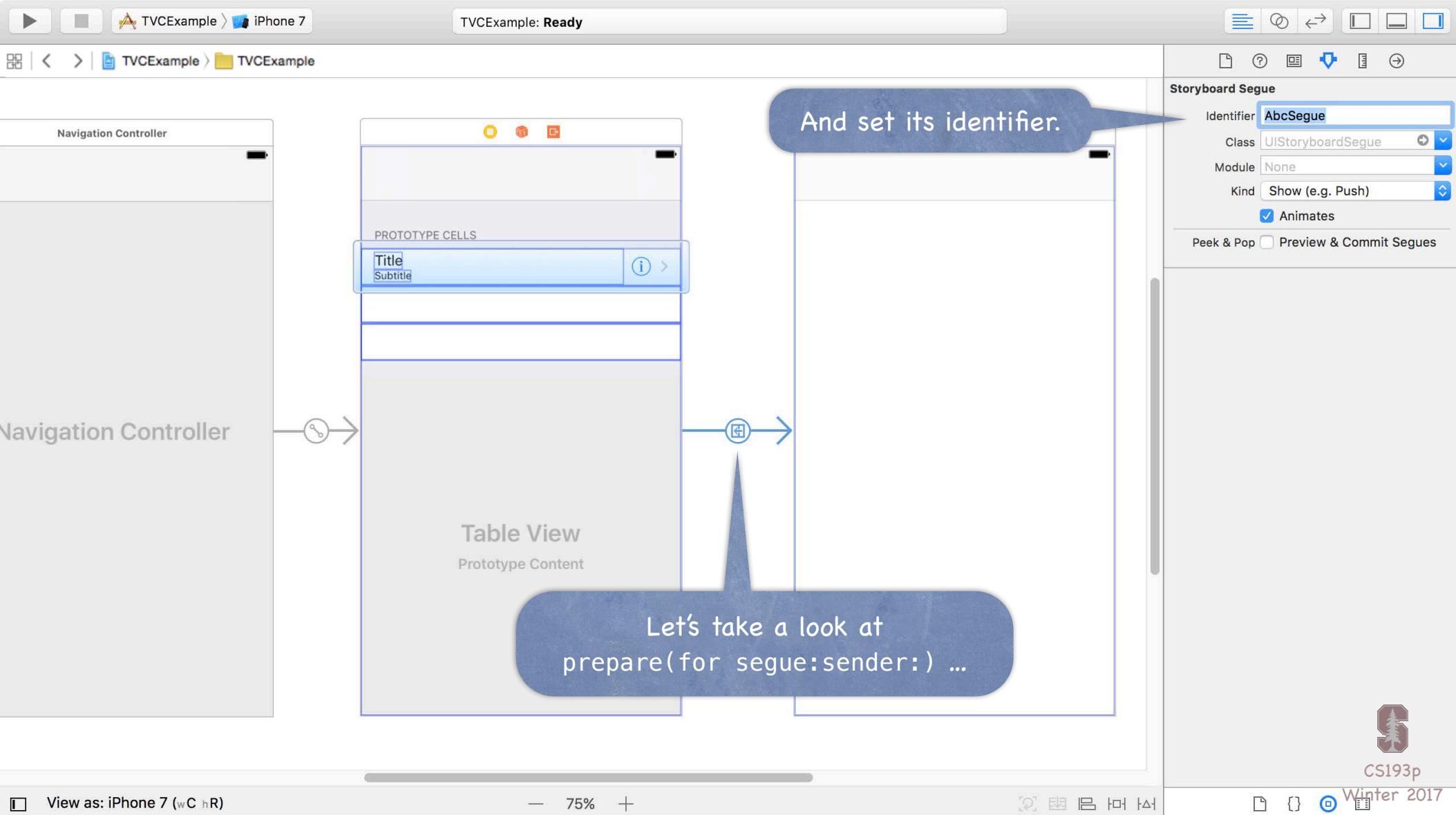












# Table View Segues

## ⌚ Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if let identifier = segue.identifier {  
        switch identifier {  
            case "XyzSegue": // handle XyzSegue here  
            case "AbcSegue":  
                break  
            default: break  
        }  
    }  
}
```

You can see now why `sender` is `Any`

Sometimes it's a UIButton, sometimes it's a UITableViewCell



CS193p

Winter 2017

# Table View Segues

## ⌚ Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if let identifier = segue.identifier {  
        switch identifier {  
            case "XyzSegue": // handle XyzSegue here  
            case "AbcSegue":  
                if let cell = sender as? MyTableViewCell {  
                    // do something with cell  
                }  
            default: break  
        }  
    }  
}
```

So you will need to cast sender with as? to turn it into a UITableViewCell

If you have a custom UITableViewCell subclass, you can cast it to that if it matters



