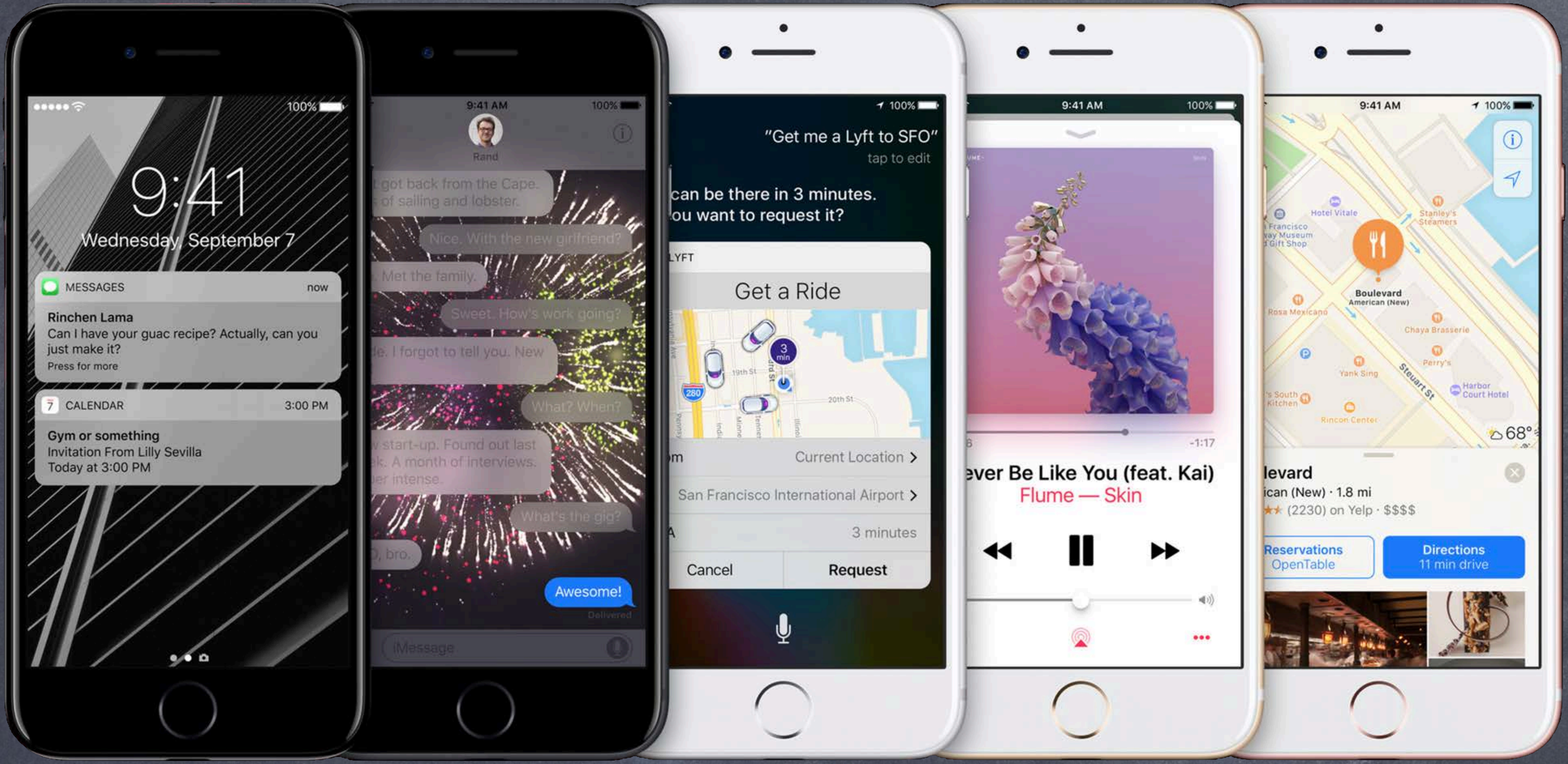


Stanford CS193p

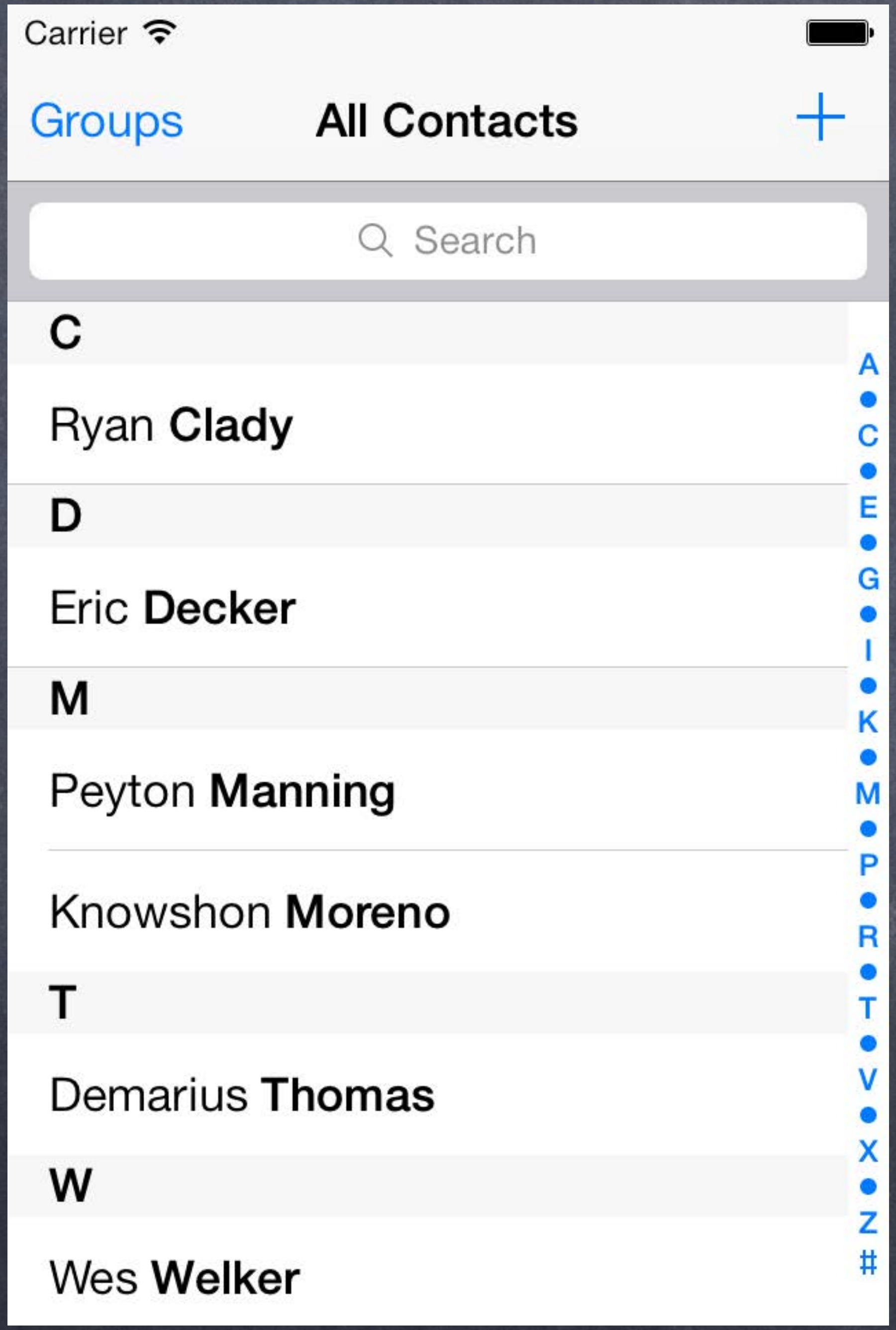
Developing Applications for iOS
Winter 2017

Today

- Table View
  - Way to display large data sets
  - Demo: Twitter Client

UITableView

`UITableViewStyle.plain`



Carrier

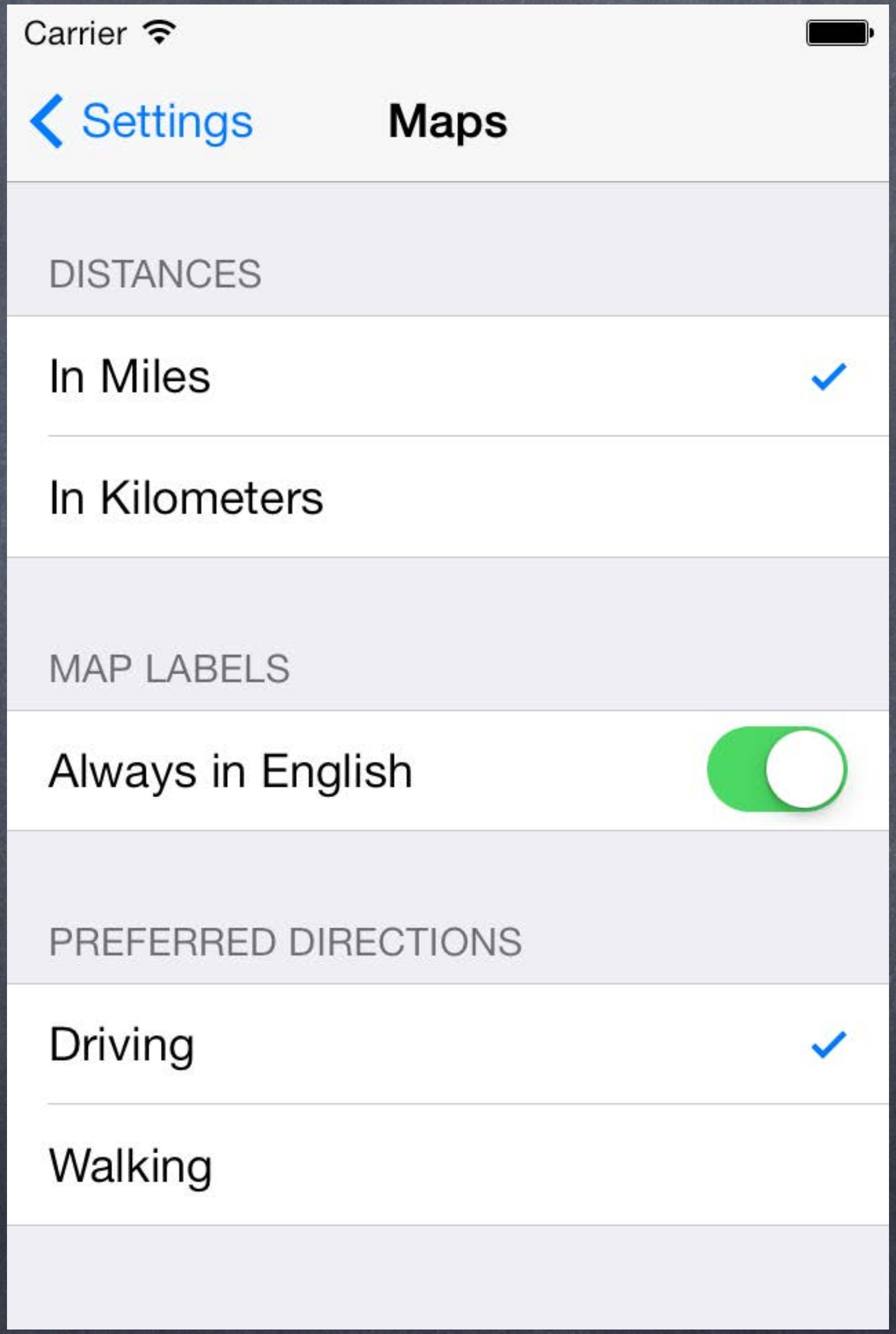
Groups | All Contacts | +

Search

C
Ryan **Clady**
D
Eric **Decker**
M
Peyton **Manning**
Knowshon **Moreno**
T
Demarius **Thomas**
W
Wes **Welker**

Dynamic (List) & Plain (ungrouped)

`.grouped`



Carrier

Settings | Maps

DISTANCES
- In Miles ✓
- In Kilometers

MAP LABELS
- Always in English (on)

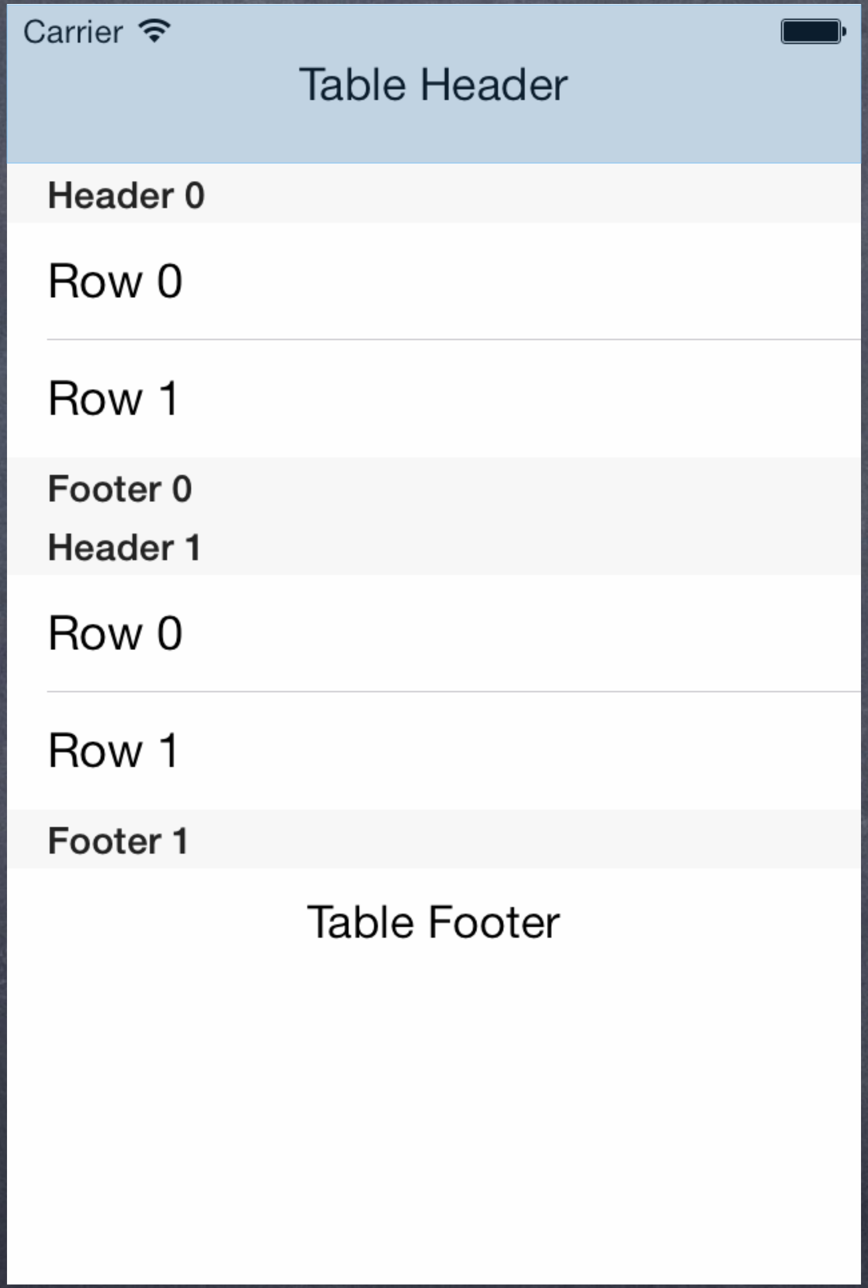
PREFERRED DIRECTIONS
- Driving ✓
- Walking

Static & Grouped

UITableView

Plain Style

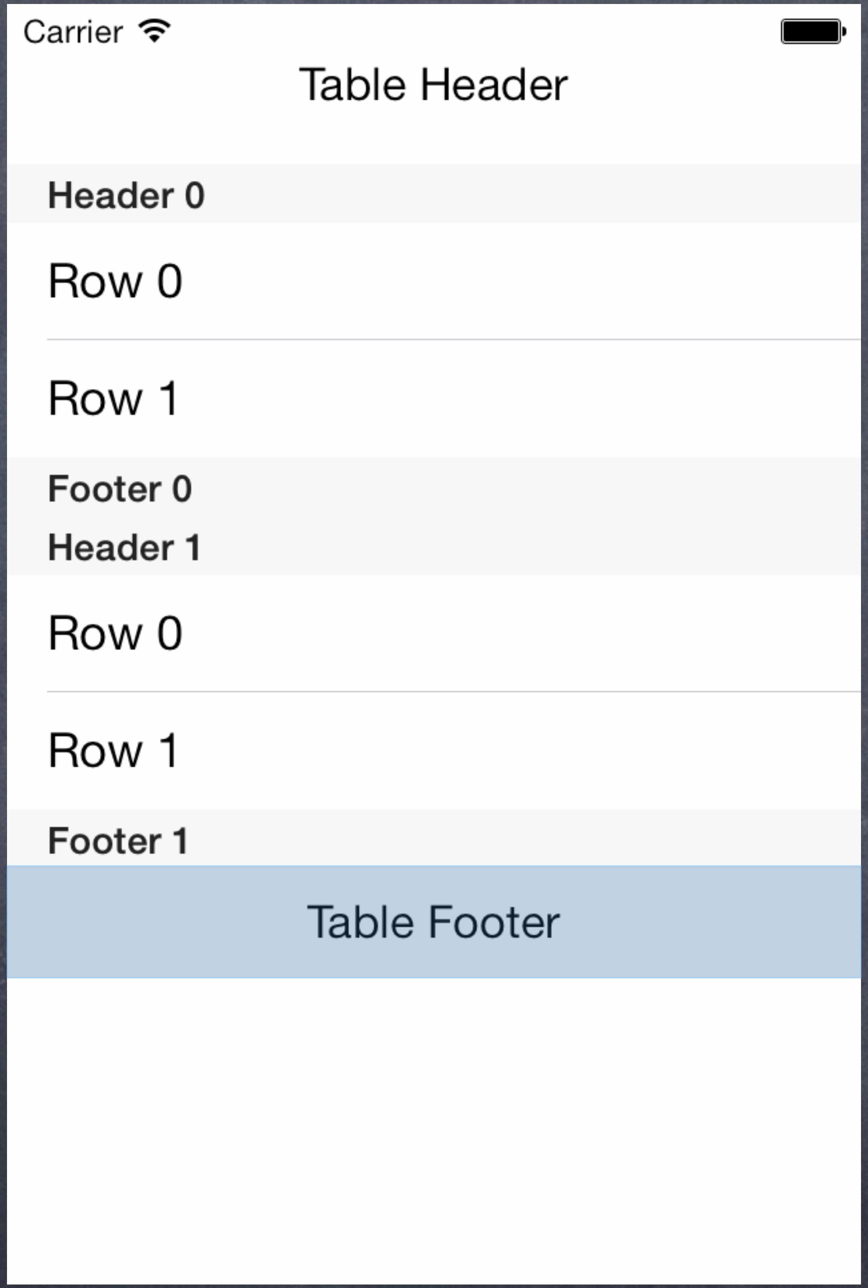
Table Header



```
var tableHeaderView: UIView
```

UITableView

Plain Style



Carrier

Table Header

Header 0

Row 0

Row 1

Footer 0

Header 1

Row 0

Row 1

Footer 1

Table Footer

Table Header

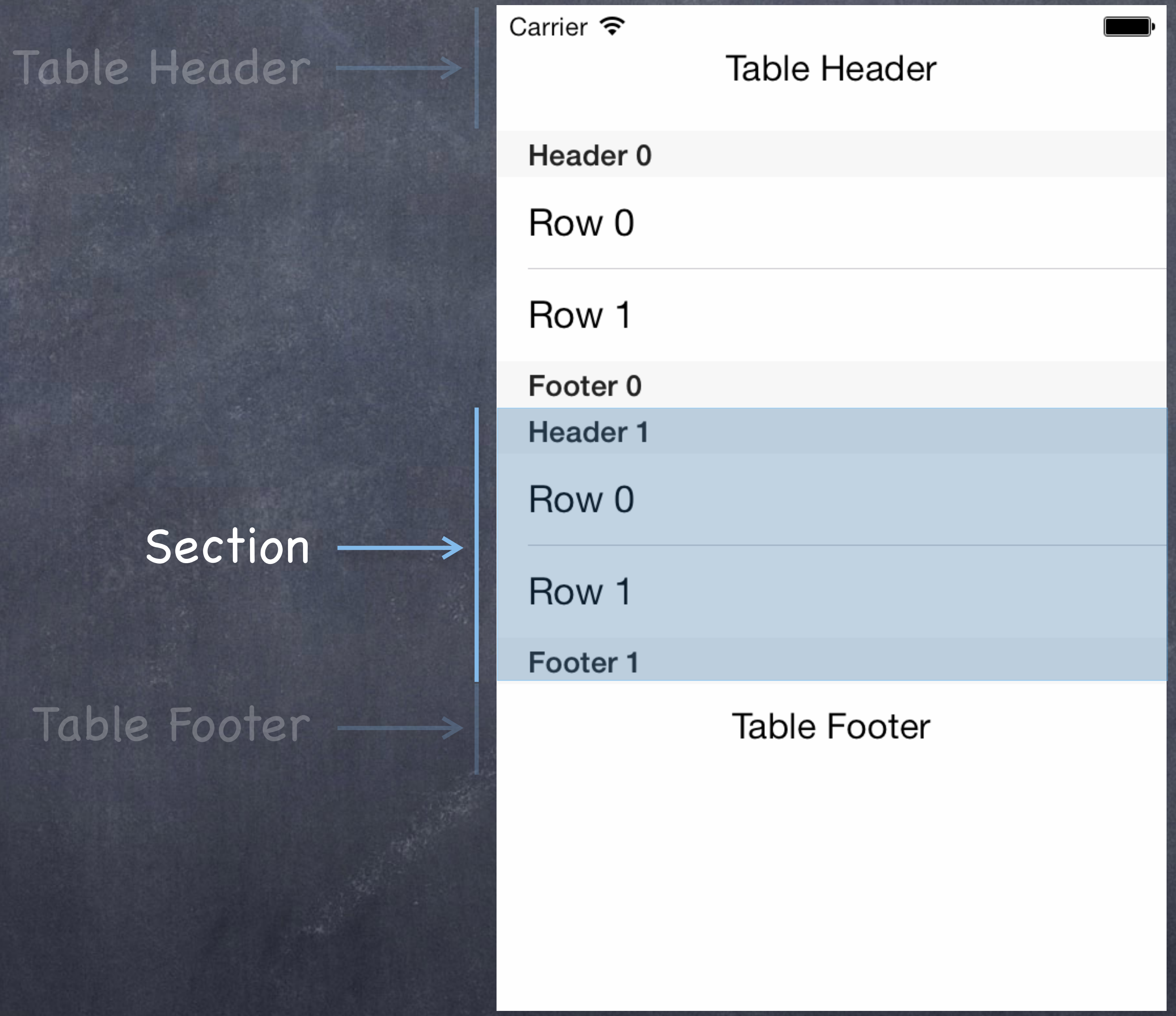
Table Footer

`var tableFooterView: UIView`

CS193p
Winter 2017

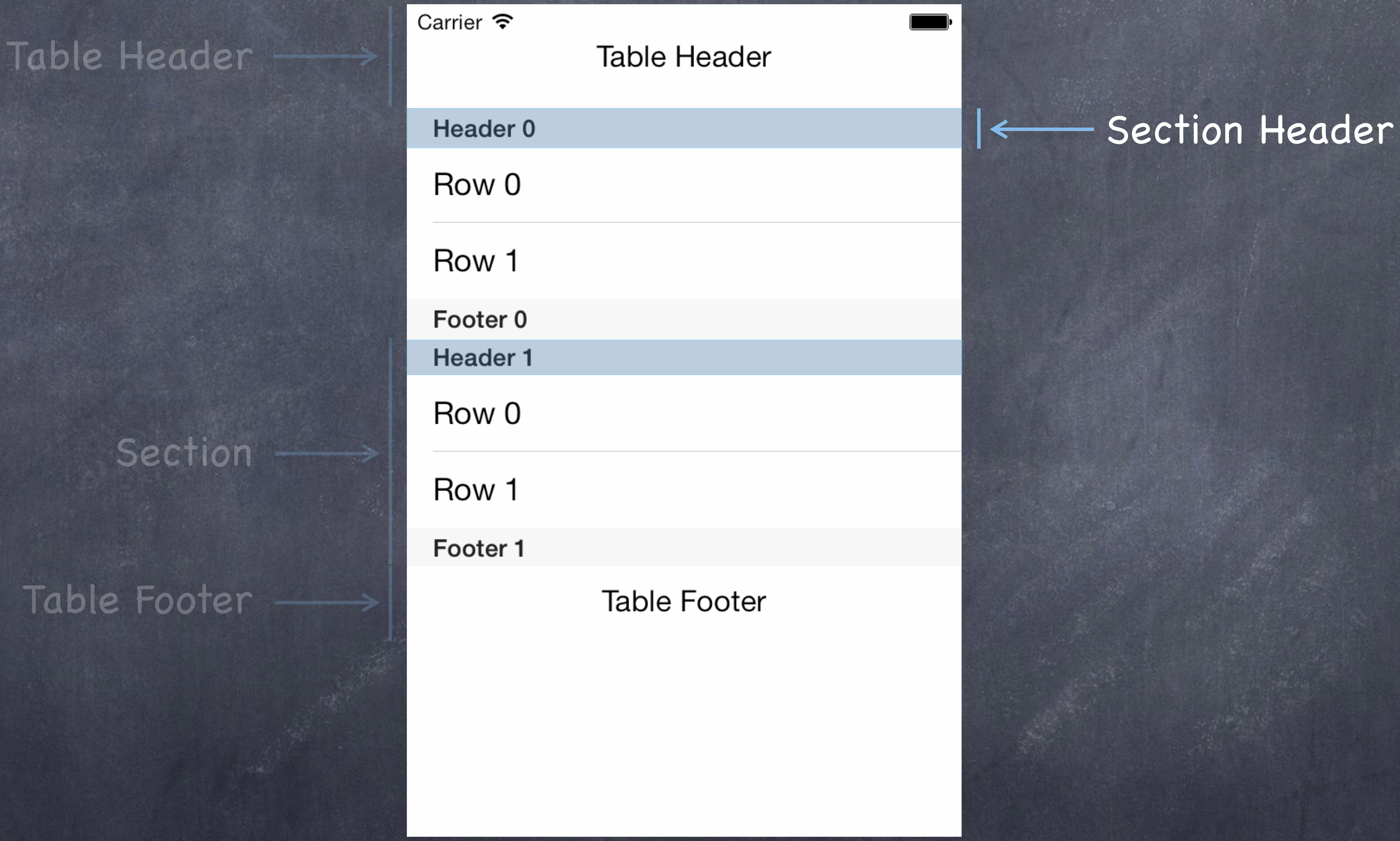
UITableView

Plain Style



UITableView

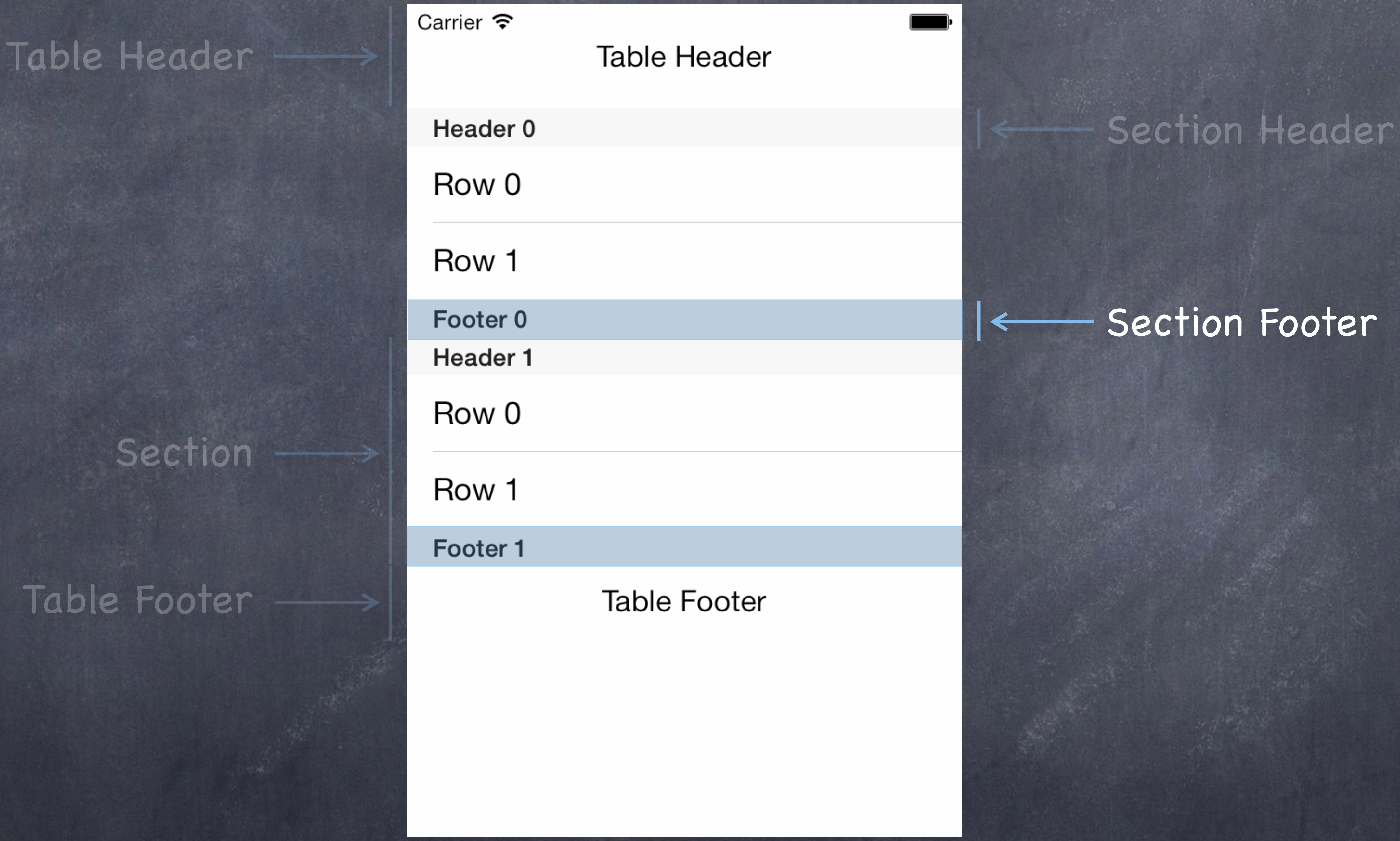
Plain Style



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

UITableView

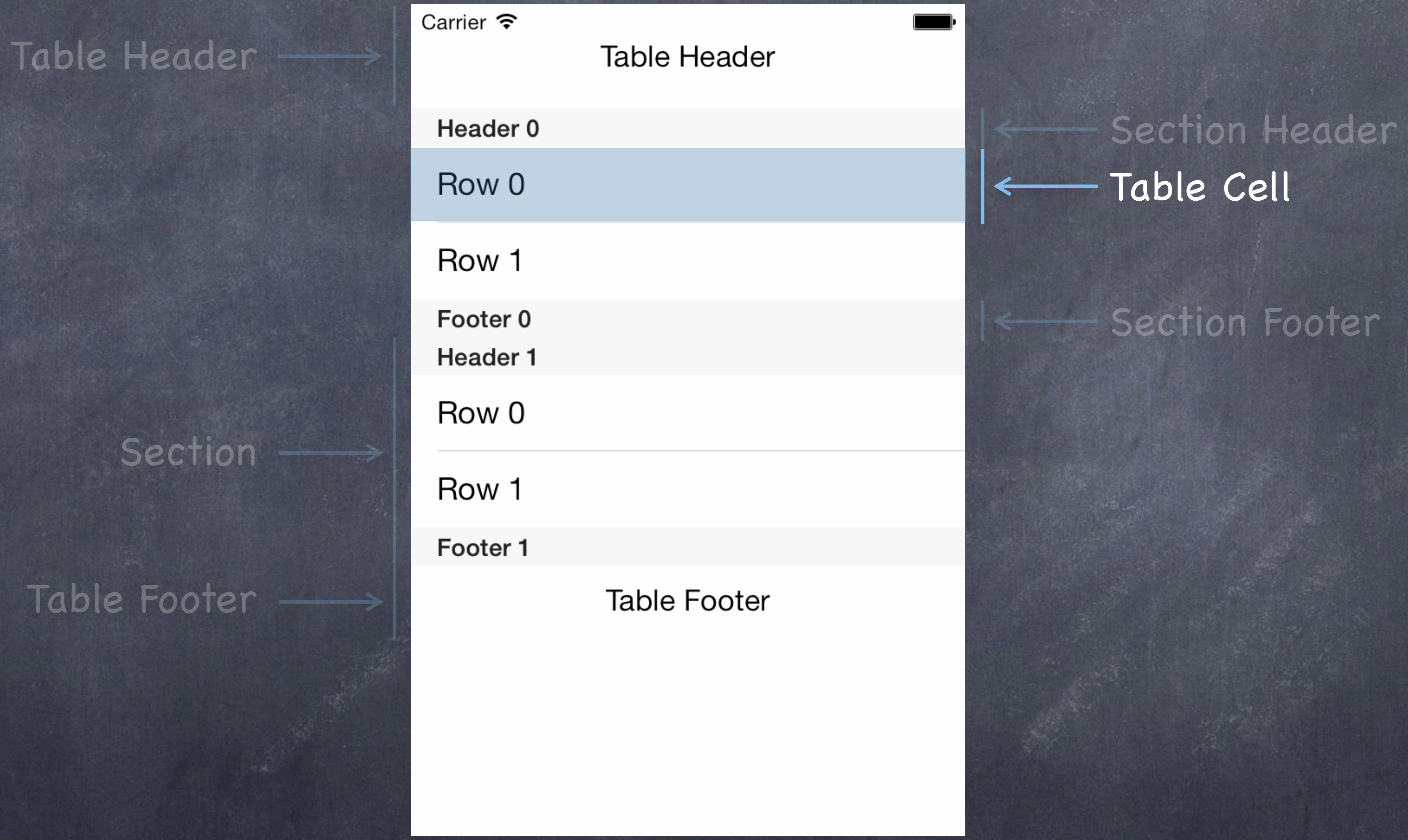
Plain Style



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

UITableView

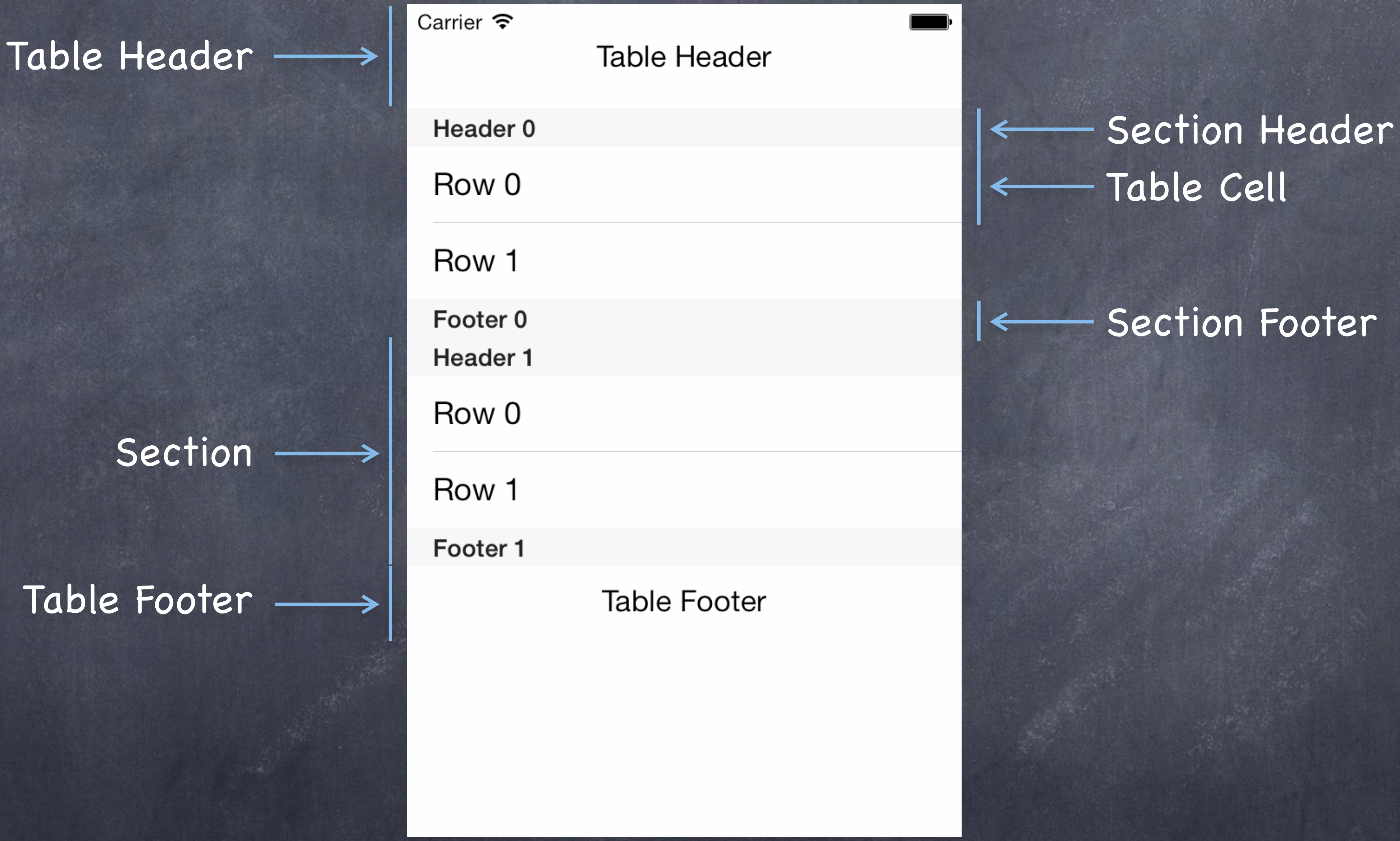
Plain Style



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

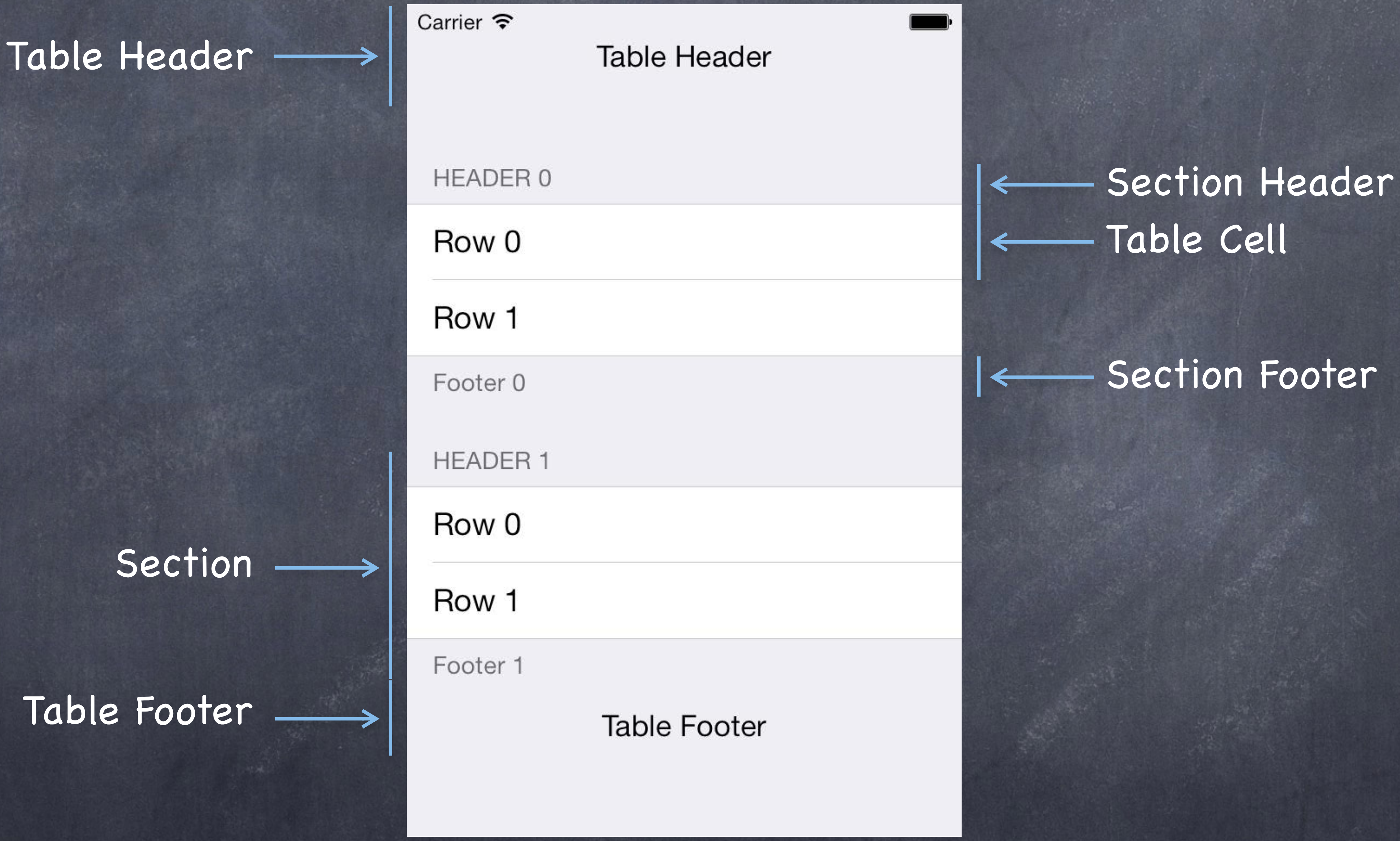
UITableView

Plain Style

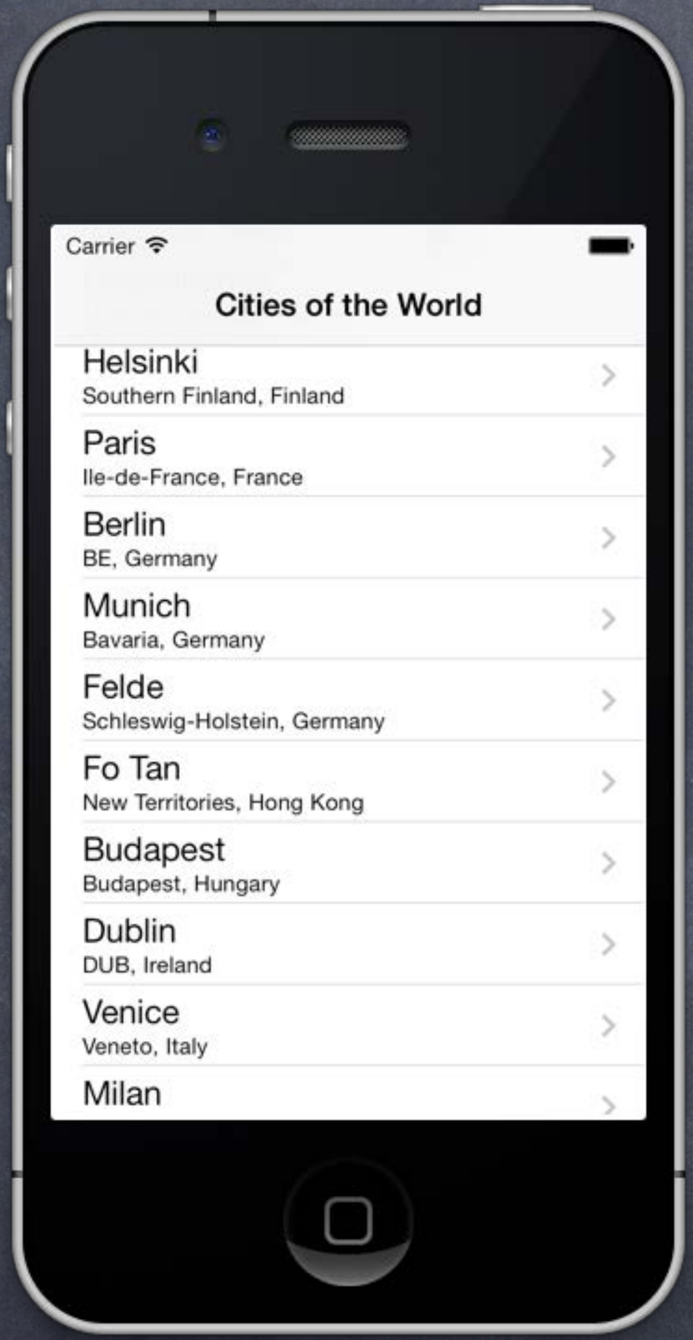


UITableView

Grouped Style



Sections or Not



Carrier

Cities of the World

Helsinki
Southern Finland, Finland

Paris
Ile-de-France, France

Berlin
BE, Germany

Munich
Bavaria, Germany

Felde
Schleswig-Holstein, Germany

Fo Tan
New Territories, Hong Kong

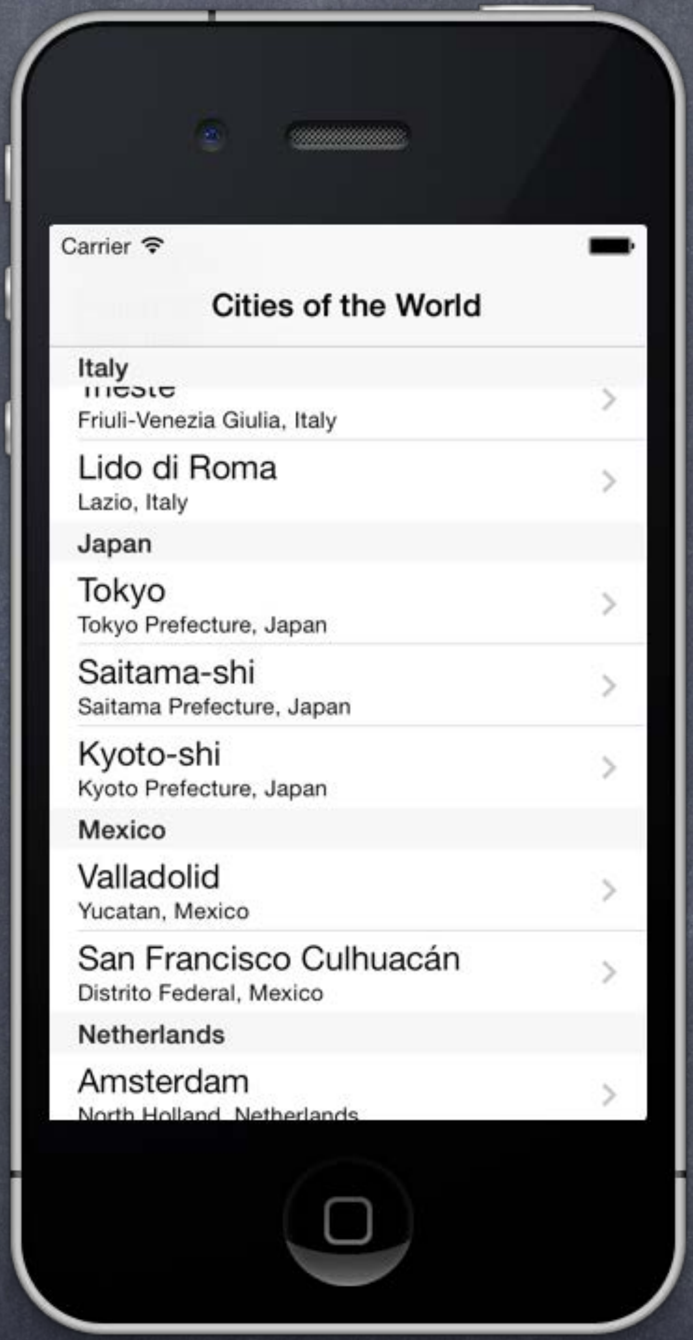
Budapest
Budapest, Hungary

Dublin
DUB, Ireland

Venice
Veneto, Italy

Milan

No Sections



Carrier

Cities of the World

Italy

Friuli-Venezia Giulia, Italy

Lido di Roma
Lazio, Italy

Japan

Tokyo
Tokyo Prefecture, Japan

Saitama-shi
Saitama Prefecture, Japan

Kyoto-shi
Kyoto Prefecture, Japan

Mexico

Valladolid
Yucatan, Mexico

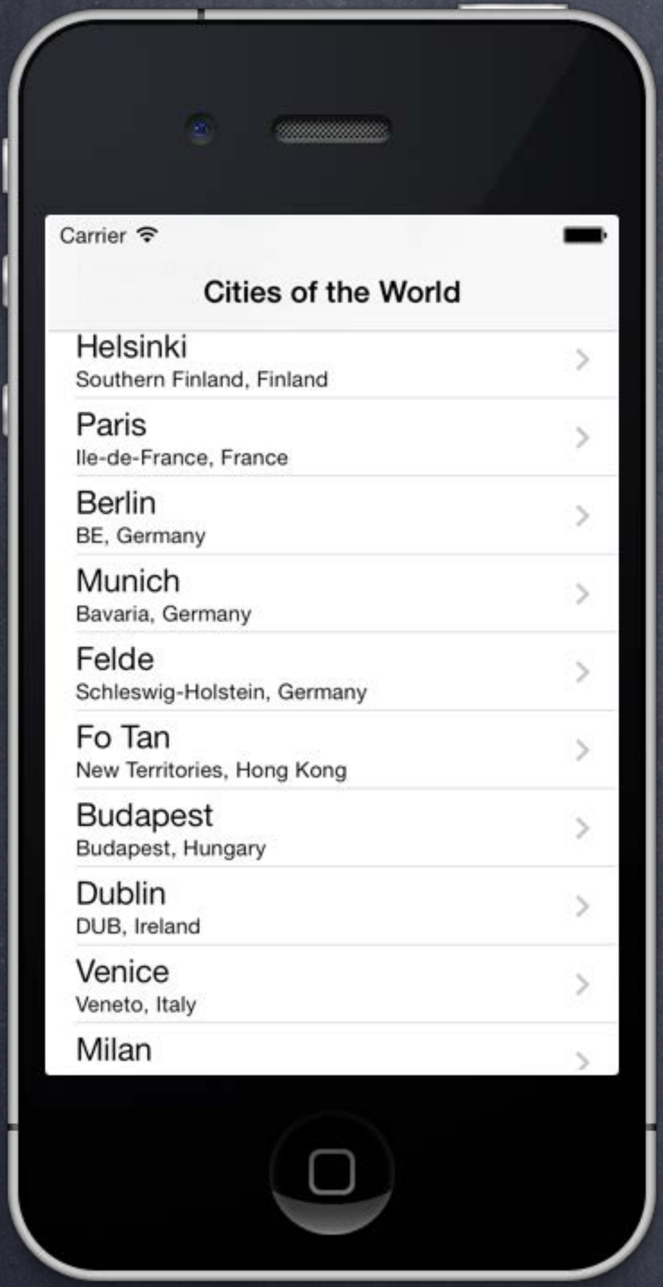
San Francisco Culhuacán
Distrito Federal, Mexico

Netherlands

Amsterdam
North Holland, Netherlands

Sections

Cell Type

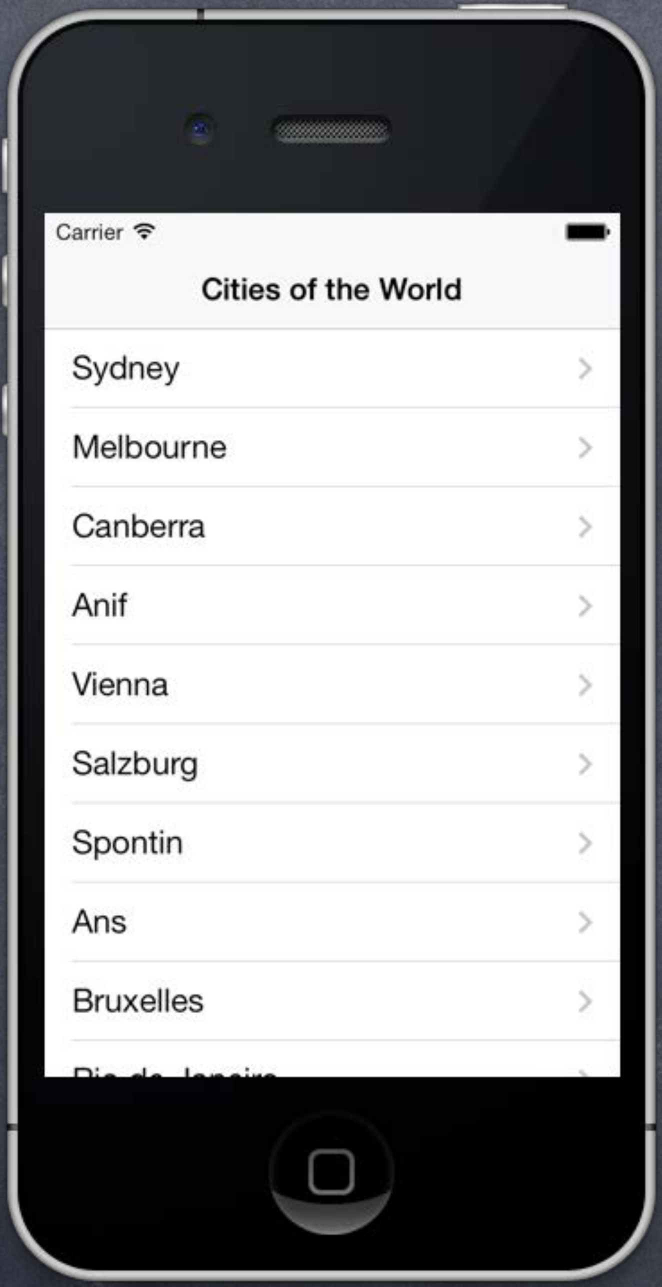

Carrier

Cities of the World

Helsinki	Southern Finland, Finland
Paris	Ile-de-France, France
Berlin	BE, Germany
Munich	Bavaria, Germany
Felde	Schleswig-Holstein, Germany
Fo Tan	New Territories, Hong Kong
Budapest	Budapest, Hungary
Dublin	DUB, Ireland
Venice	Veneto, Italy
Milan	