CS193p

Winter 2017

# Table View Segues

## ⌚ Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    if let identifier = segue.identifier {  
        switch identifier {  
            case "XyzSegue": // handle XyzSegue here  
            case "AbcSegue":  
                if let cell = sender as? UITableViewCell,  
                    let indexPath = tableView.indexPath(for: cell) {  
  
                }  
                default: break  
            }  
    }  
}
```

indexPath(for cell:) does not accept Any.  
It has to be a UITableViewCell of some sort.

Usually we will need the IndexPath of the UITableViewCell  
Because we use that to index into our internal data structures



# Table View Segues

## ⌚ Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case "XyzSegue": // handle XyzSegue here
        case "AbcSegue":
            if let cell = sender as? MyTableViewCell,
               let indexPath = tableView.indexPath(for: cell),
               let seguedToMVC = segue.destination as? MyVC {
                ...
            }
        default: break
    }
}
```

Now we just get our destination MVC as the proper class as usual ...



CS193p

Winter 2017

# Table View Segues

## • Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case "XyzSegue": // handle XyzSegue here
        case "AbcSegue":
            if let cell = sender as? MyTableViewCell,
                let indexPath = tableView.indexPath(for: cell),
                let seguedToMVC = segue.destination as? MyVC {
                    seguedToMVC.publicAPI = data[indexPath.section][indexPath.row]
            }
        default: break
    }
}
```

and then get data from our internal data structure using the IndexPath's section and row



# Table View Segues

## • Preparing to segue from a row in a table view

The sender argument to prepareForSegue is the UITableViewCell of that row ...

```
func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if let identifier = segue.identifier {
        switch identifier {
        case "XyzSegue": // handle XyzSegue here
        case "AbcSegue":
            if let cell = sender as? MyTableViewCell,
                let indexPath = tableView.indexPath(for: cell),
                let seguedToMVC = segue.destination as? MyVC {
                    seguedToMVC.publicAPI = data[indexPath.section][indexPath.row]
            }
        default: break
    }
}
```

and then get data from our internal data structure using the IndexPath's section and row  
and use that information to prepare the segued-to API using its public API



CS193p

Winter 2017

# UITableViewDelegate

- ⦿ So far we've only talked about the UITableView's dataSource
  - But UITableView has another protocol-driven delegate called its delegate
- ⦿ The delegate controls how the UITableView is displayed
  - Not the data it displays (that's the dataSource's job), how it is displayed
- ⦿ Common for dataSource and delegate to be the same object
  - Usually the Controller of the MVC containing the UITableView
  - Again, this is set up automatically for you if you use UITableViewController
- ⦿ The delegate also lets you observe what the table view is doing
  - Especially responding to when the user selects a row
  - Usually you will just segue when this happens, but if you want to track it directly ...



# UITableView “Target/Action”

- ⌚ UITableViewDelegate method sent when row is selected

This is sort of like “table view target/action” (only needed if you’re not segueing, of course)

Example: if the master in a split view wants to update the detail without segueing to a new one

```
func tableView(UITableView, didSelectRowAt indexPath: IndexPath) {  
    // go do something based on information about my Model  
    // corresponding to indexPath.row in indexPath.section  
    // maybe directly update the Detail if I'm the Master in a split view?  
}
```

- ⌚ Delegate method sent when Detail Disclosure button is touched

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

Again, you can just segue from that Detail Disclosure button if you prefer



CS193p

Winter 2017

# UITableViewDelegate

- 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 (i.e. deleting, inserting, moving rows)

- Copying/pasting rows



# UITableView

## ⌚ What if your Model changes?

```
func reloadData()
```

Causes the UITableView to call `numberOfSectionsInTableView` and `numberOfRowsInSection` all over again and then `cellForRowAtIndex` on each visible row

Relatively heavyweight, 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)
```



CS193p

Winter 2017

# UITableView

## • Controlling the height of rows

Row height can be fixed (UITableView's `var rowHeight: CGFloat`)

Or it can be determined using autolayout (`rowHeight = UITableViewAutomaticDimension`)

If you do automatic, help the table view out by setting `estimatedRowHeight` to something

The UITableView's delegate can also control row heights ...

```
func tableView(UITableView, {estimated}heightForRowAt indexPath: IndexPath) -> CGFloat
```

Beware: the non-estimated version of this could get called A LOT if you have a big table



CS193p

Winter 2017

# UITableView

- There are dozens of other methods in UITableView itself

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 (UITableView is a subclass of UIScrollView).

Selection management (allows multiple selection, getting the selected row, etc.).

Moving, inserting and deleting rows, etc.

As always, part of learning the material in this course is studying the documentation



CS193p

Winter 2017