Subtitle

`UITableViewCellStyle.subtitle`

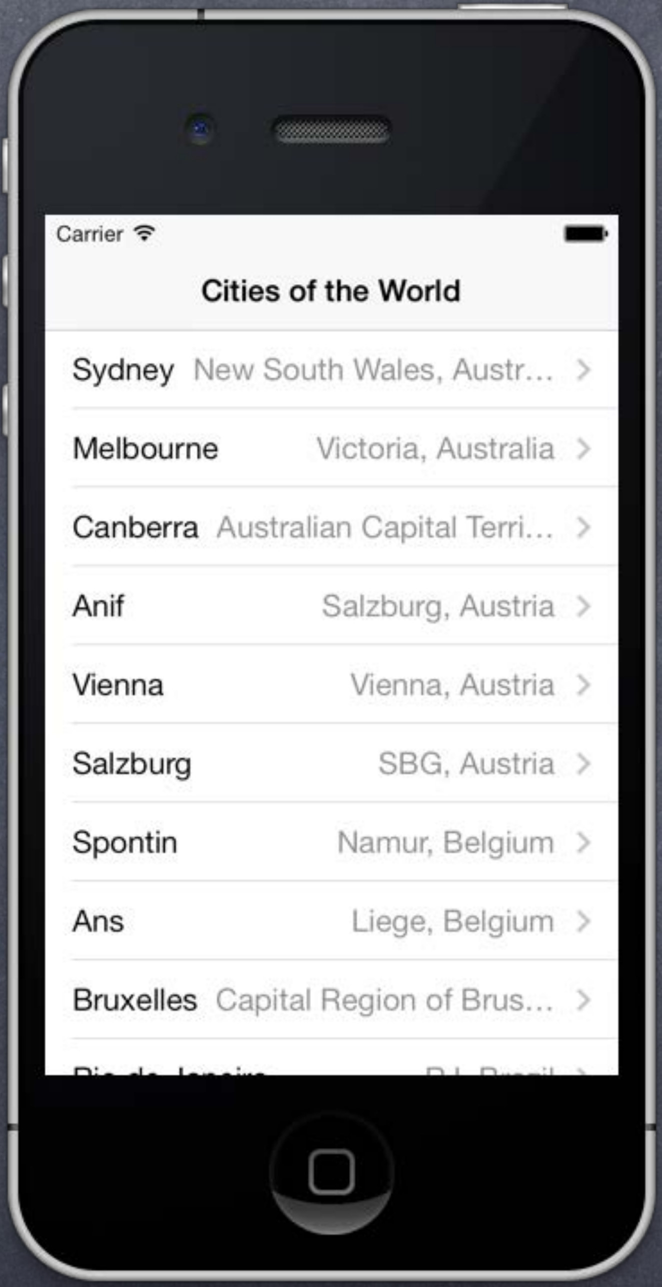

Carrier

Cities of the World

Sydney
Melbourne
Canberra
Anif
Vienna
Salzburg
Spontin
Ans
Bruxelles

Basic

`.default`

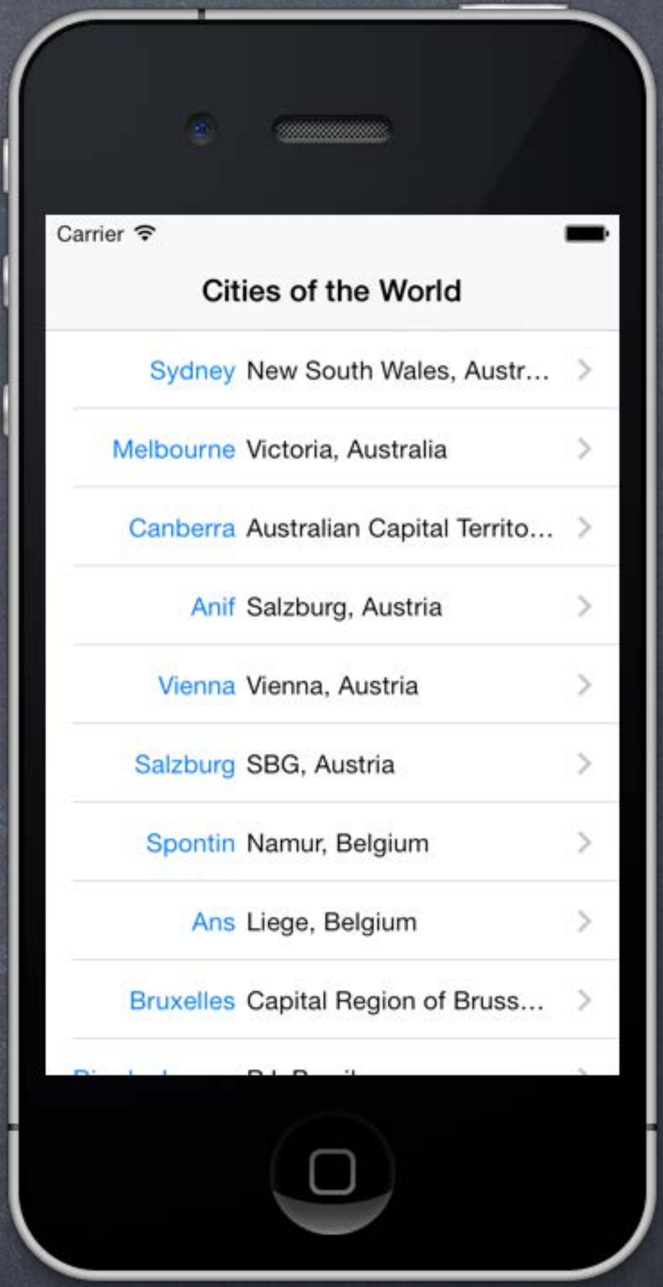

Carrier

Cities of the World

Sydney	New South Wales, Austr...
Melbourne	Victoria, Australia
Canberra	Australian Capital Terri...
Anif	Salzburg, Austria
Vienna	Vienna, Austria
Salzburg	SBG, Austria
Spontin	Namur, Belgium
Ans	Liege, Belgium
Bruxelles	Capital Region of Brus...

Right Detail

`.value1`

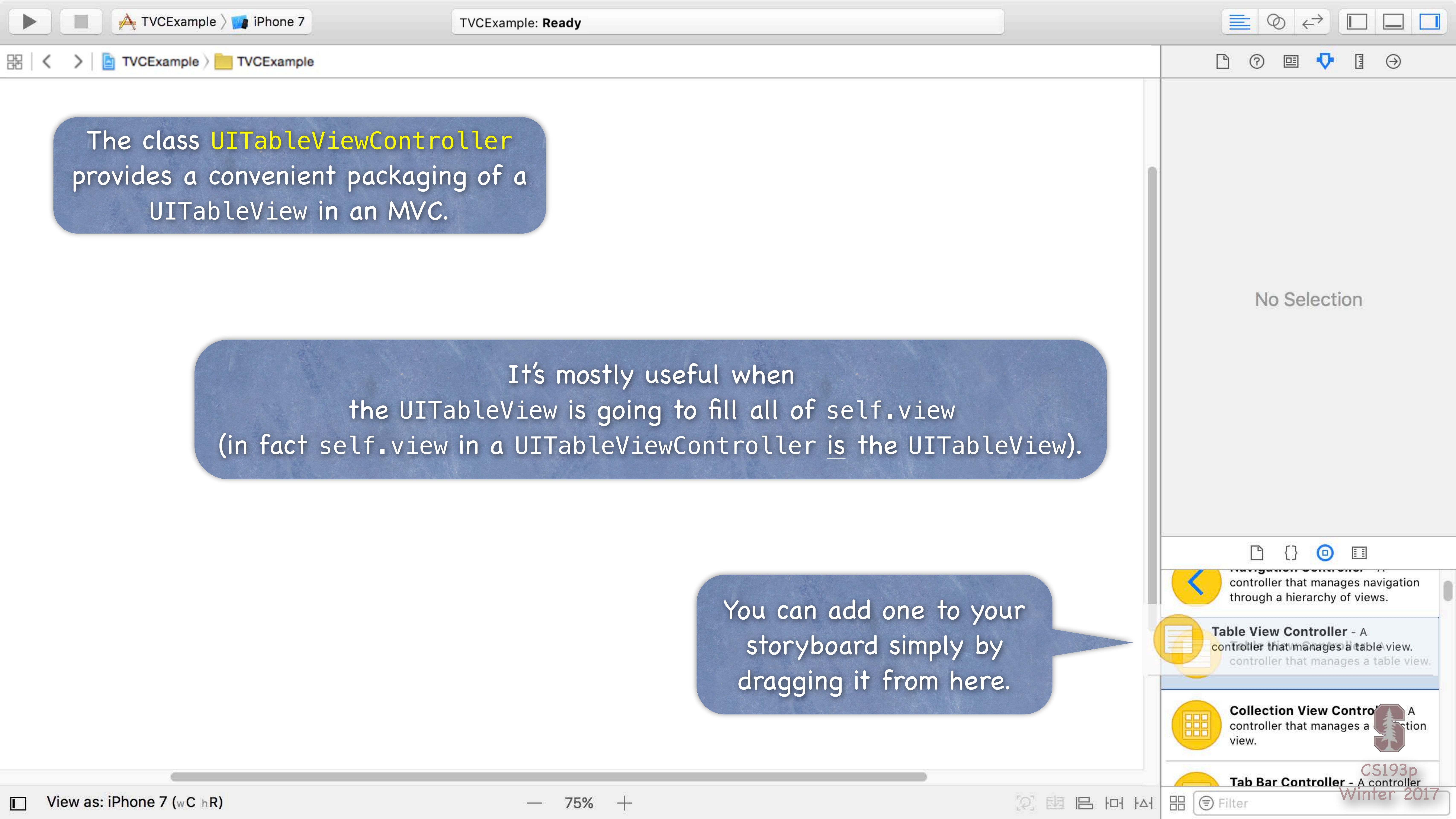

Carrier

Cities of the World

Sydney	New South Wales, Austr...
Melbourne	Victoria, Australia
Canberra	Australian Capital Territo...
Anif	Salzburg, Austria
Vienna	Vienna, Austria
Salzburg	SBG, Austria
Spontin	Namur, Belgium
Ans	Liege, Belgium
Bruxelles	Capital Region of Bruss...

Left Detail

`.value2`



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

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.

No Selection

controller that manages navigation through a hierarchy of views.

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

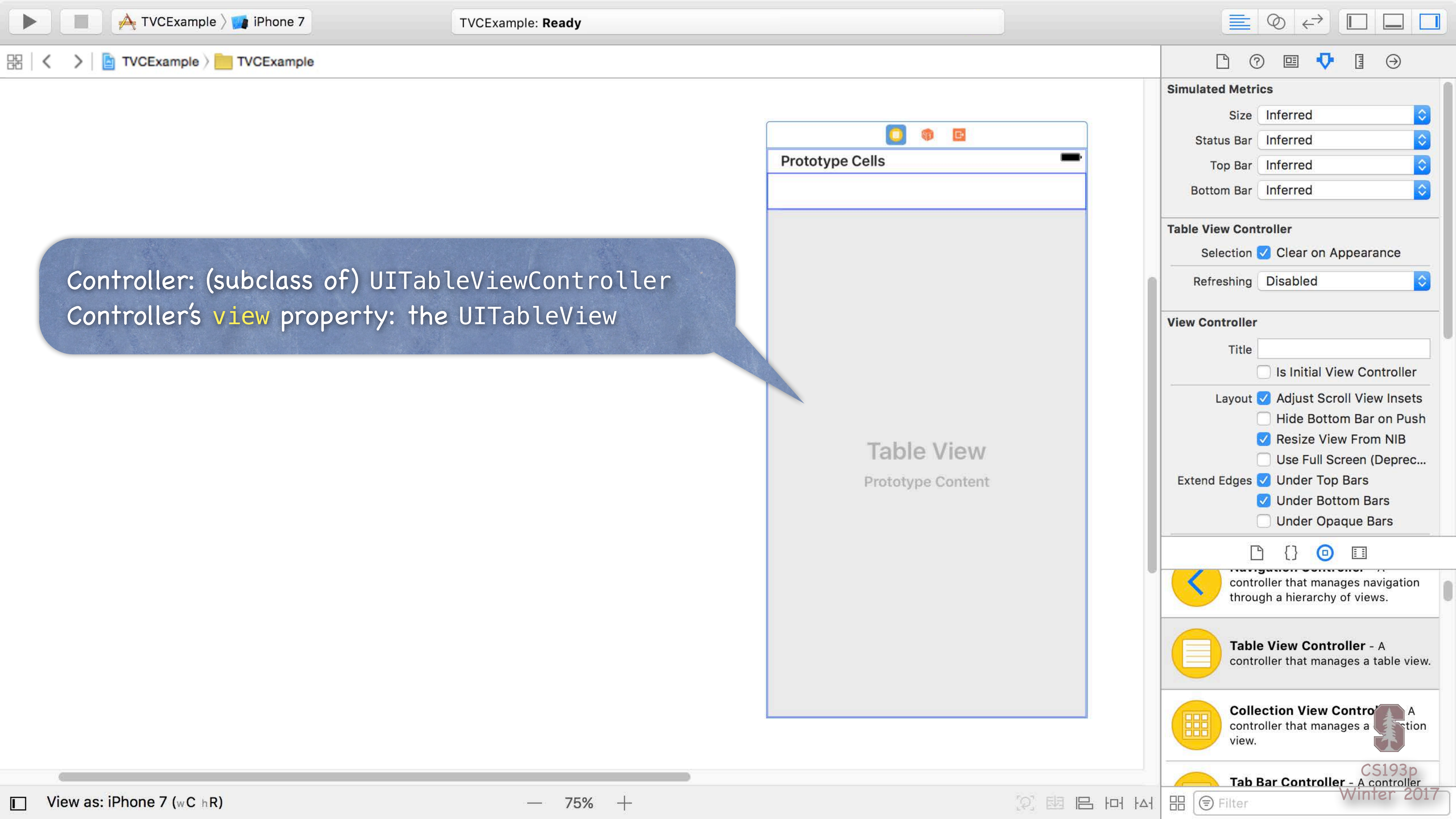
Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller

Filter

View as: iPhone 7 (wC hR)

75%



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

Simulated Metrics

Size	Inferred
Status Bar	Inferred
Top Bar	Inferred
Bottom Bar	Inferred

Table View Controller

Selection [✓ Clear on Appearance]
Refreshing: Disabled

View Controller

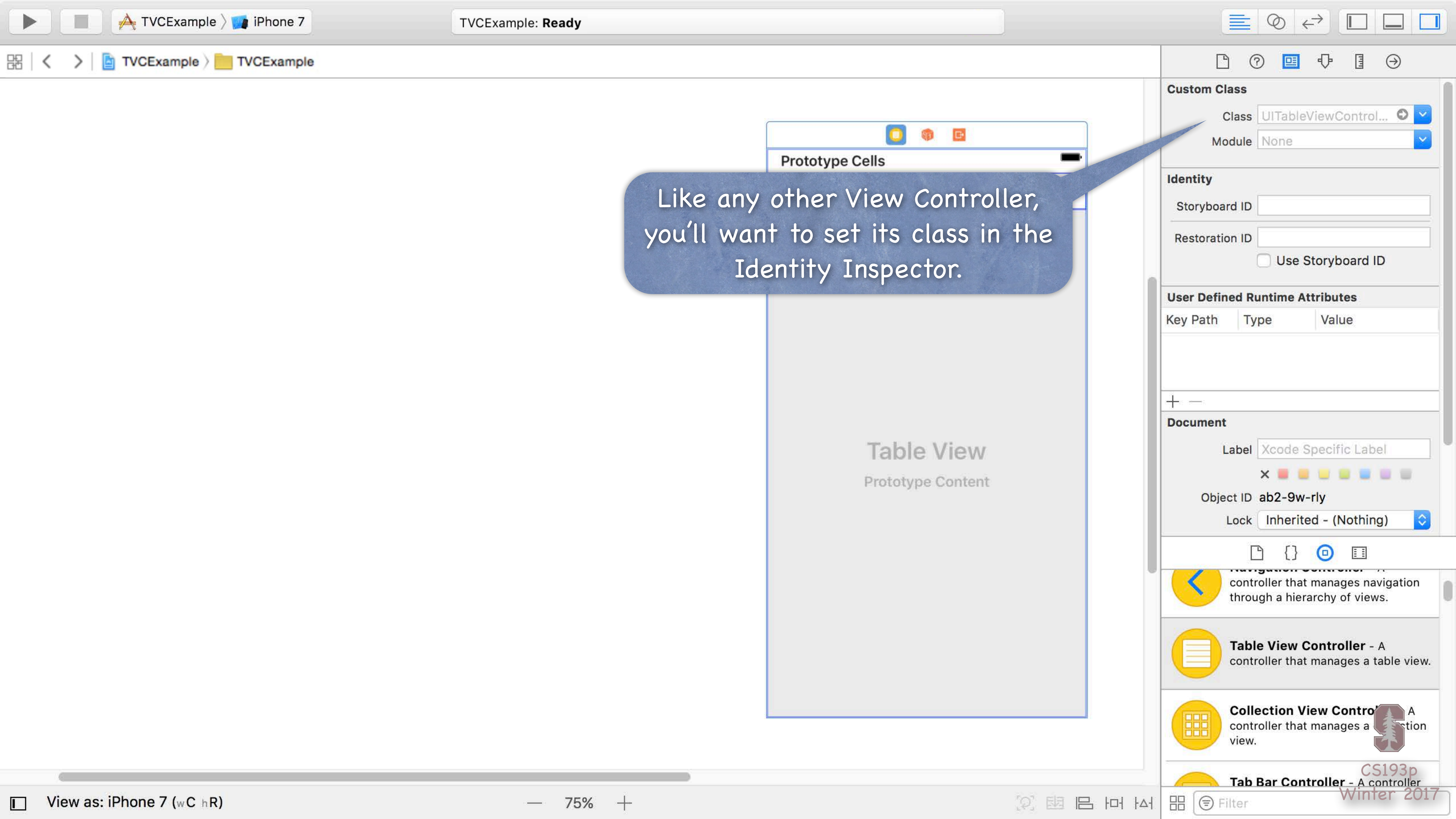
Title:
Is Initial View Controller
Layout [✓ Adjust Scroll View Insets]
Hide Bottom Bar on Push
[✓ Resize View From NIB]
Use Full Screen (Deprec...
Extend Edges [✓ Under Top Bars]
[✓ Under Bottom Bars]
Under Opaque Bars

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



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

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)

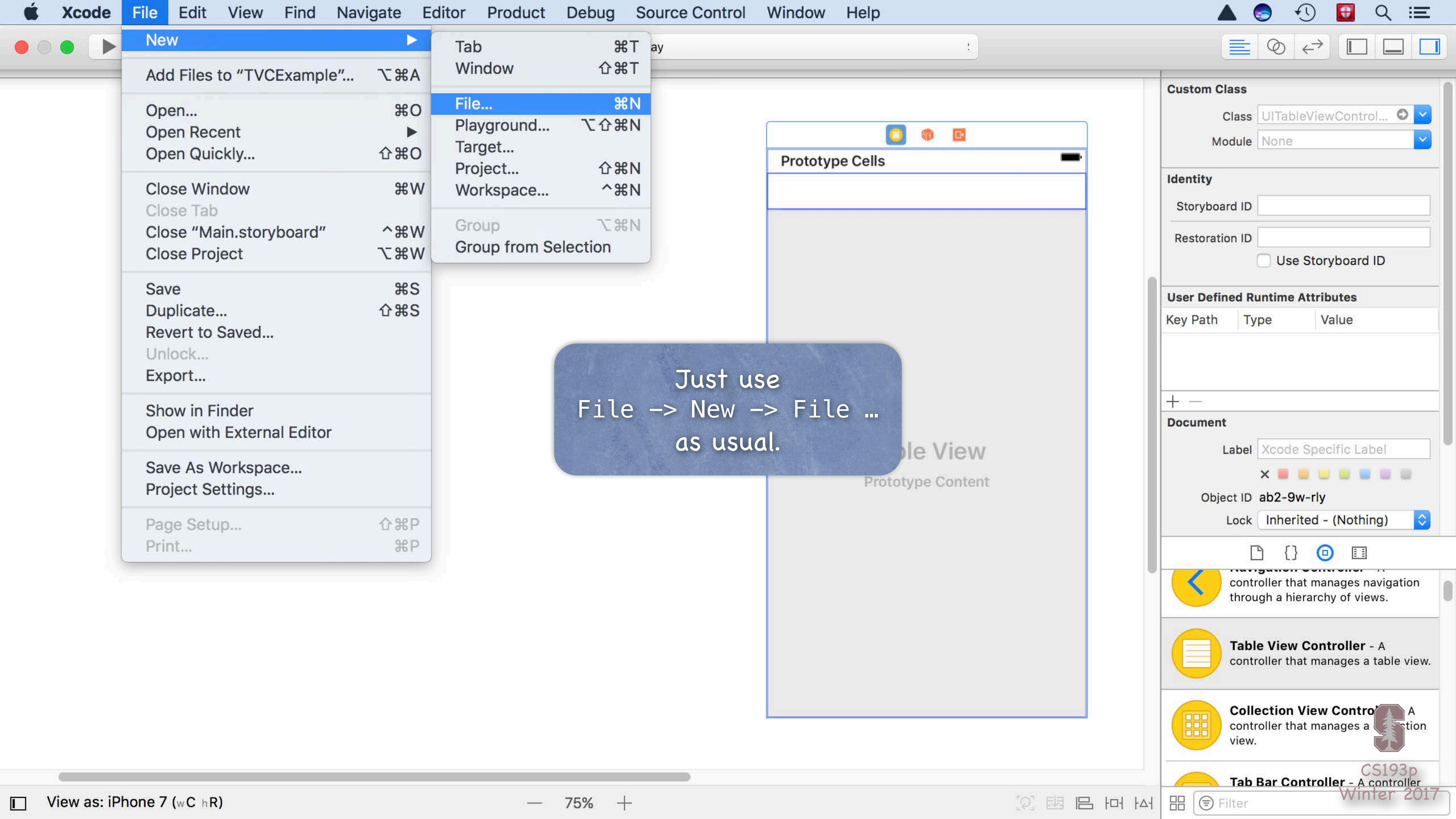
controller that manages navigation through a hierarchy of views.

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

Collection View Contro... A ... controller that manages a ...tion view.

Tab Bar Controller - A controller

Filter



Custom Class
Class: UITableViewControl...
Module: None

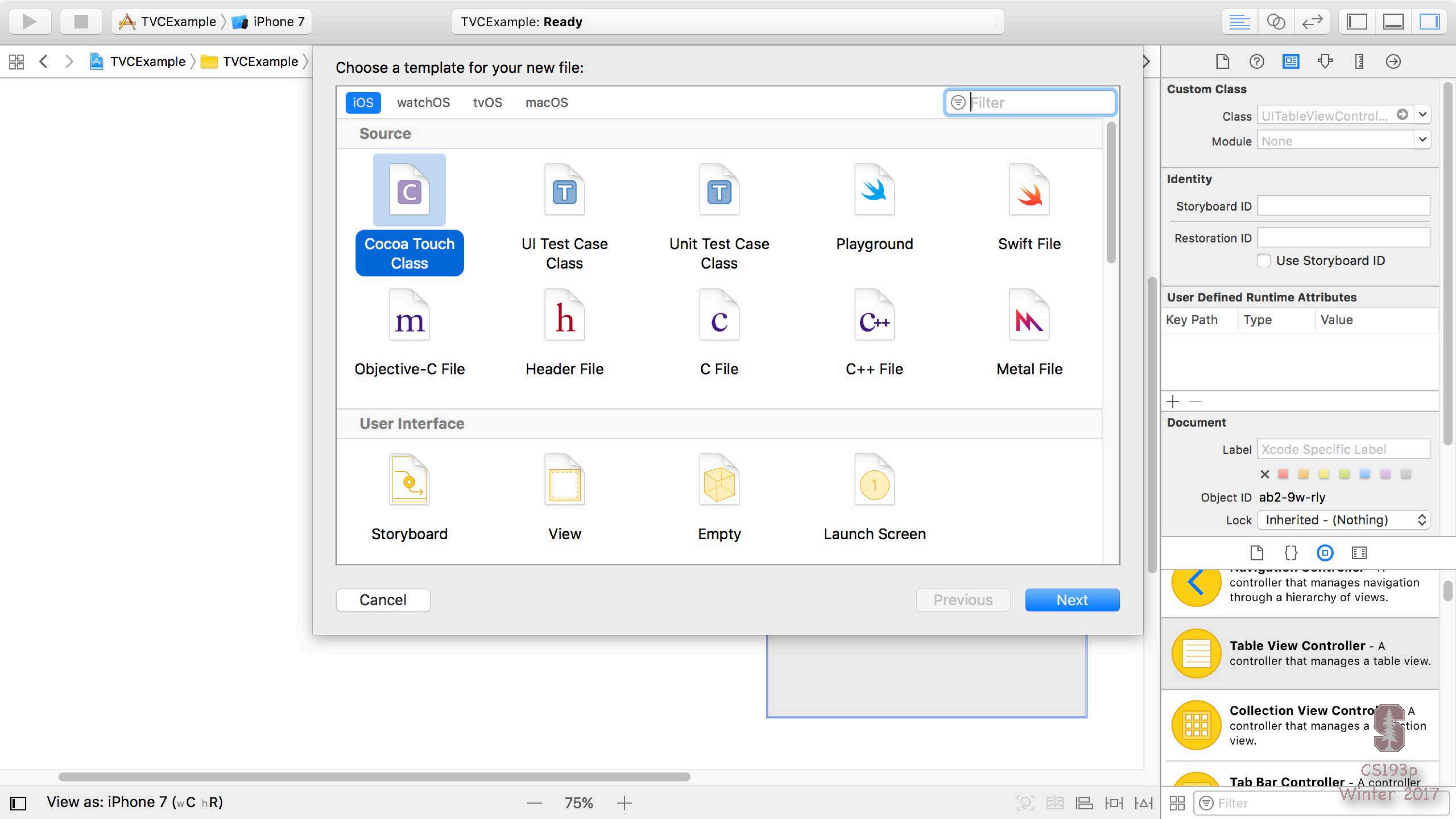
Identity
Storyboard ID:
Restoration ID:
Use Storyboard ID

User Defined Runtime Attributes

Key Path	Type	Value

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

Just use
File -> New -> File …
as usual.



TVCExample › iPhone 7   TVCExample: **Ready**

TVCExample › TVCExample ›

Choose a template for your new file:

iOS watchOS tvOS macOS   Filter

Source

Cocoa Touch Class   UI Test Case Class   Unit Test Case Class   Playground   Swift File

Objective-C File   Header File   C File   C++ File   Metal File

User Interface

Storyboard   View   Empty   Launch Screen

Cancel   Previous   Next

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)

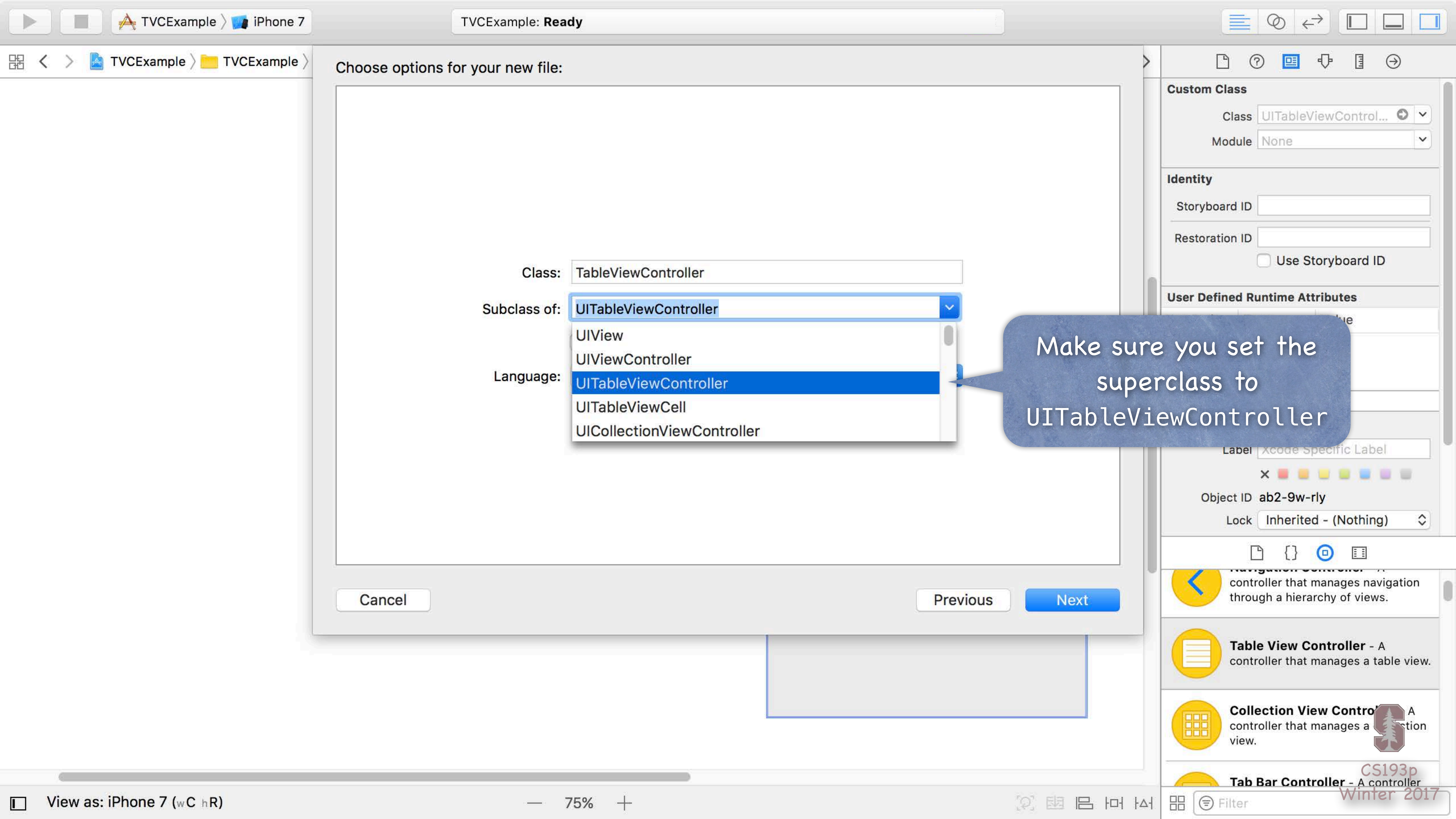
Navigation Controller - A controller that manages navigation through a hierarchy of views.

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

View as: iPhone 7 (wC hR)   75%



TVCExample: **Ready**

Choose options for your new file:

Class: TableViewController

Subclass of: UITableViewController

- UIView
- UIViewController
- UITableViewController
- UITableViewCell
- UICollectionViewController

Language:

Custom Class

Class: UITableViewControl...

Module: None

Identity

Storyboard ID

Restoration ID

Use Storyboard ID

User Defined Runtime Attributes

Label: Xcode Specific Label

Object ID: ab2-9w-rly

Lock: Inherited - (Nothing)

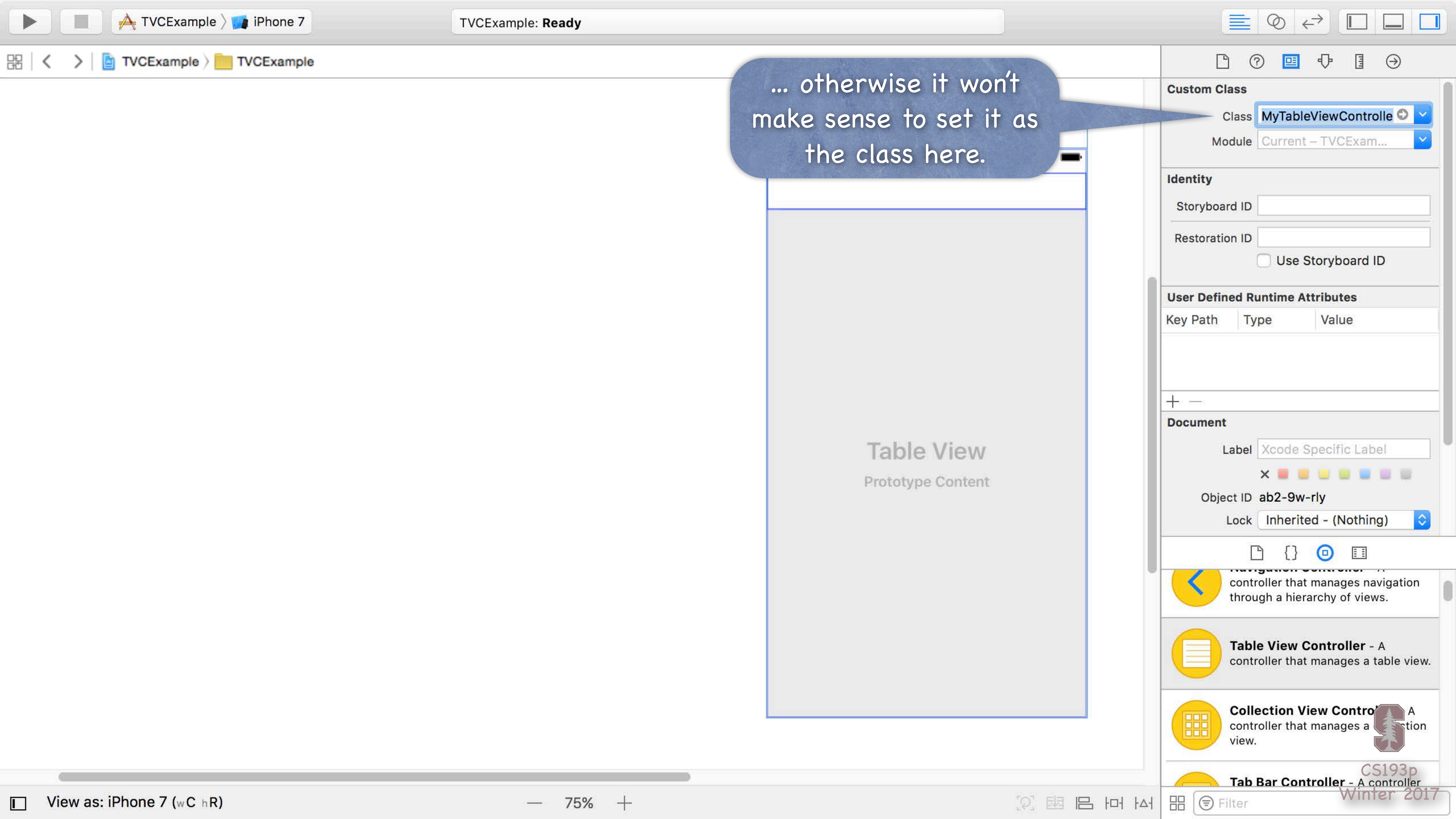
Table View Controller - A controller that manages a table view.

Collection View Contro... A controller that manages a ...tion view.

Tab Bar Controller - A controller

View as: iPhone 7 (wC hR)

Make sure you set the superclass to UITableViewController



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

TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Custom Class

Class MyTableViewControlle

Module Current – TVCExam...

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 Contro[illegible] - A controller that manages a [illegible]tion view.

Tab Bar Controller - A controller

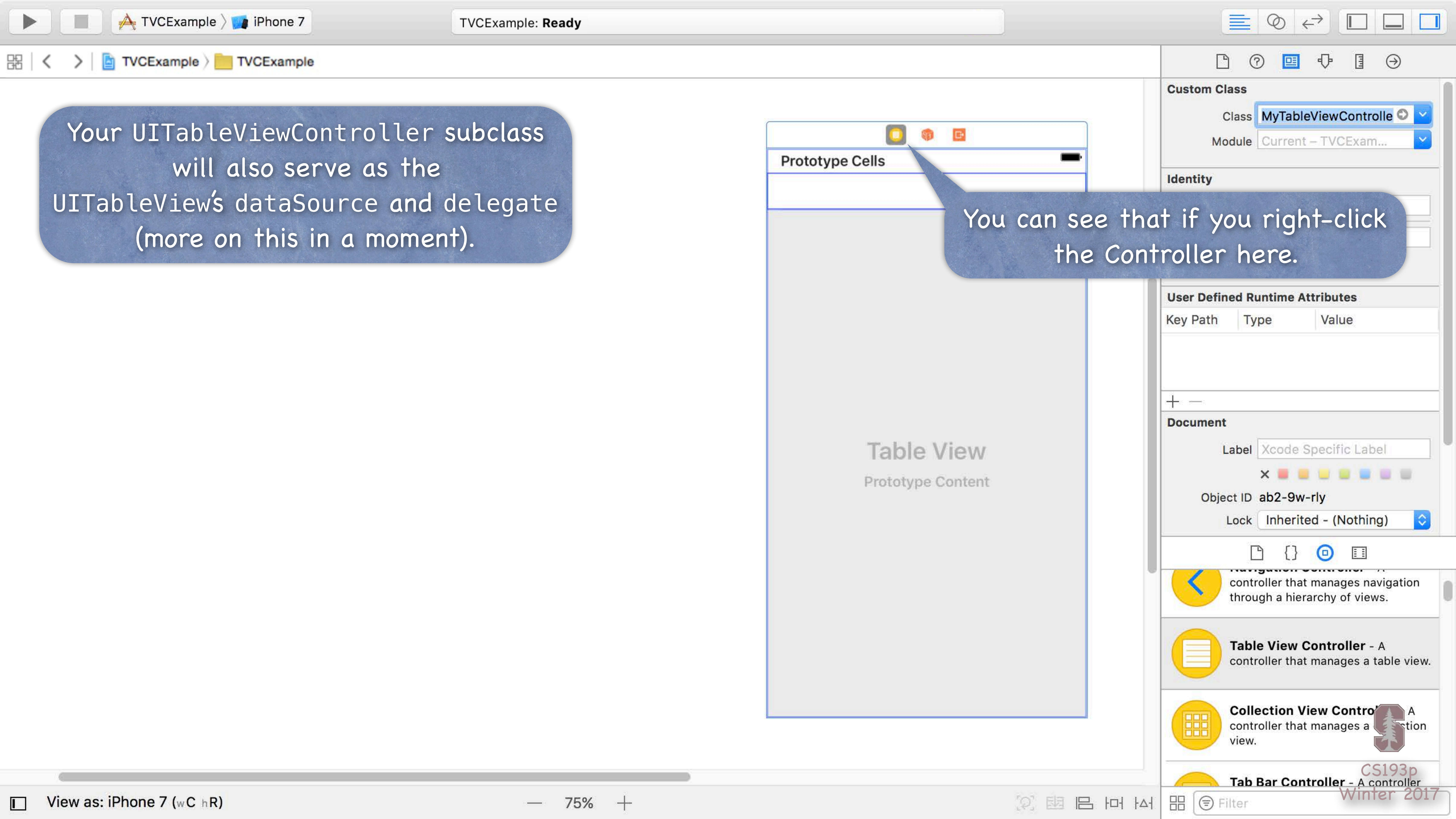
Table View

Prototype Content

View as: iPhone 7 (wC hR)

75%

Filter



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.

TVCExample: **Ready**

Custom Class

Class: MyTableViewControlle

Module: Current – TVCExam...

Identity

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.

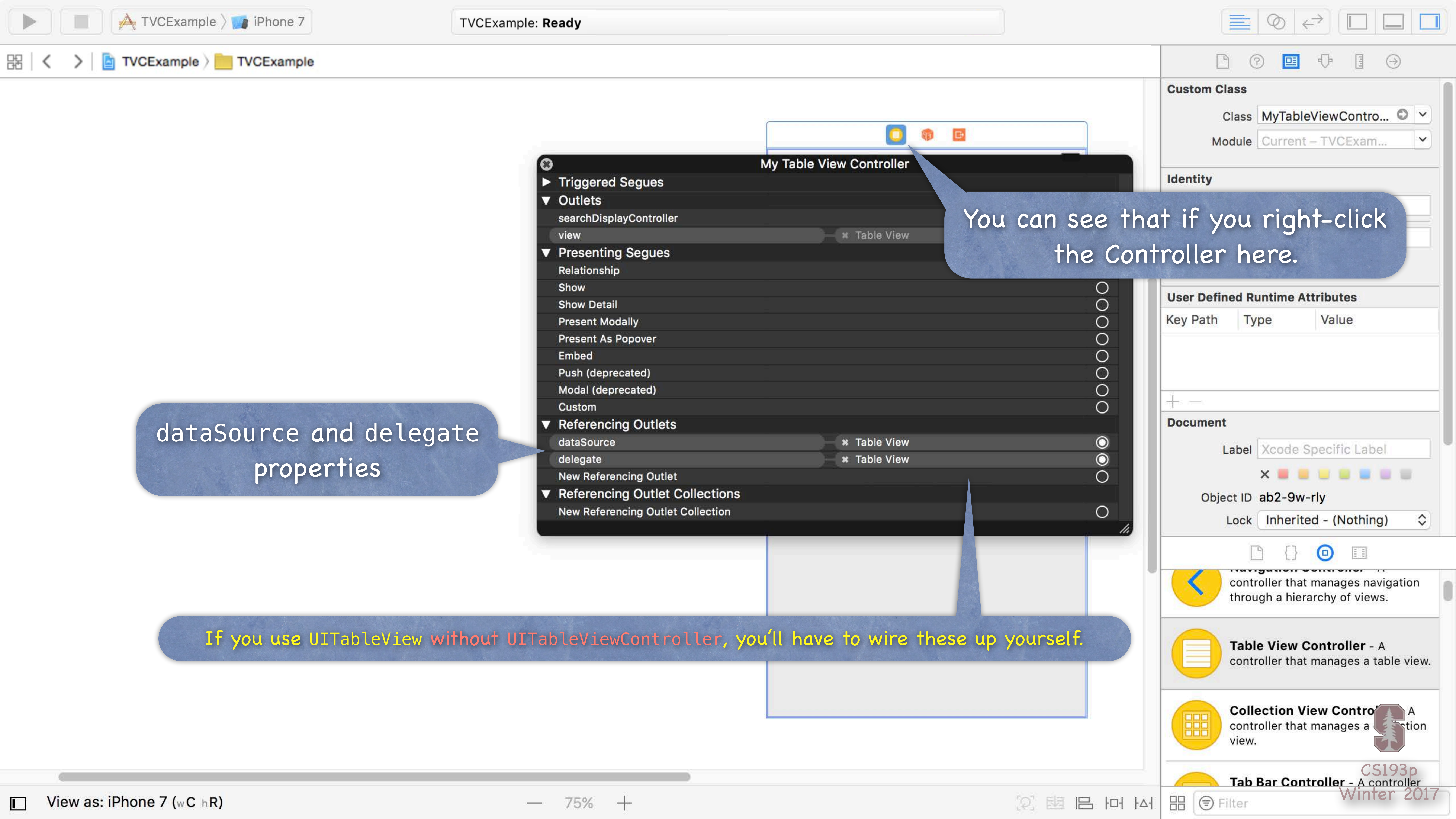
Table View Controller - A controller that manages a table view.

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

Tab Bar Controller - A controller

View as: iPhone 7 (wC hR)

75%



TVCExample > iPhone 7   TVCExample: **Ready**

TVCExample > TVCExample

My Table View Controller

- ▶ Triggered Segues
- ▼ Outlets
  - searchDisplayController
  - view — Table View
- ▼ Presenting Segues
  - Relationship
  - Show
  - Show Detail
  - Present Modally
  - Present As Popover
  - Embed
  - Push (deprecated)
  - Modal (deprecated)
  - Custom
- ▼ Referencing Outlets
  - dataSource — Table View
  - delegate — Table View
  - New Referencing Outlet
- ▼ Referencing Outlet Collections
  - New Referencing Outlet Collection

Custom Class

Class: MyTableViewContro...
Module: Current – TVCExam...

Identity

User Defined Runtime Attributes

Key Path	Type	Value

Document

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

controller that manages navigation through a hierarchy of views.

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

Collection View Contro... A controller that manages a ...tion view.

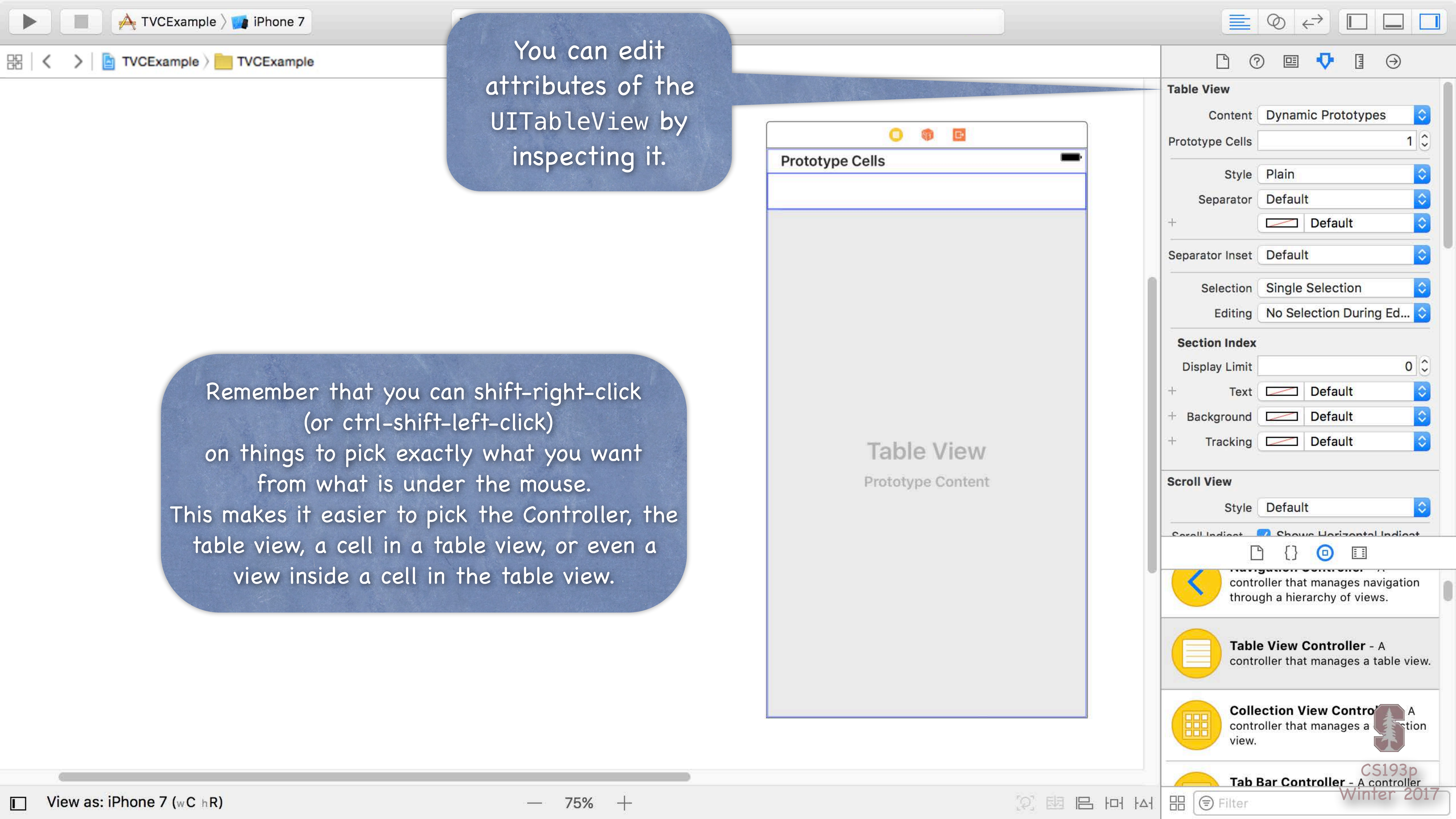
Tab Bar Controller - A controller

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.

TVCExample 〉 iPhone 7

TVCExample 〉 TVCExample

Prototype Cells

Table View
Prototype Content

Table View

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

Section Index

Display Limit: 0
Text: Default
Background: Default
Tracking: Default

Scroll View

Style: Default

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

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

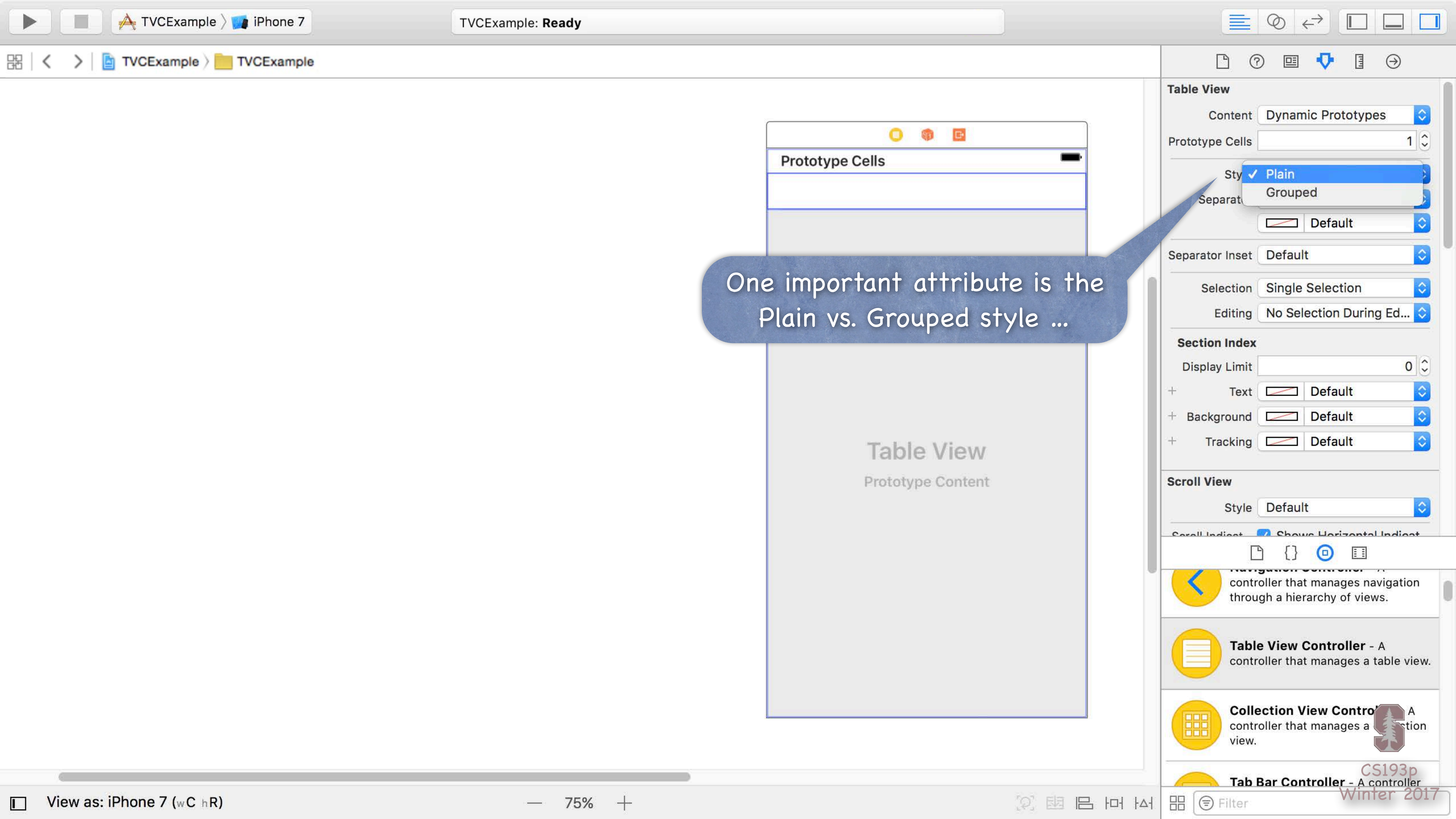
Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller

View as: iPhone 7 (wC hR)

75%

Filter



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

TVCExample > iPhone 7
TVCExample: **Ready**
TVCExample > TVCExample

Prototype Cells

Table View
Prototype Content

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

Scroll View

Style: 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

View as: iPhone 7 (wC hR)   75%

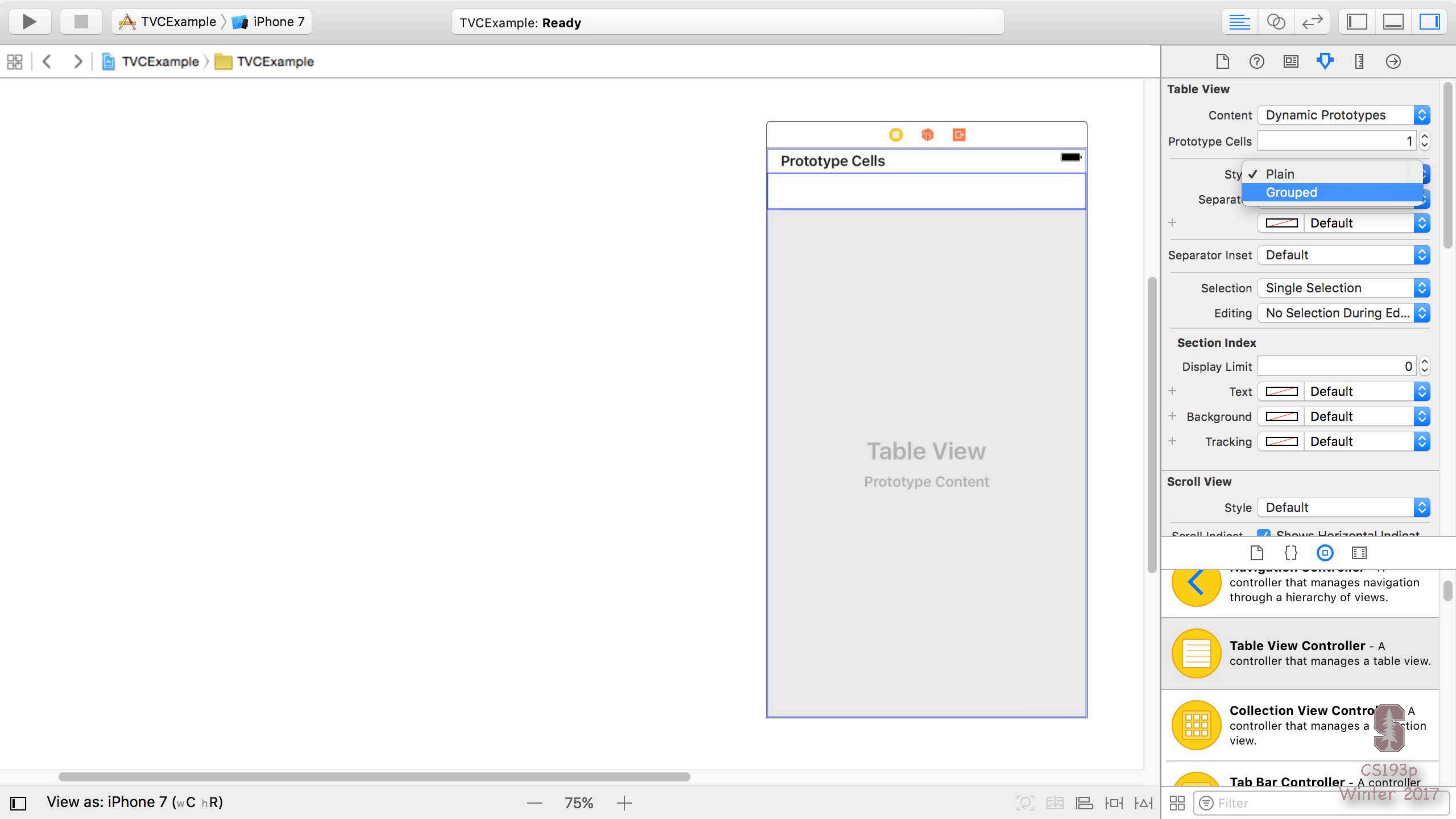


Table View

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

Scroll View

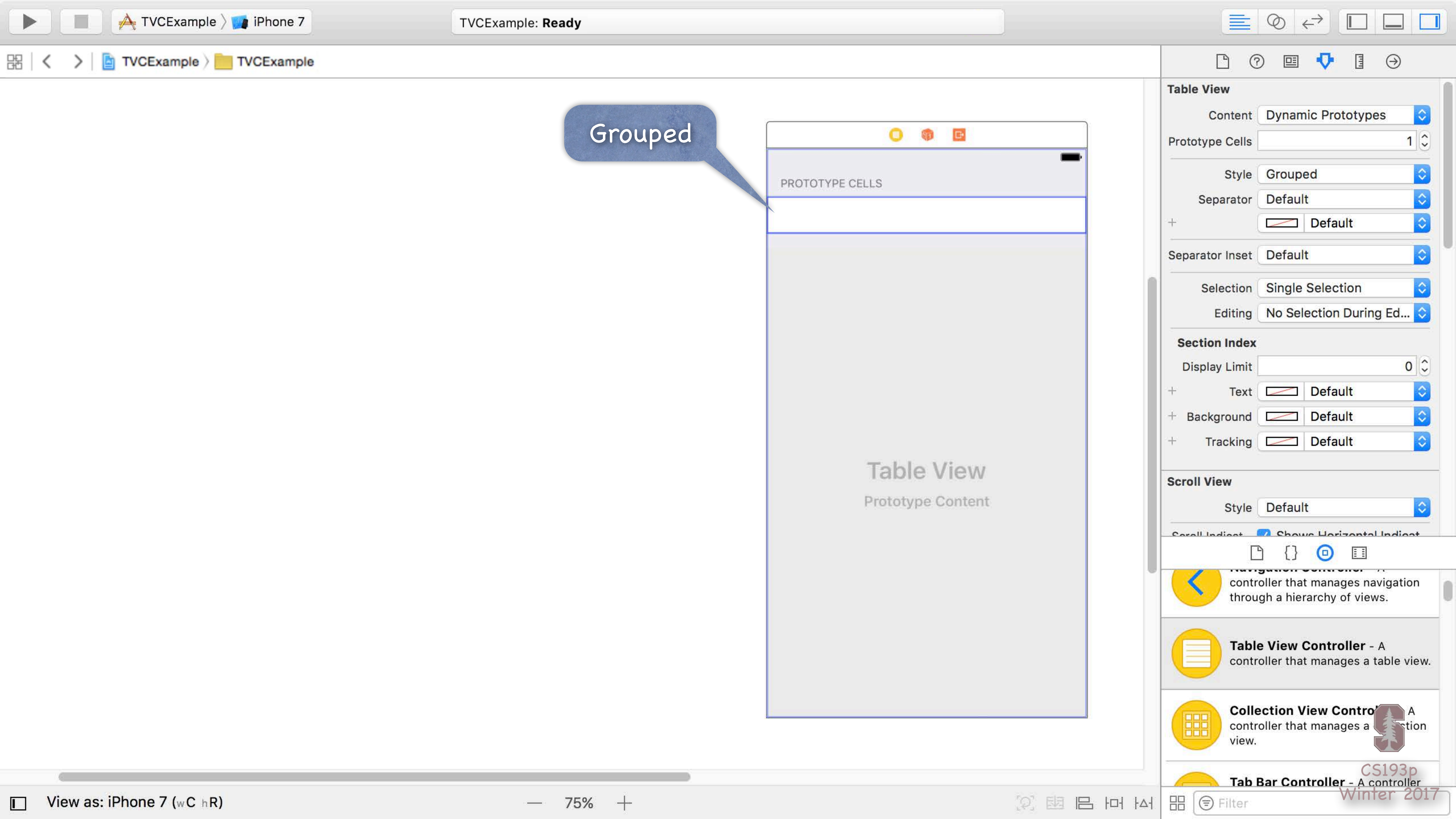
Style: Default
Scroll Indicat... ✓ Shows Horizontal Indicat...

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



TVCExample > iPhone 7

TVCExample: **Ready**

TVCExample > TVCExample

PROTOTYPE CELLS

Table View

Prototype Content

Table View

Content: Dynamic Prototypes

Prototype Cells: 1

Style: Grouped

Separator: Default

Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

controller that manages navigation through a hierarchy of views.

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

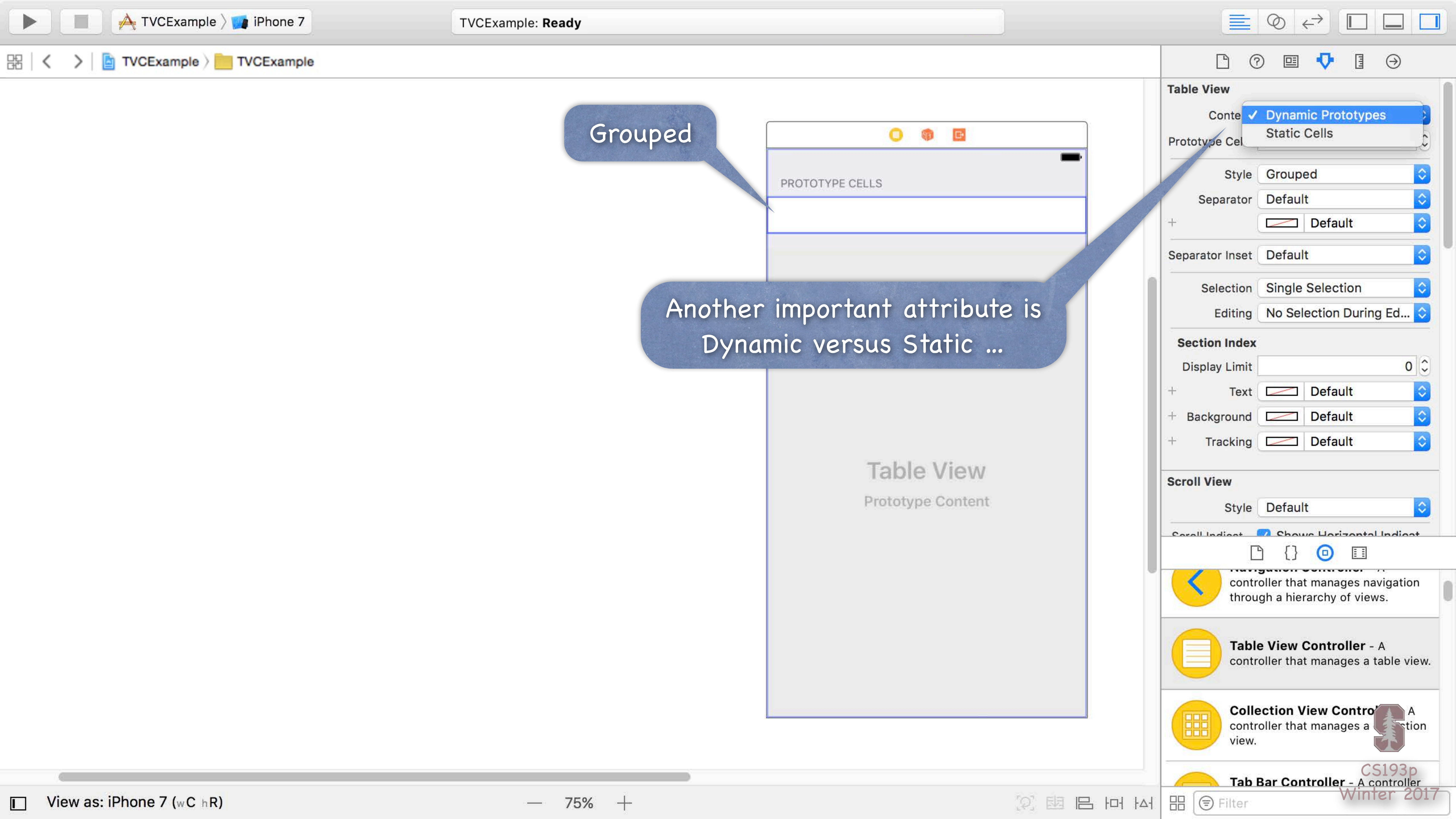
Collection View Contro... A controller that manages a ...tion view.

Tab Bar Controller - A controller

View as: iPhone 7 (wC hR)

75%

Filter



Grouped

Another important attribute is Dynamic versus Static …

TVCExample: **Ready**

Table View

Content: ✓ Dynamic Prototypes / Static Cells

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

Section Index

Display Limit	0
Text	Default
Background	Default
Tracking	Default

Scroll View

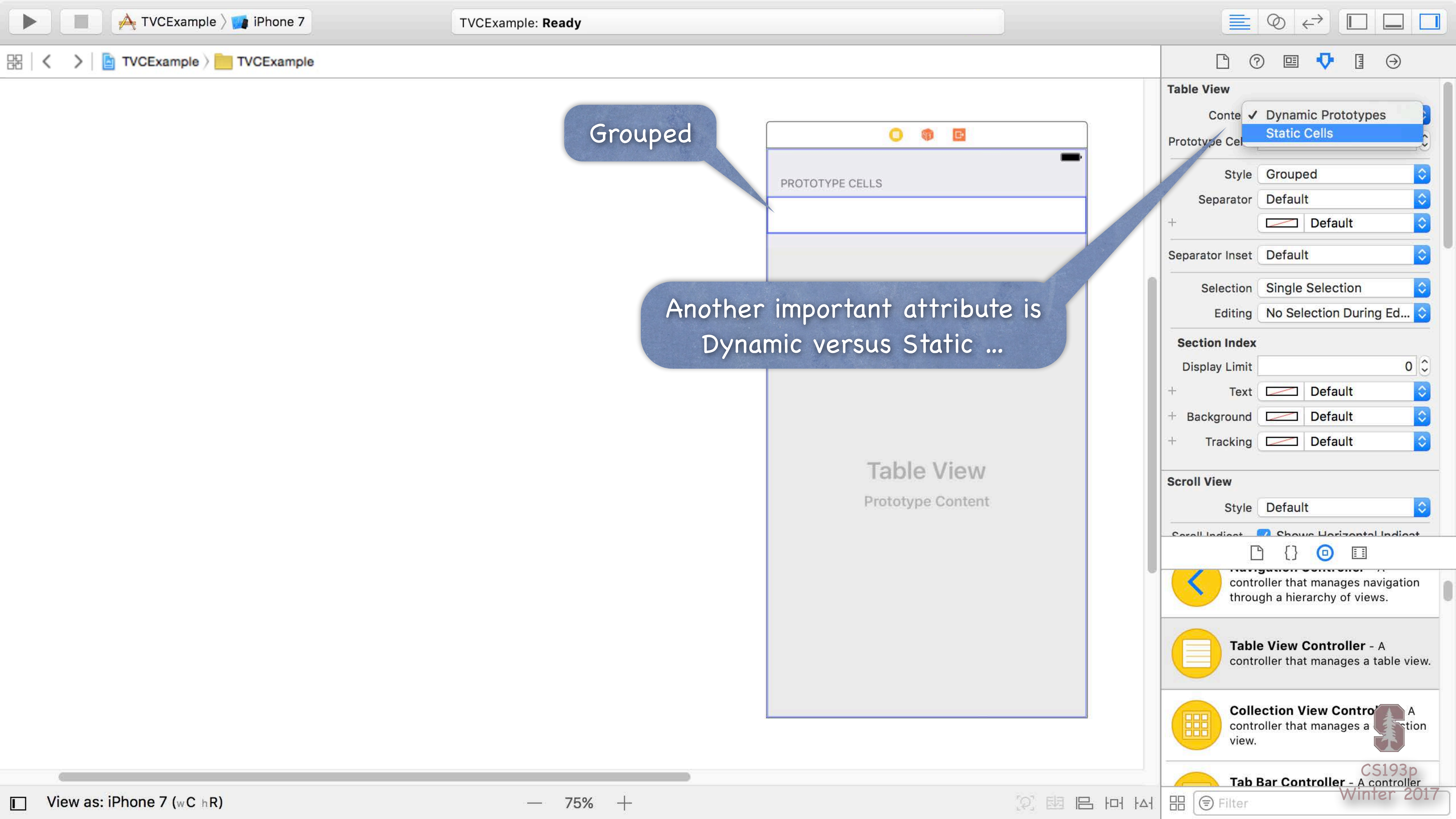
Style	Default

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



Grouped

Another important attribute is Dynamic versus Static …

TVCExample > iPhone 7

TVCExample: **Ready**

TVCExample > TVCExample

PROTOTYPE CELLS

Table View

Prototype Content

Table View

Content: ✓ Dynamic Prototypes / Static Cells

Prototype Cells

Style: Grouped

Separator: Default

Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

Scroll Indicat: Shows Horizontal Indicat

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

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

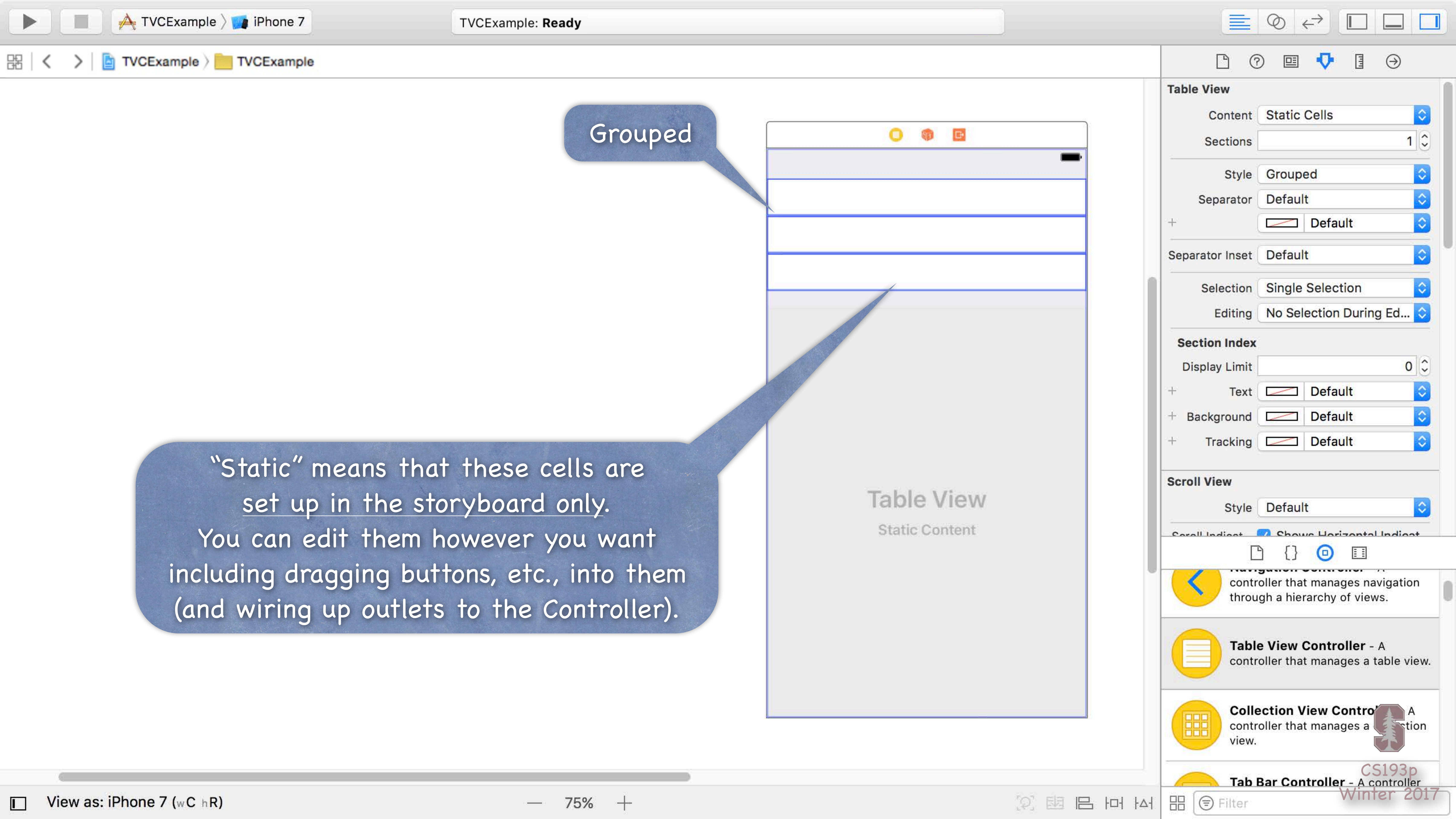
Collection View Controller - A controller that manages a collection view.

Tab Bar Controller - A controller

View as: iPhone 7 (wC hR)

75%

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).

Table View

Content	Static Cells
Sections	1
Style	Grouped
Separator	Default
	Default
Separator Inset	Default
Selection	Single Selection
Editing	No Selection During Ed...

Section Index

Display Limit	0
Text	Default
Background	Default
Tracking	Default

Scroll View

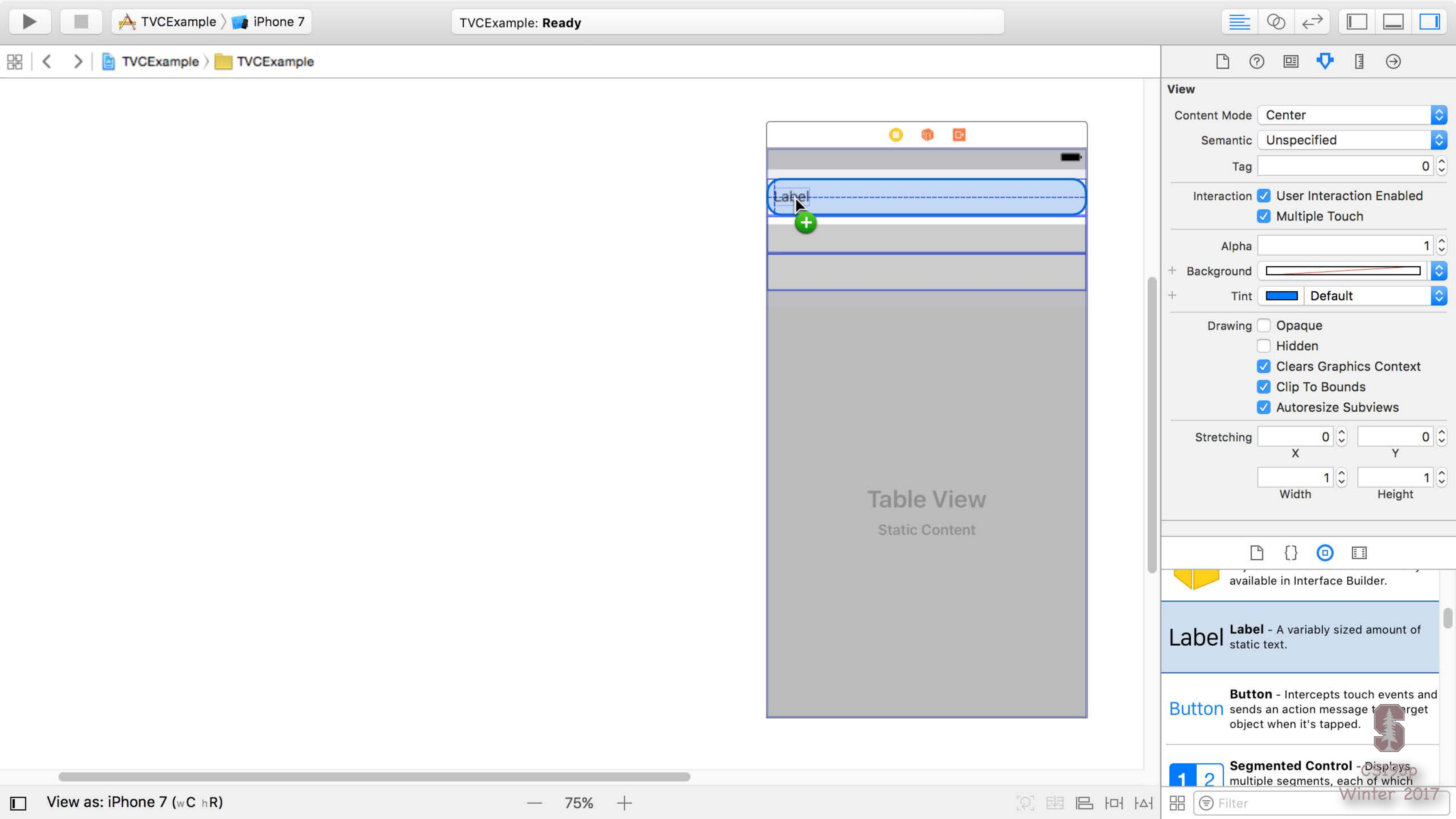
Style	Default

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



TVCExample: **Ready**

View

Content Mode: Center

Semantic: Unspecified

Tag: 0

Interaction: ✓ User Interaction Enabled, ✓ Multiple Touch

Alpha: 1

Background

Tint: Default

Drawing: Opaque, Hidden, ✓ Clears Graphics Context, ✓ Clip To Bounds, ✓ Autoresize Subviews

Stretching: X 0, Y 0, Width 1, Height 1

available in Interface Builder.

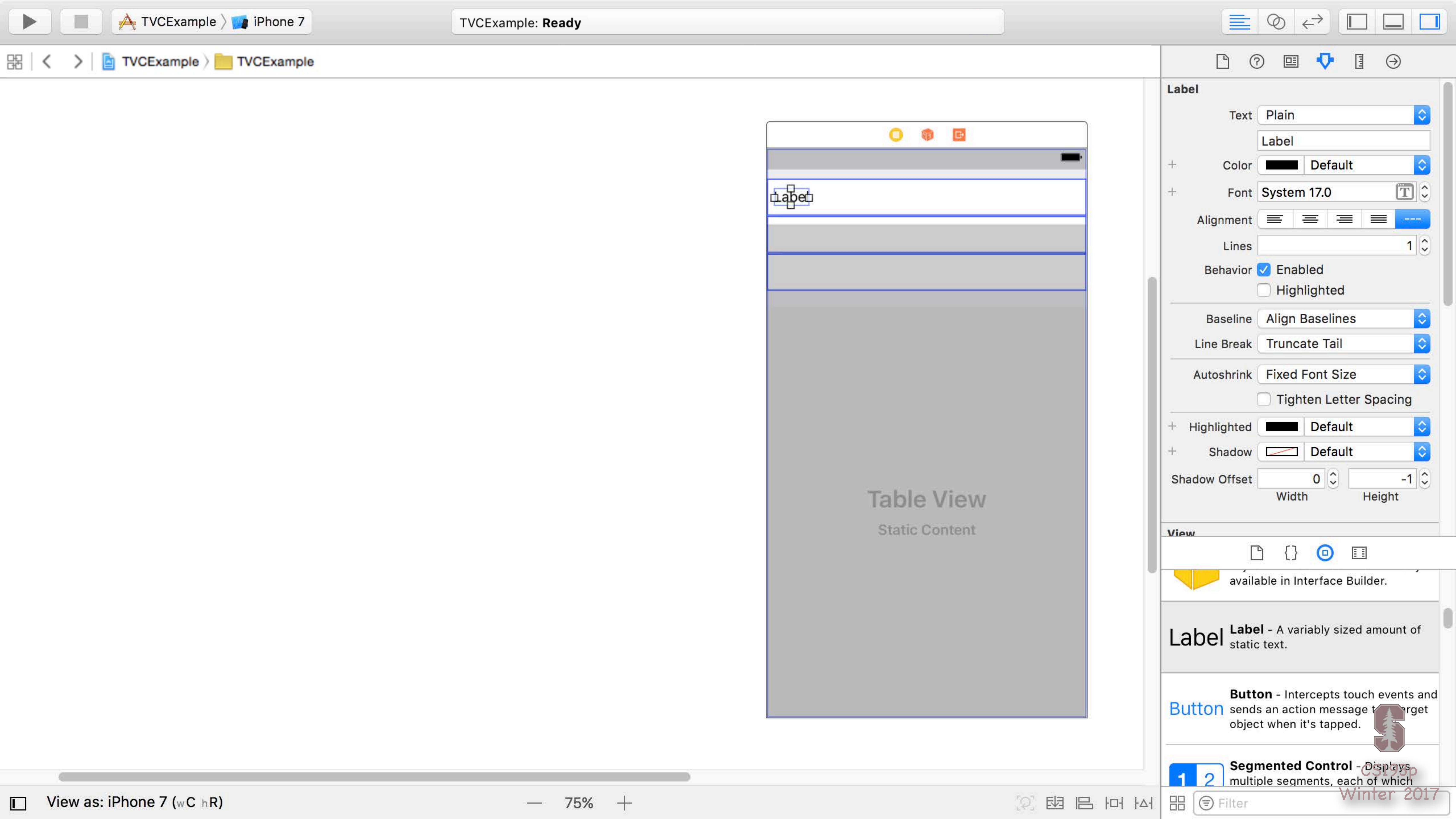
Label - A variably sized amount of static text.

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

View as: iPhone 7 (wC hR)

75%



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 (Width) -1 (Height)

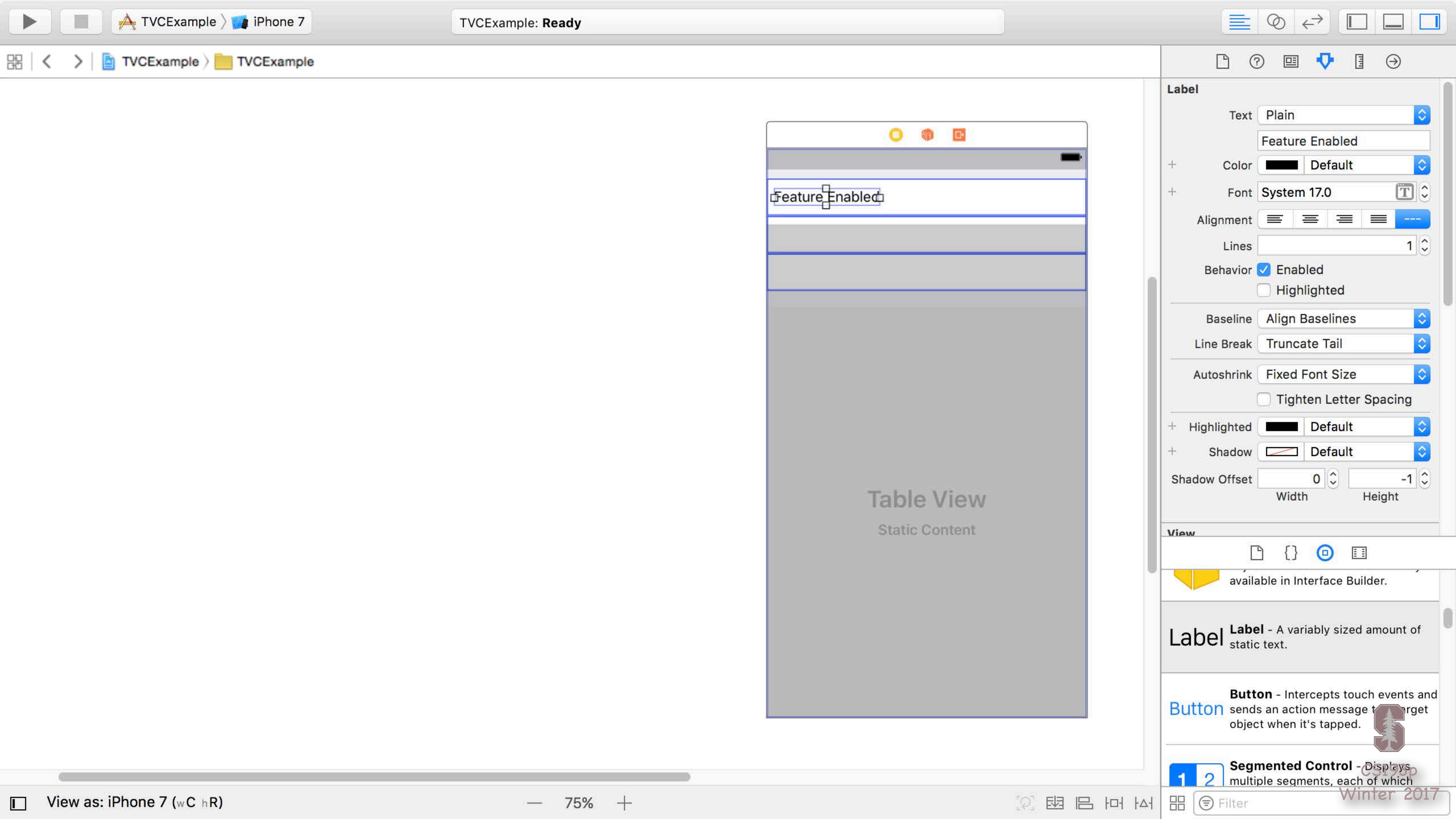
View

available in Interface Builder.

Label - A variably sized amount of 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



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Feature Enabled

Table View

Static Content

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 Width, -1 Height

View

available in Interface Builder.

Label **Label** - A variably sized amount of static text.

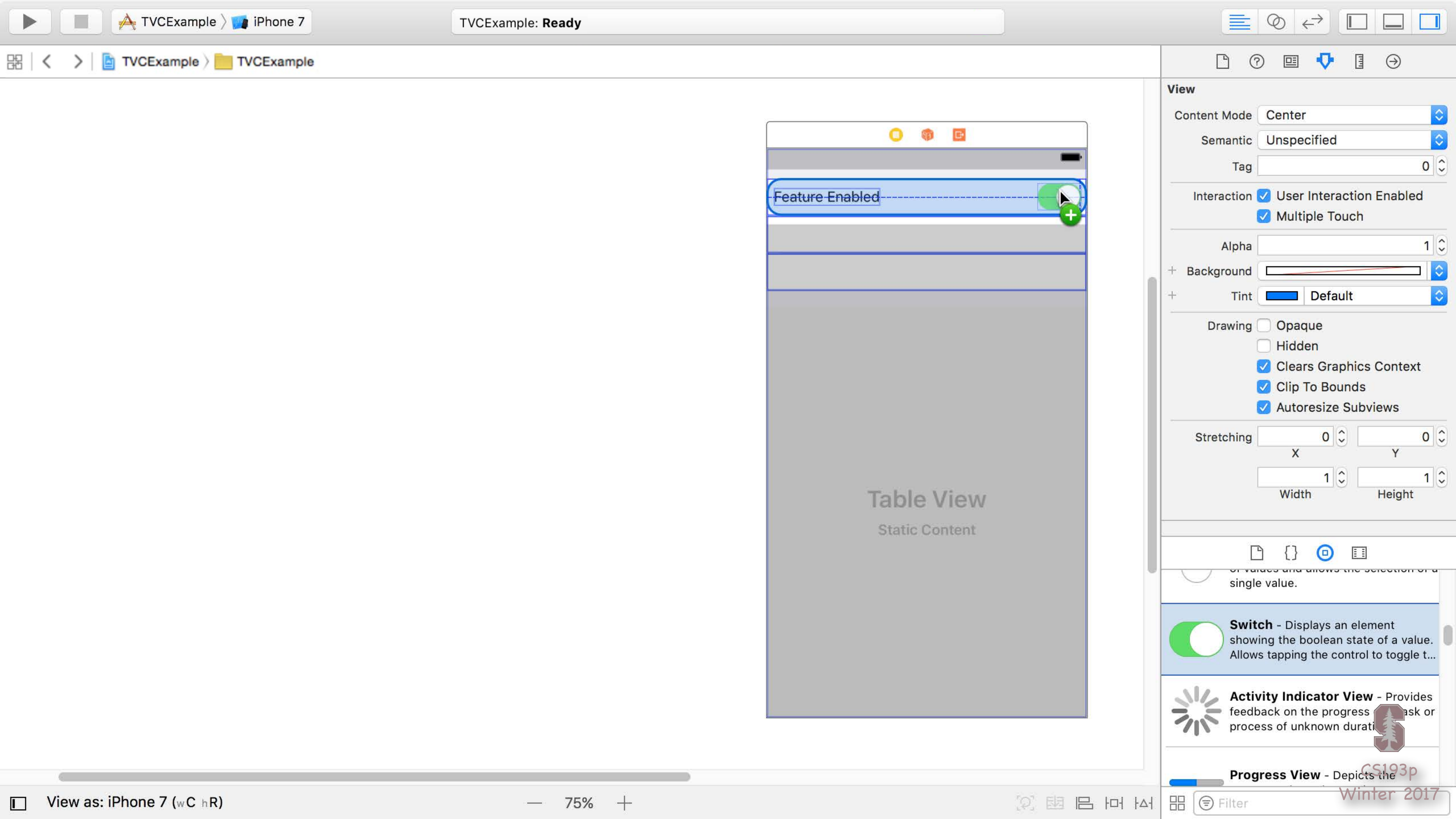
Button **Button** - Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control - Displays multiple segments, each of which

View as: iPhone 7 (wC hR)

75%

Filter



TVCExample › iPhone 7

TVCExample: **Ready**

TVCExample › TVCExample

Feature Enabled

Table View

Static Content

View

Content Mode: Center

Semantic: Unspecified

Tag: 0

Interaction: ✓ User Interaction Enabled

✓ Multiple Touch

Alpha: 1

Background

Tint: Default

Drawing: ☐ Opaque

☐ Hidden

✓ Clears Graphics Context

✓ Clip To Bounds

✓ Autoresize Subviews

Stretching: X 0, Y 0, Width 1, Height 1

single value.

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

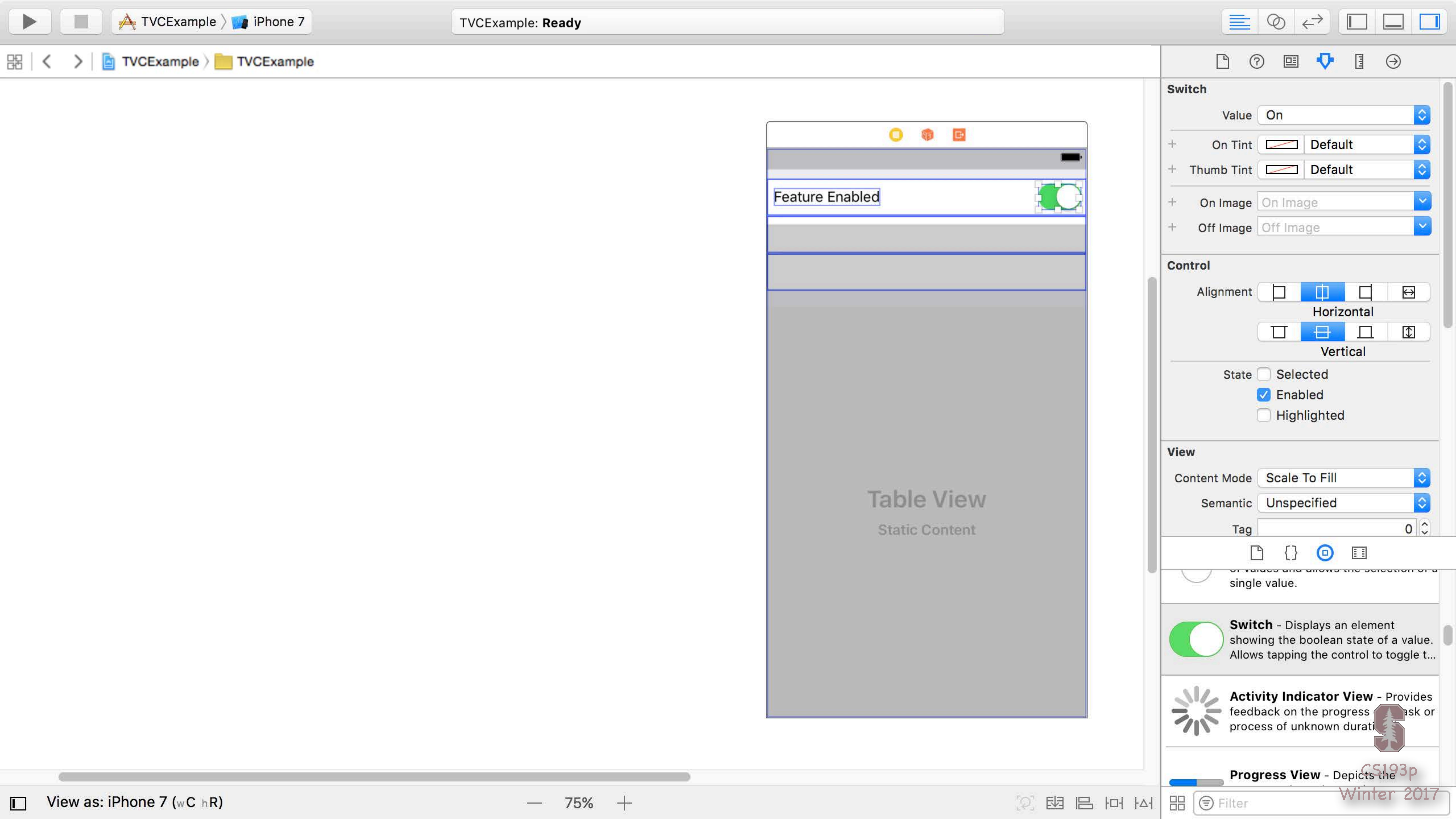
Activity Indicator View - Provides feedback on the progress ... ask or process of unknown durati...

Progress View - Depicts the

View as: iPhone 7 (wC hR)

75%

Filter



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

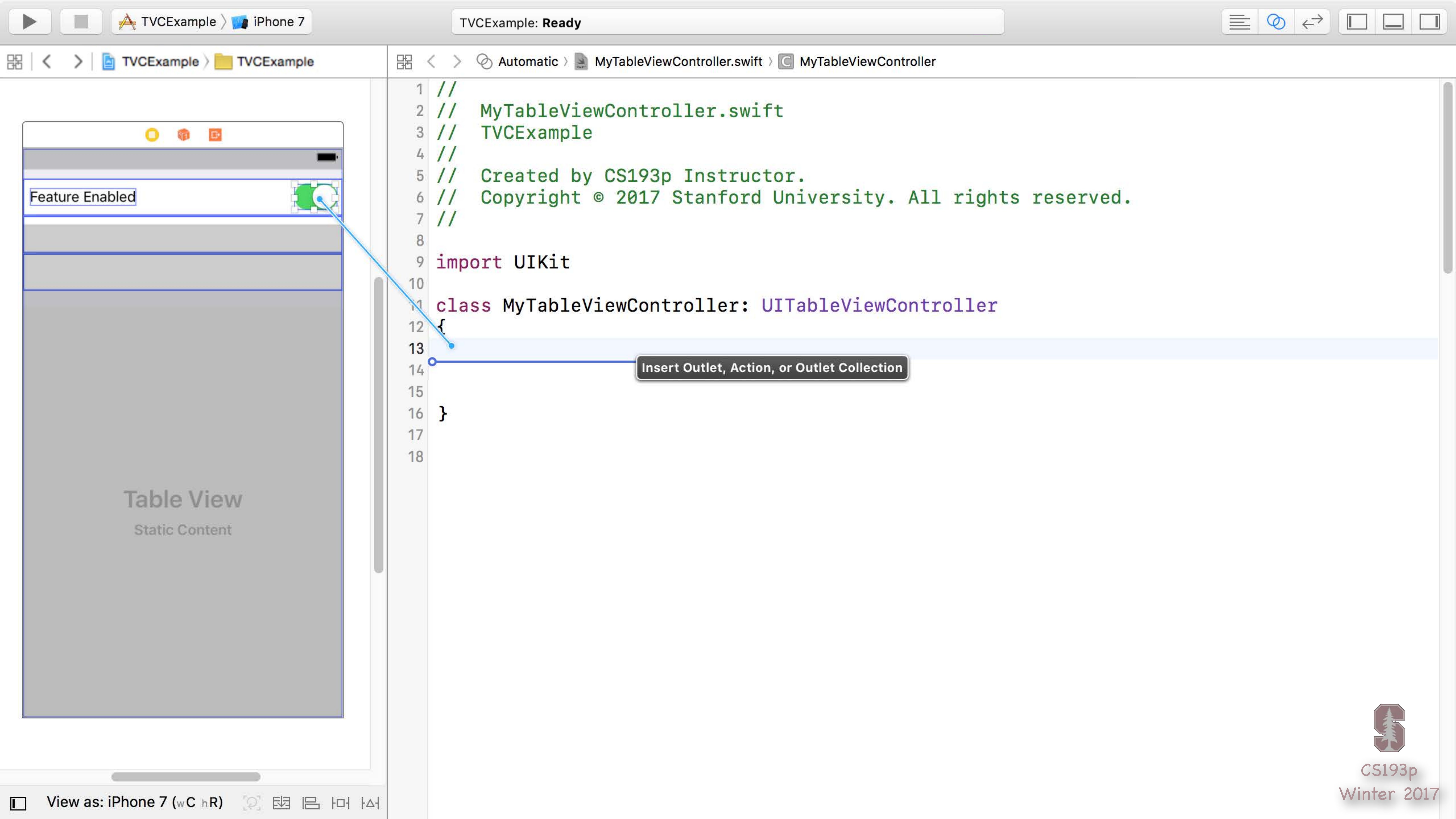
Tag: 0

single value.

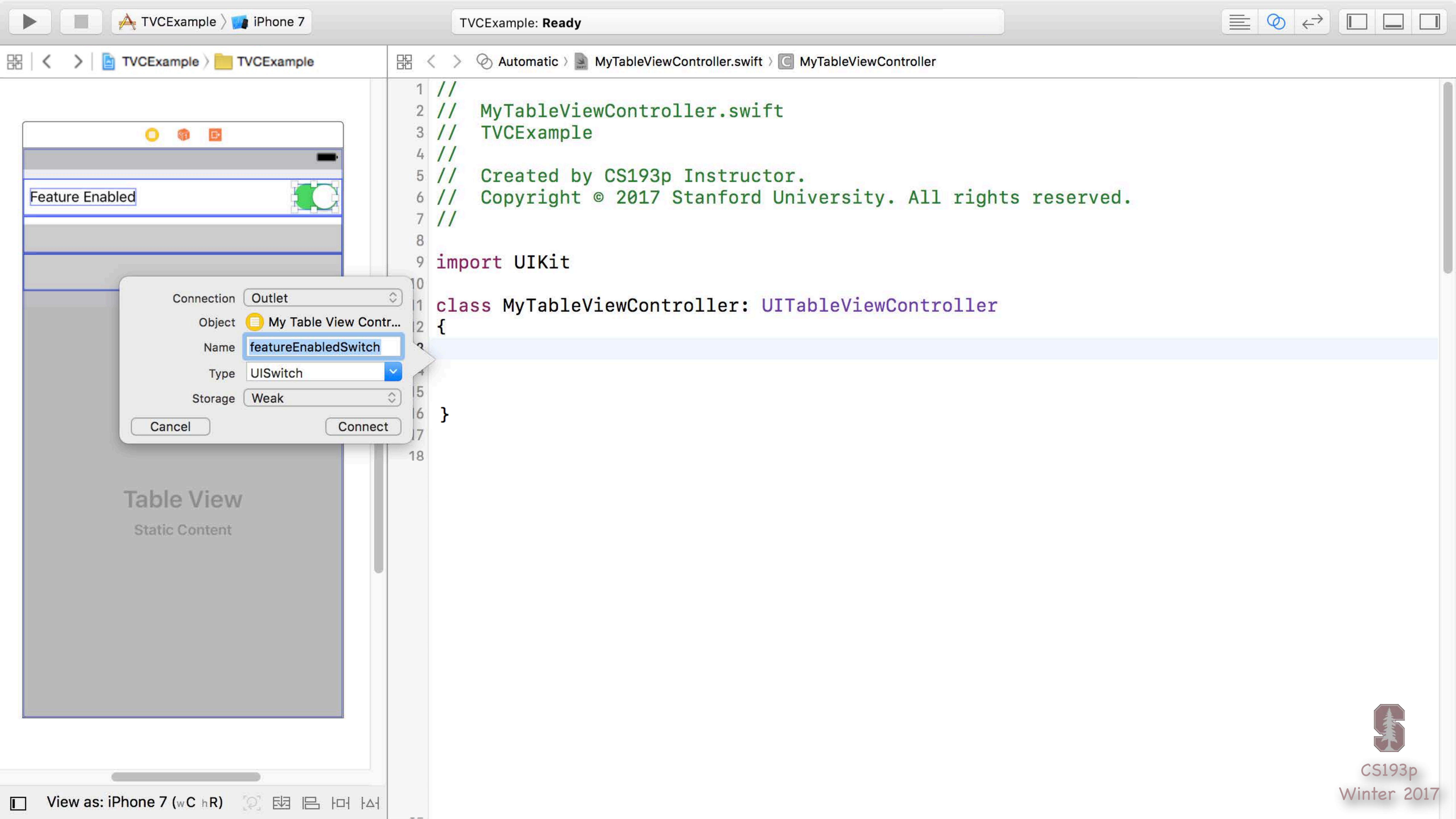
Switch - Displays an element showing the boolean state of a value. Allows tapping the control to toggle t...

Activity Indicator View - Provides feedback on the progress ... ask or process of unknown durati...

Progress View - Depicts the

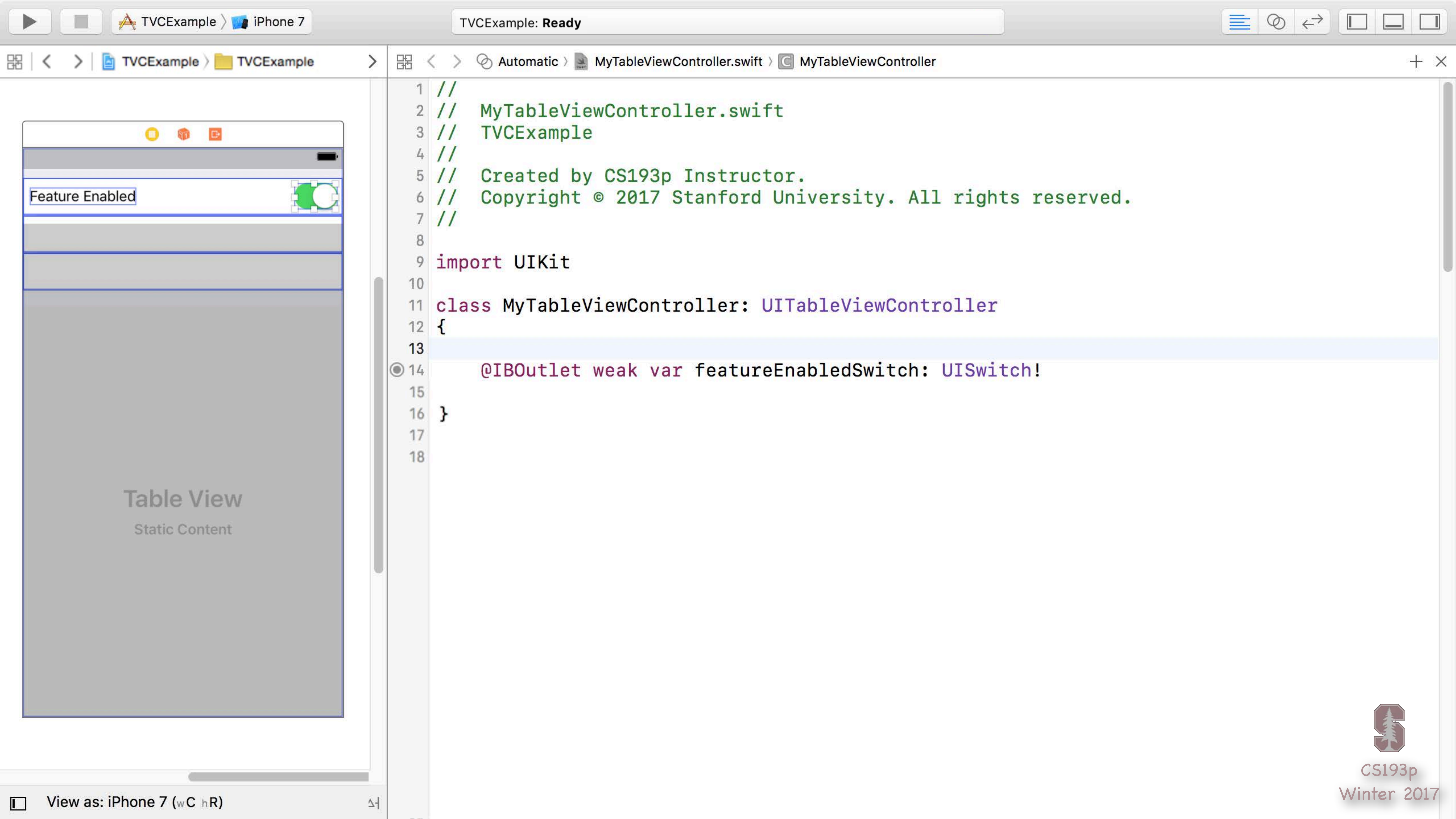


```
//
//  MyTableViewController.swift
//  TVCExample
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class MyTableViewController: UITableViewController
{



}
```

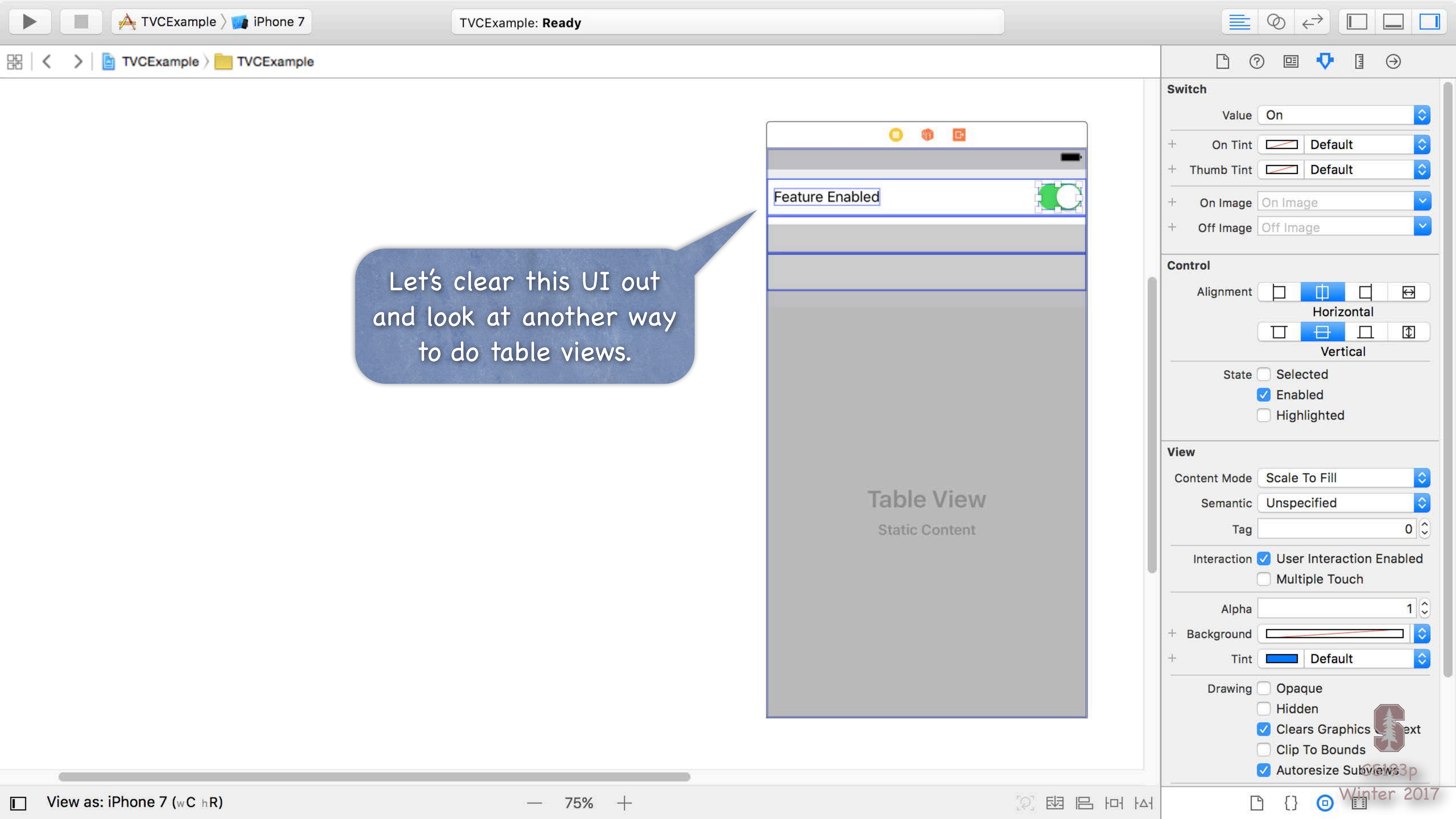


Connection: Outlet
Object: My Table View Contr...
Name: featureEnabledSwitch
Type: UISwitch
Storage: Weak

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

import UIKit

class MyTableViewController: UITableViewController
{


}
```



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

import UIKit

class MyTableViewController: UITableViewController
{

    @IBOutlet weak var featureEnabledSwitch: UISwitch!

}
```


TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Feature Enabled

Table View

Static Content

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

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

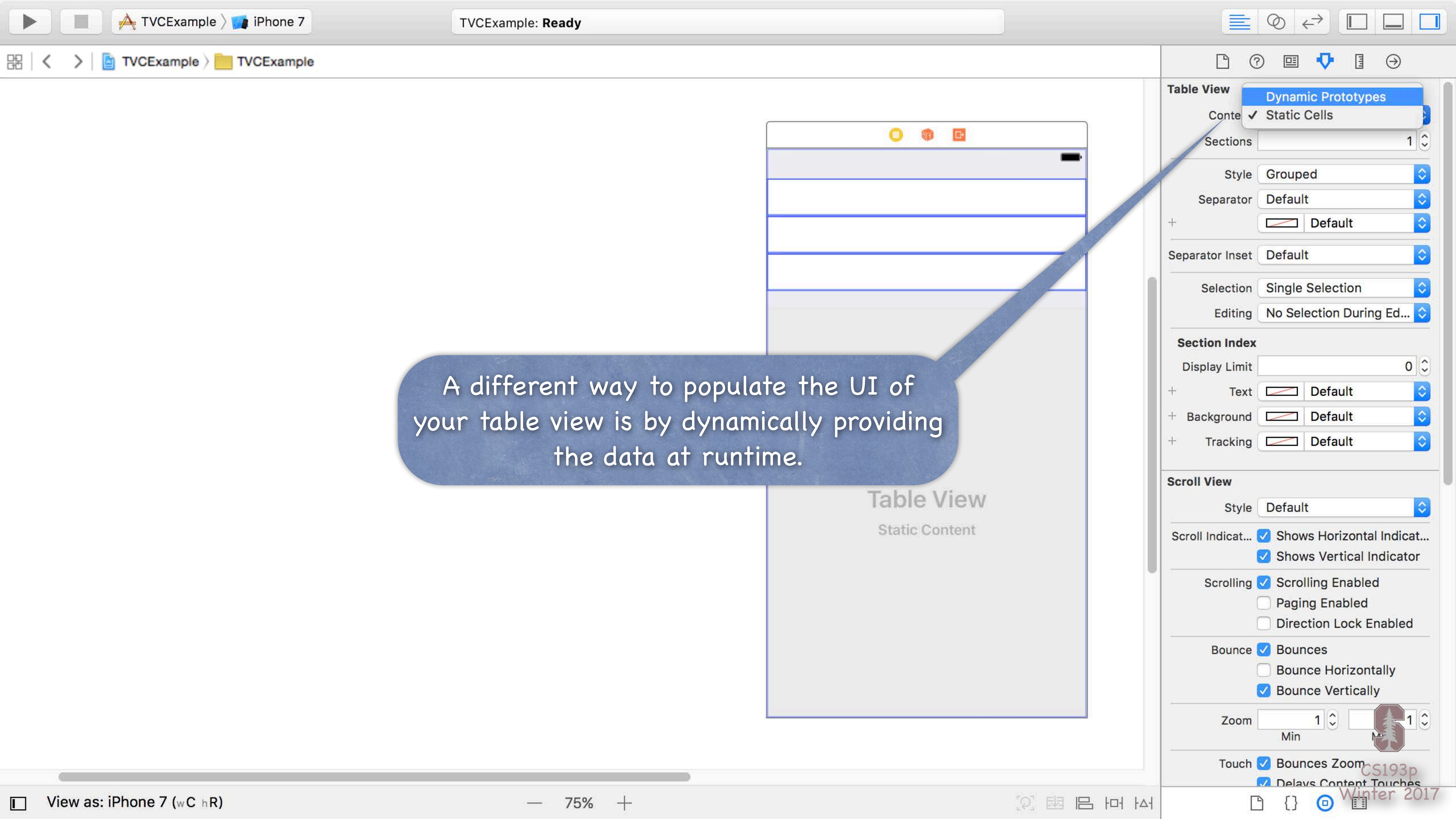
Background

Tint: Default

Drawing: ☐ Opaque, ☐ Hidden, ☑ Clears Graphics Context, ☐ Clip To Bounds, ☑ Autoresize Subviews

View as: iPhone 7 (wC hR)

75%

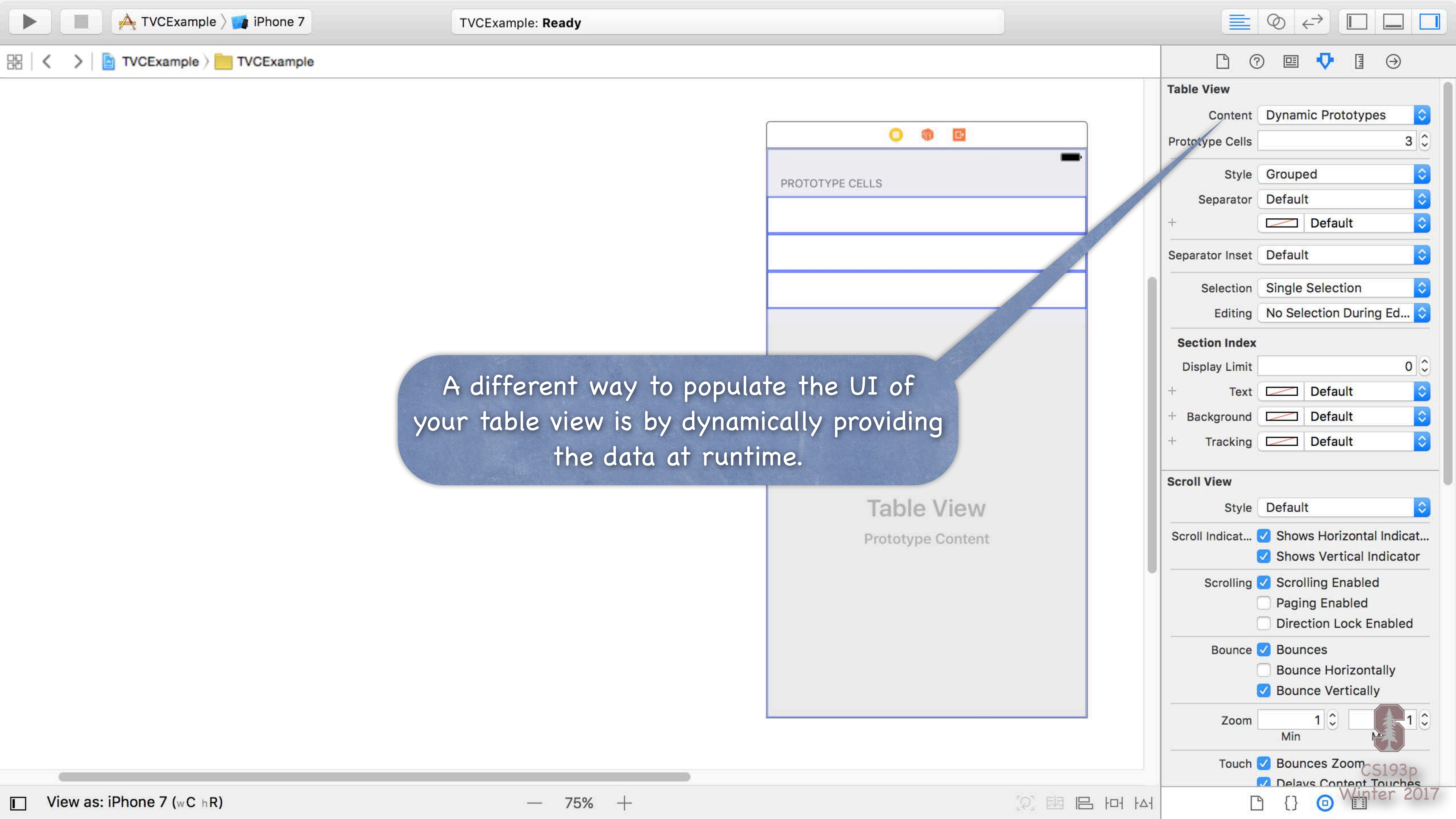


A different way to populate the UI of your table view is by dynamically providing the data at runtime.

Table View
Content: Dynamic Prototypes / ✓ Static Cells
Sections: 1
Style: Grouped
Separator: Default
Default
Separator Inset: Default
Selection: Single Selection
Editing: No Selection During Ed...

Section Index
Display Limit: 0
Text: Default
Background: Default
Tracking: Default

Scroll View
Style: Default
Scroll Indicat...: ✓ Shows Horizontal Indicat...
✓ Shows Vertical Indicator
Scrolling: ✓ Scrolling Enabled
Paging Enabled
Direction Lock Enabled
Bounce: ✓ Bounces
Bounce Horizontally
✓ Bounce Vertically
Zoom: 1 Min, 1 Max
Touch: ✓ Bounces Zoom
✓ Delays Content Touches



TVCExample › iPhone 7

TVCExample: **Ready**

TVCExample › TVCExample

PROTOTYPE CELLS

Table View

Prototype Content

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

Table View

Content	Dynamic Prototypes
Prototype Cells	3
Style	Grouped
Separator	Default
	Default
Separator Inset	Default
Selection	Single Selection
Editing	No Selection During Ed...

Section Index

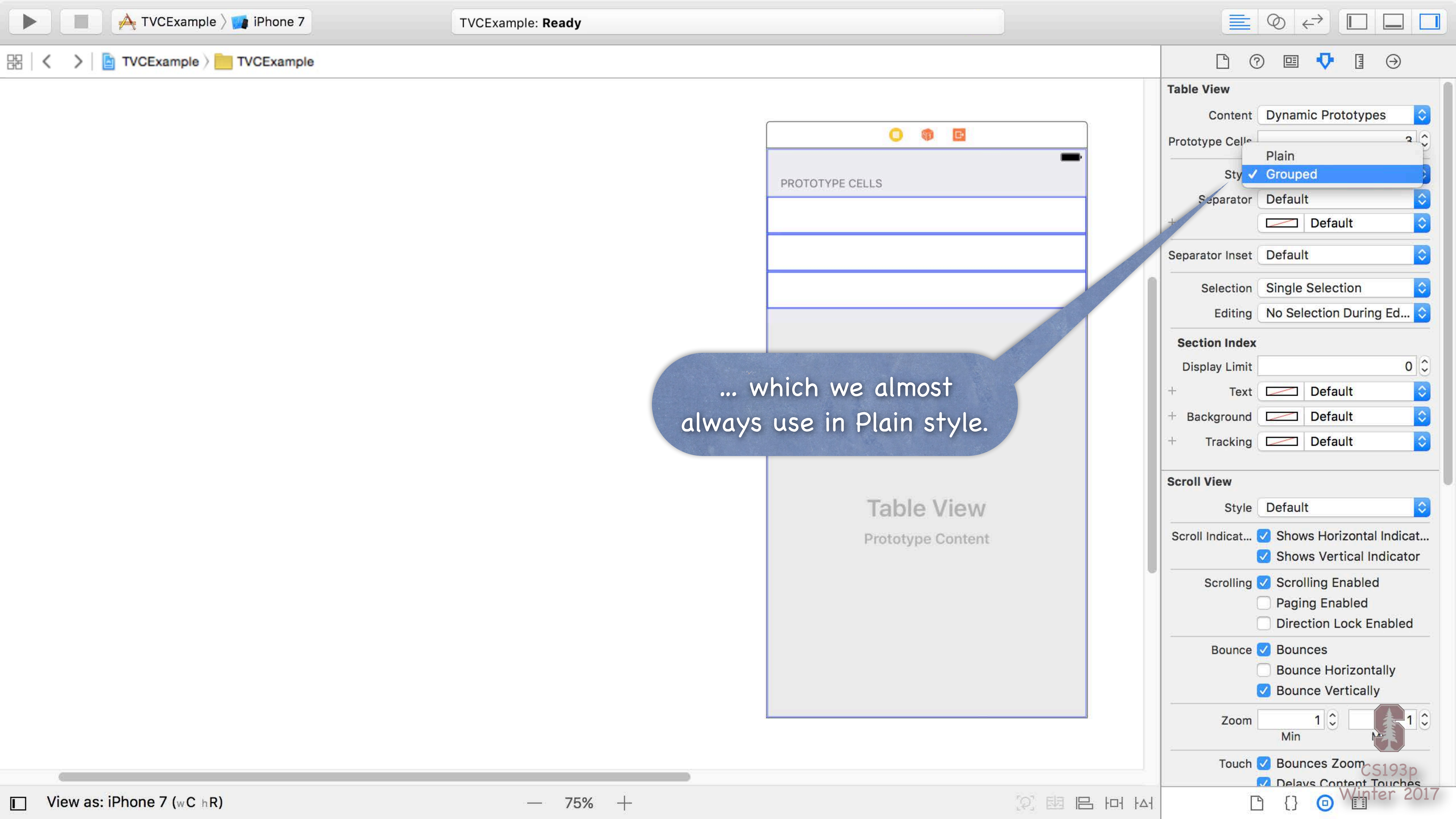
Display Limit	0
Text	Default
Background	Default
Tracking	Default

Scroll View

Style	Default
Scroll Indicat...	✓ Shows Horizontal Indicat...
	✓ Shows Vertical Indicator
Scrolling	✓ Scrolling Enabled
	Paging Enabled
	Direction Lock Enabled
Bounce	✓ Bounces
	Bounce Horizontally
	✓ Bounce Vertically
Zoom	1 Min, 1 Max
Touch	✓ Bounces Zoom
	✓ Delays Content Touches

View as: iPhone 7 (wC hR)

75%



TVCExample: **Ready**

TVCExample 〉 TVCExample

Table View

Content	Dynamic Prototypes
Prototype Cells	3
Style	Plain / ✓ Grouped
Separator	Default
	Default
Separator Inset	Default
Selection	Single Selection
Editing	No Selection During Ed...

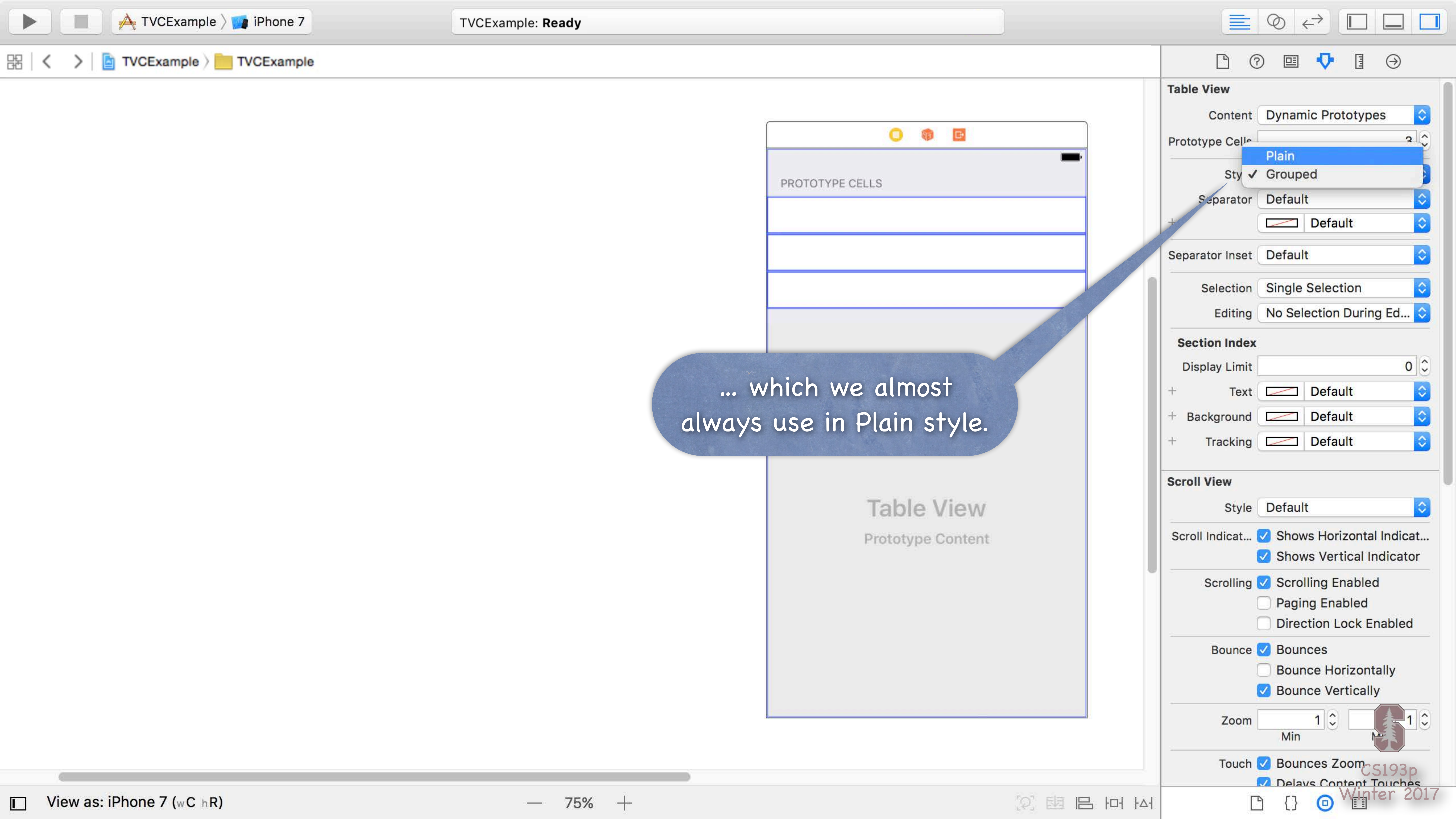
Section Index

Display Limit	0
Text	Default
Background	Default
Tracking	Default

Scroll View

Style	Default
Scroll Indicat...	✓ Shows Horizontal Indicat...
	✓ Shows Vertical Indicator
Scrolling	✓ Scrolling Enabled
	☐ Paging Enabled
	☐ Direction Lock Enabled
Bounce	✓ Bounces
	☐ Bounce Horizontally
	✓ Bounce Vertically
Zoom	1 (Min) 1 (Max)
Touch	✓ Bounces Zoom
	✓ Delays Content Touches

… which we almost always use in Plain style.



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

PROTOTYPE CELLS

Table View

Prototype Content

… which we almost always use in Plain style.

Table View

Content: Dynamic Prototypes

Prototype Cells: 3

Style: Plain / ✓ Grouped

Separator: Default

Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

Section Index

Display Limit: 0

Text: Default

Background: Default

Tracking: Default

Scroll View

Style: Default

Scroll Indicat...: ✓ Shows Horizontal Indicat...

✓ Shows Vertical Indicator

Scrolling: ✓ Scrolling Enabled

Paging Enabled

Direction Lock Enabled

Bounce: ✓ Bounces

Bounce Horizontally

✓ Bounce Vertically

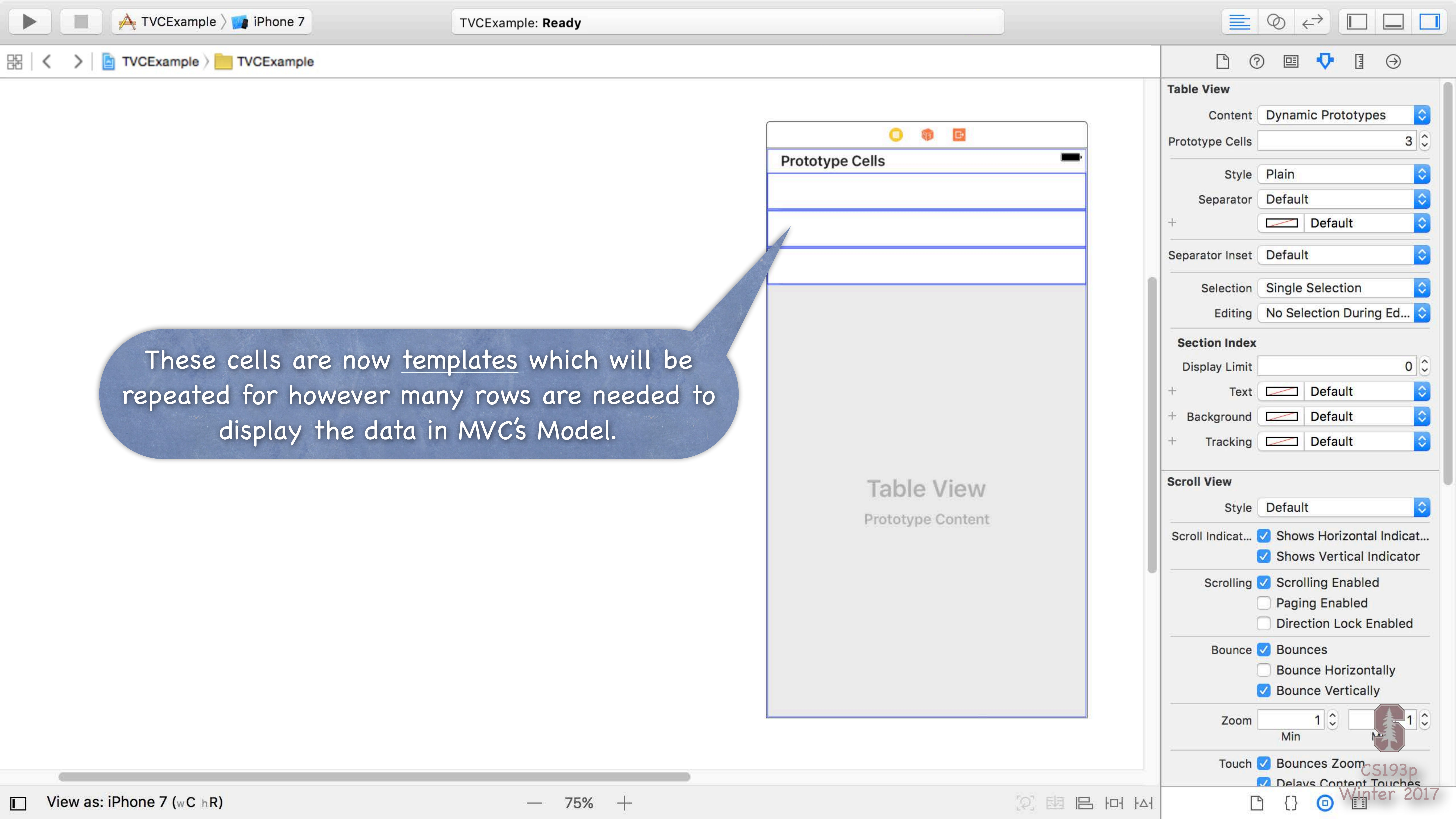
Zoom: 1 Min, 1 Max

Touch: ✓ Bounces Zoom

✓ Delays Content Touches

View as: iPhone 7 (wC hR)

75%



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

Table View

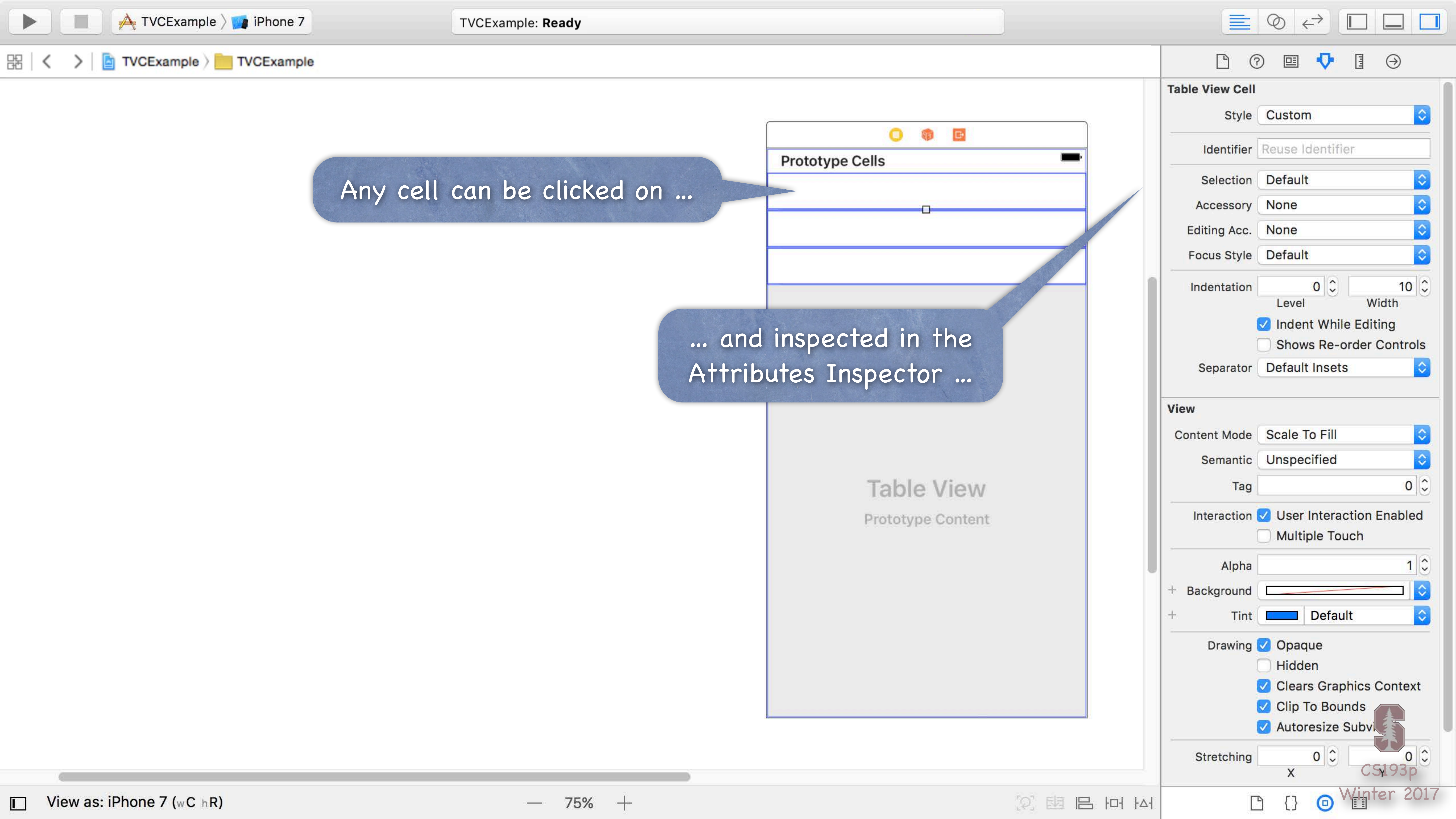
Content	Dynamic Prototypes
Prototype Cells	3
Style	Plain
Separator	Default
	Default
Separator Inset	Default
Selection	Single Selection
Editing	No Selection During Ed...

Section Index

Display Limit	0
Text	Default
Background	Default
Tracking	Default

Scroll View

Style	Default
Scroll Indicat...	✓ Shows Horizontal Indicat...
	✓ 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



Any cell can be clicked on …

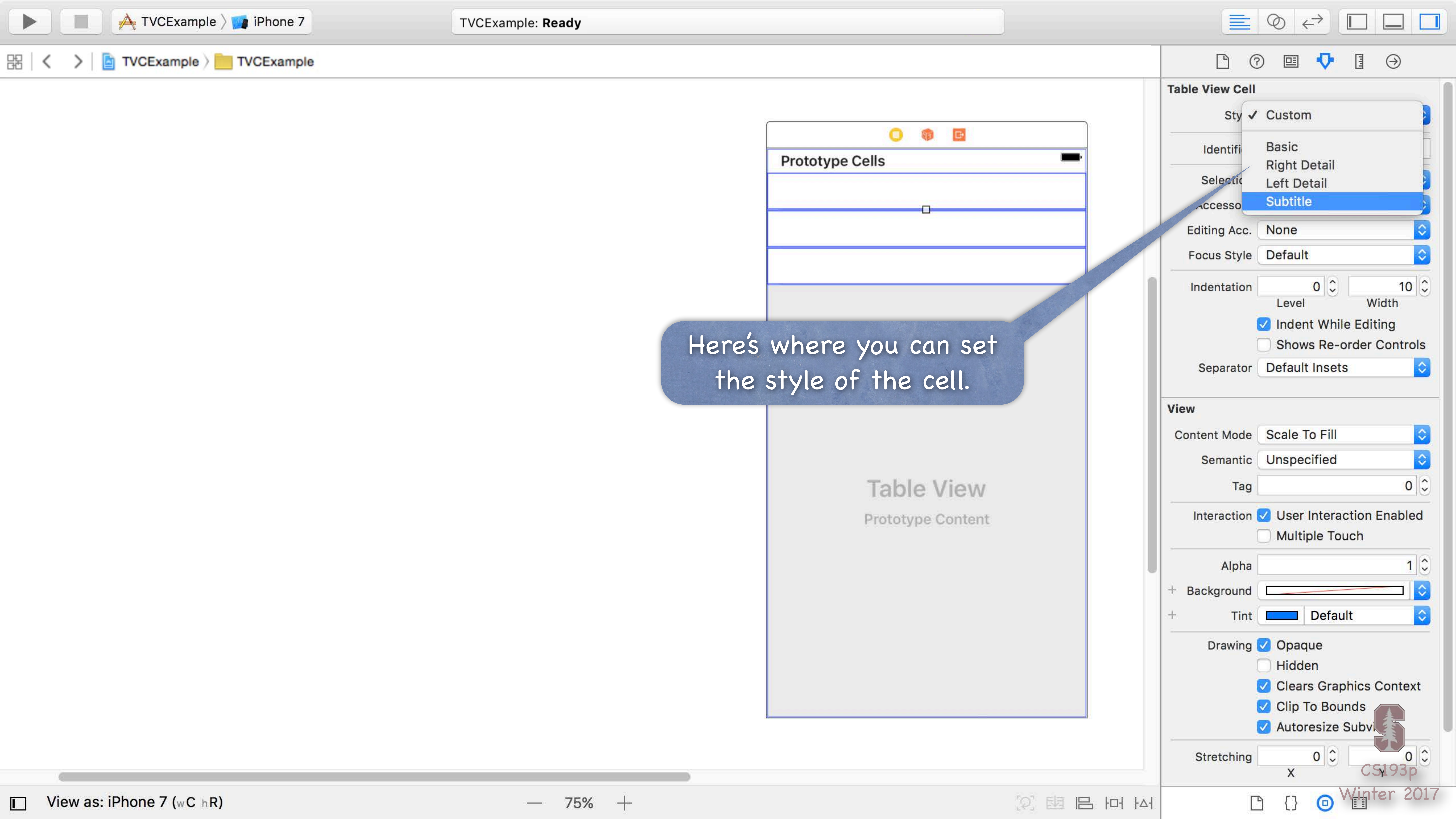
… and inspected in the Attributes Inspector …

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	
Tint	Default
Drawing	☑ Opaque
	☐ Hidden
	☑ Clears Graphics Context
	☑ Clip To Bounds
	☑ Autoresize Subvi
Stretching	0 X, 0 Y



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

Table View Cell

Style: ✓ Custom

- Basic
- Right Detail
- Left Detail
- Subtitle

Identifier

Selection

Accessory

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

Tint: Default

Drawing: [✓] Opaque

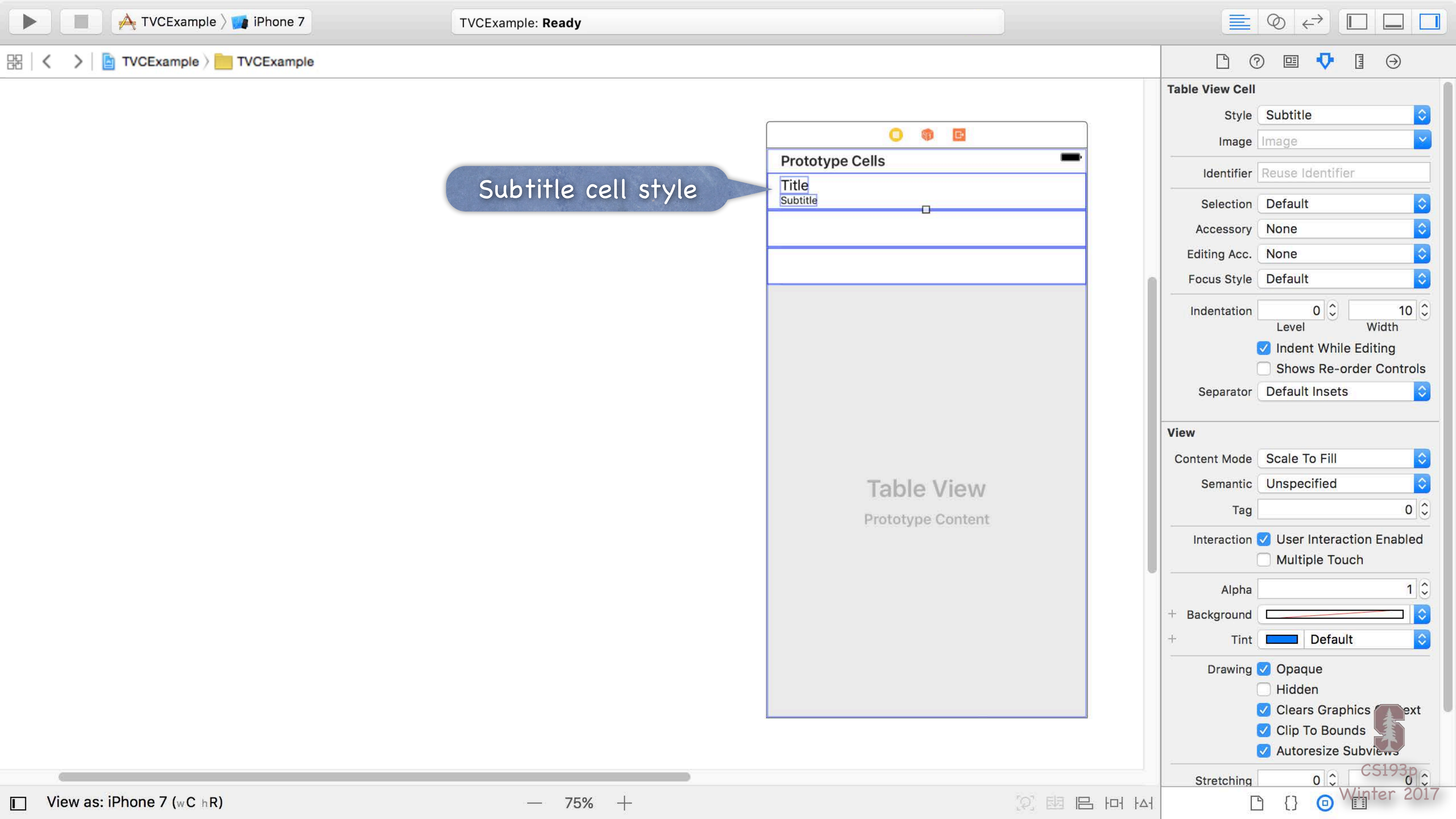
[ ] Hidden

[✓] Clears Graphics Context

[✓] Clip To Bounds

[✓] Autoresize Subvi

Stretching: 0 X, 0 Y


TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Subtitle cell style

Prototype Cells

Title
Subtitle

Table View
Prototype Content

Table View Cell

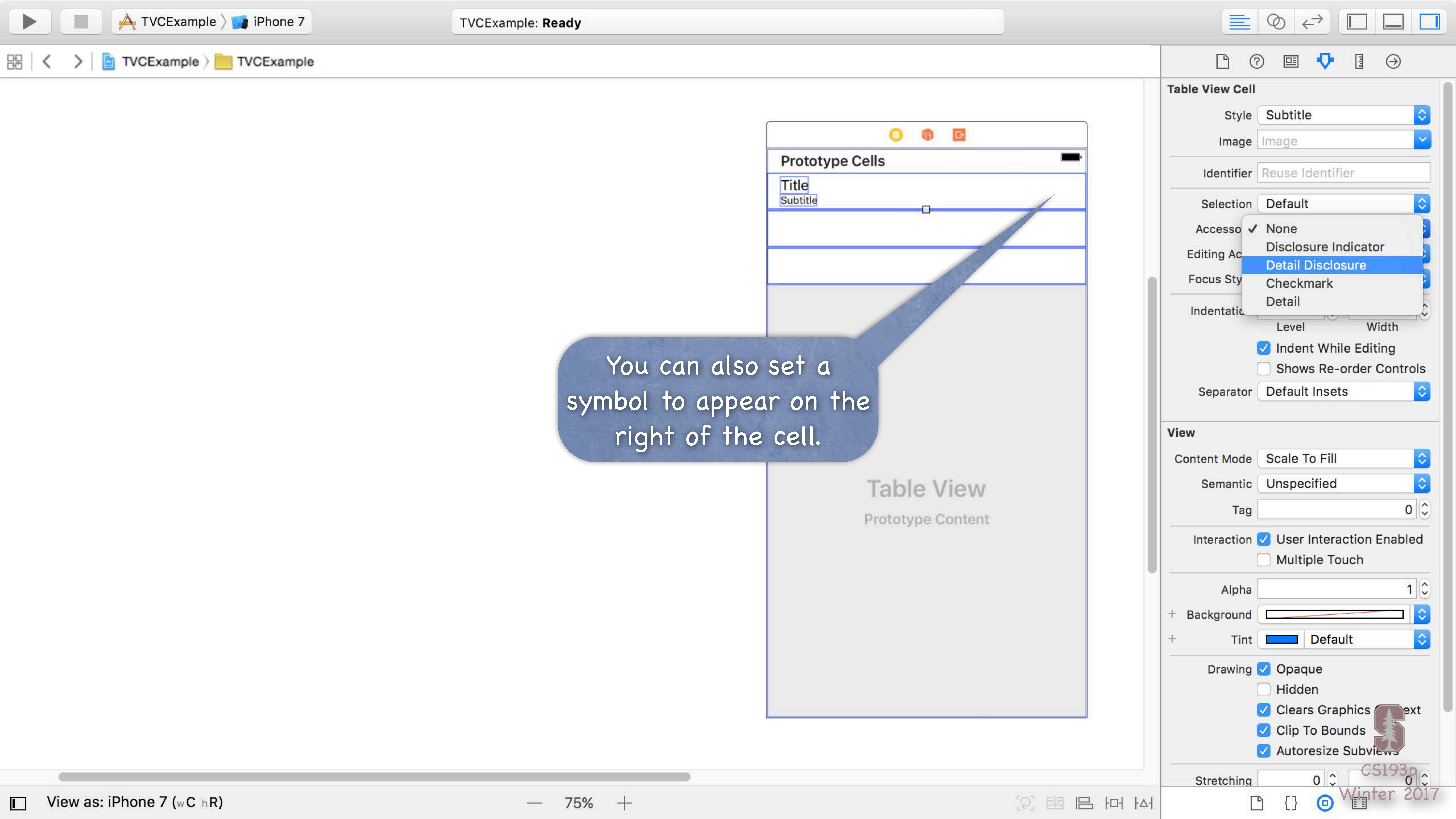
Style	Subtitle
Image	Image
Identifier	Reuse Identifier
Selection	Default
Accessory	None
Editing Acc.	None
Focus Style	Default
Indentation	0 Level, 10 Width
	✓ Indent While Editing
	Shows Re-order Controls
Separator	Default Insets

View

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

View as: iPhone 7 (wC hR)

75%



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Prototype Cells

Title
Subtitle

Table View
Prototype Content

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

Table View Cell

Style: Subtitle
Image: Image
Identifier: Reuse Identifier
Selection: Default
Accessory: ✓ None
- Disclosure Indicator
- Detail Disclosure
- Checkmark
- Detail

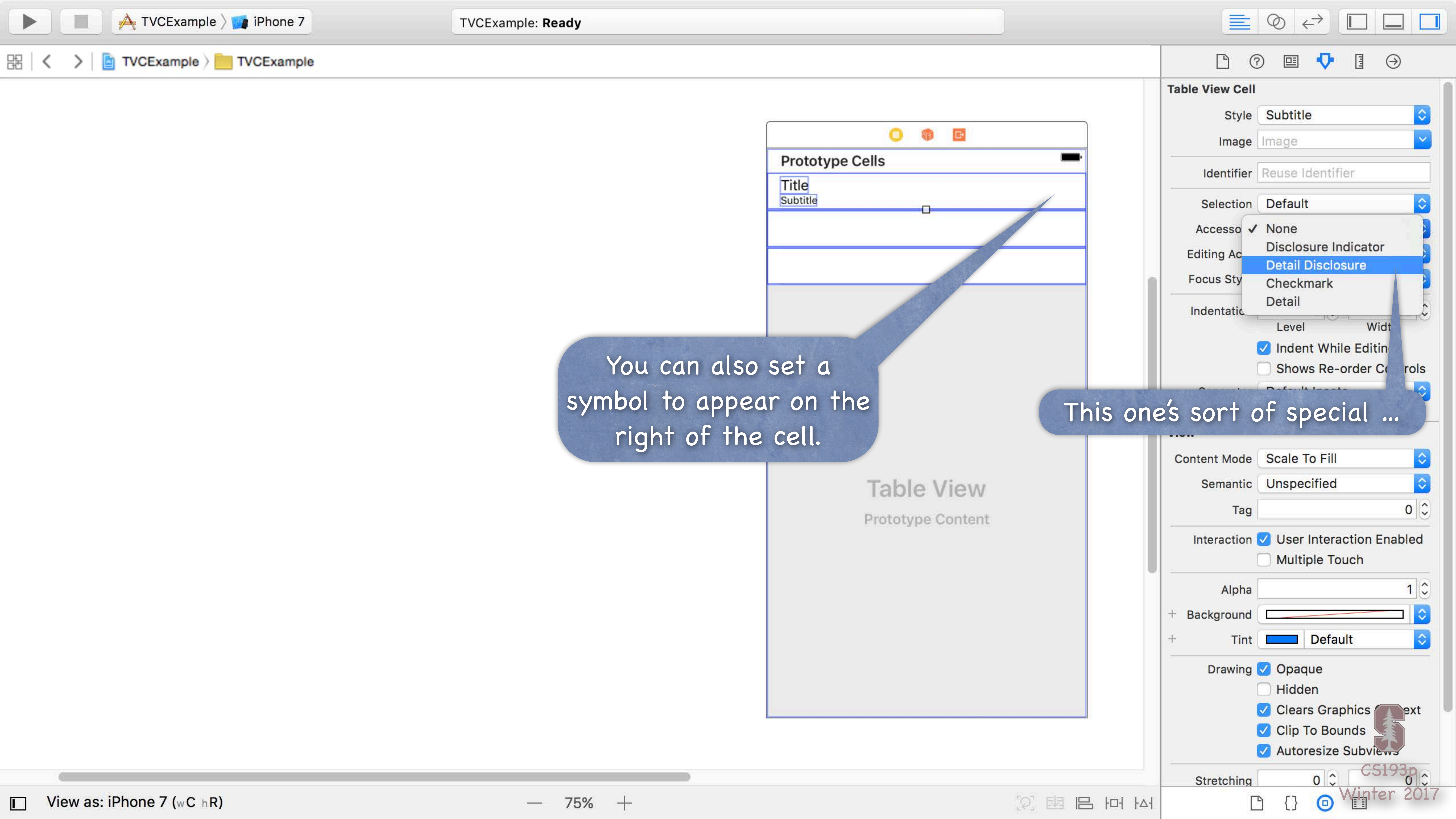
Editing Accessory
Focus Style
Indentation: Level, Width
✓ Indent While Editing
Shows Re-order Controls
Separator: Default Insets

View

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

View as: iPhone 7 (wC hR)

75%



TVCExample > iPhone 7

TVCExample: **Ready**

TVCExample > TVCExample

Table View Cell

Style: Subtitle

Image: Image

Identifier: Reuse Identifier

Selection: Default

Accessory: ✓ None / Disclosure Indicator / **Detail Disclosure** / Checkmark / Detail

Editing Ac

Focus Sty

Indentatic — Level, Widt

✓ Indent While Editing

Shows Re-order Controls

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 Context; ✓ Clip To Bounds; ✓ Autoresize Subviews

Stretching: 0 0

Prototype Cells

Title

Subtitle

Table View

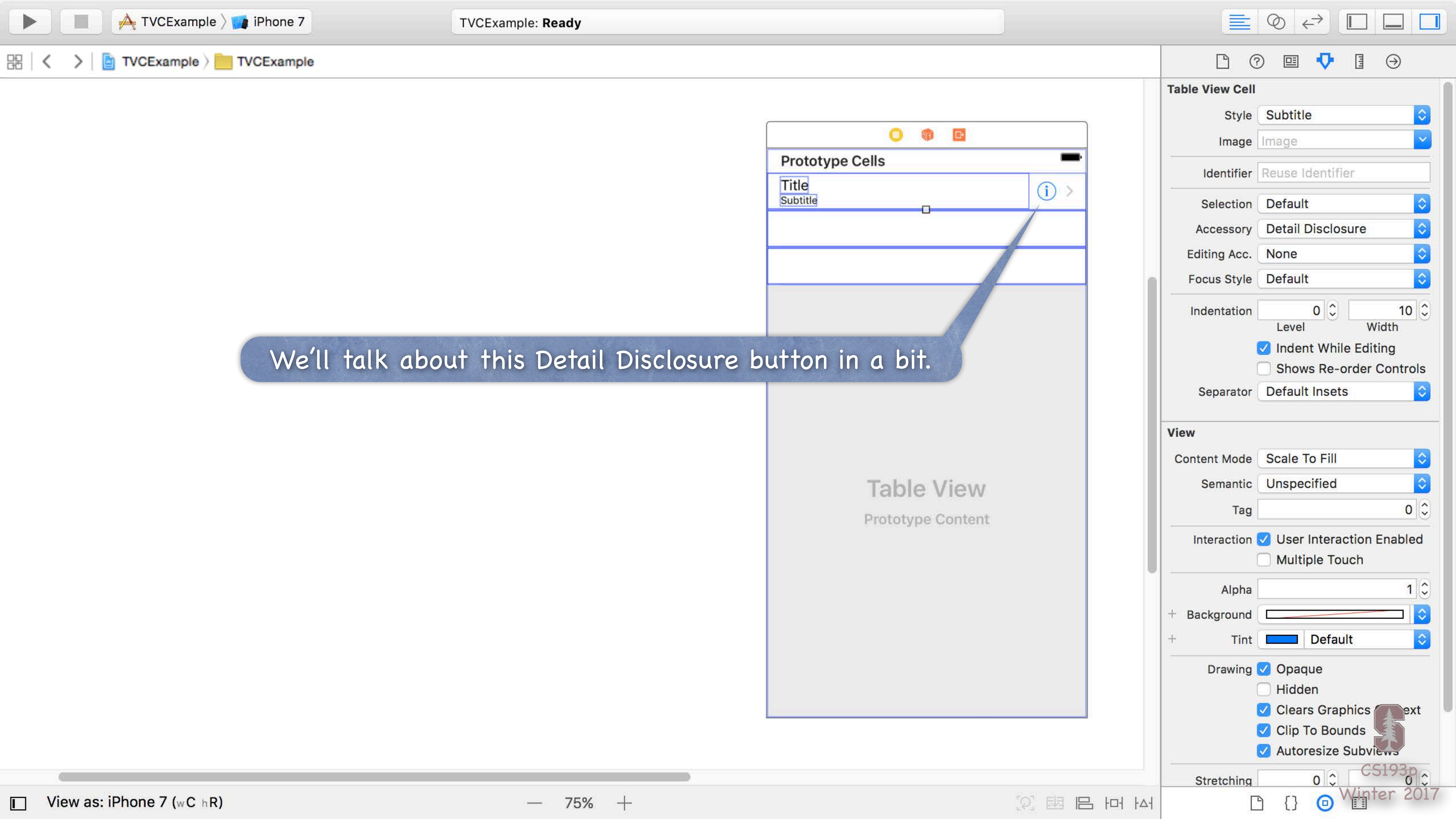
Prototype Content

View as: iPhone 7 (wC hR)

75%

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

This one's sort of special ...



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

TVCExample > iPhone 7

TVCExample: **Ready**

TVCExample > TVCExample

Prototype Cells

Title
Subtitle

Table View
Prototype Content

Table View Cell

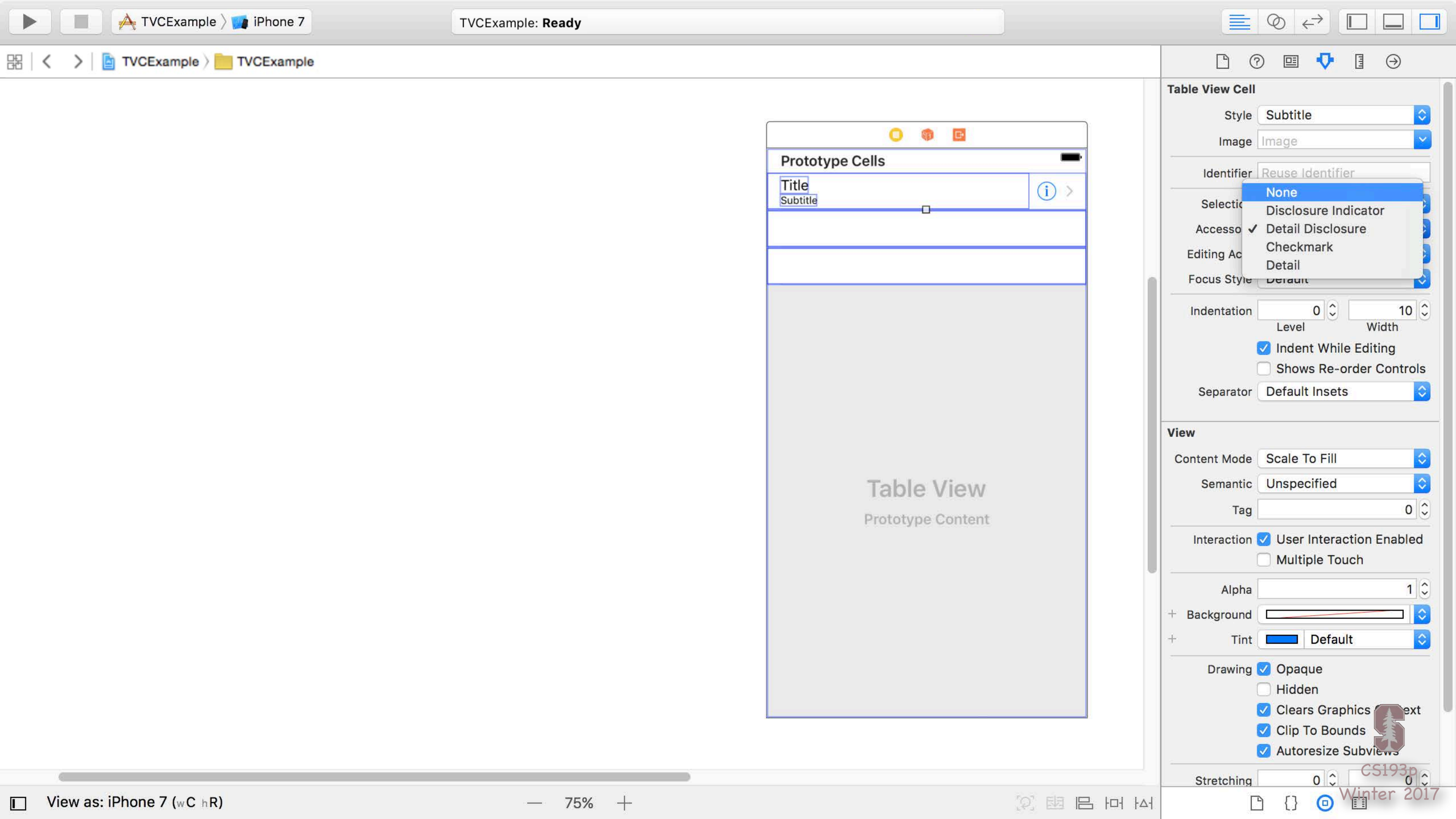
Style: Subtitle
Image: Image
Identifier: Reuse Identifier
Selection: Default
Accessory: Detail Disclosure
Editing Acc.: None
Focus Style: Default
Indentation: 0 Level, 10 Width
☑ Indent While Editing
☐ Shows Re-order Controls
Separator: Default Insets

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 Context
☑ Clip To Bounds
☑ Autoresize Subviews
Stretching: 0 0

View as: iPhone 7 (wC hR)

75%


TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Prototype Cells

Title

Subtitle

Table View

Prototype Content

Table View Cell

Style: Subtitle

Image: Image

Identifier: Reuse Identifier

Selection

Accessory

Editing Accessory

Focus Style: Default

- None
- Disclosure Indicator
- ✓ Detail Disclosure
- Checkmark
- Detail

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

Tint: Default

Drawing: [✓] Opaque

[ ] Hidden

[✓] Clears Graphics Context

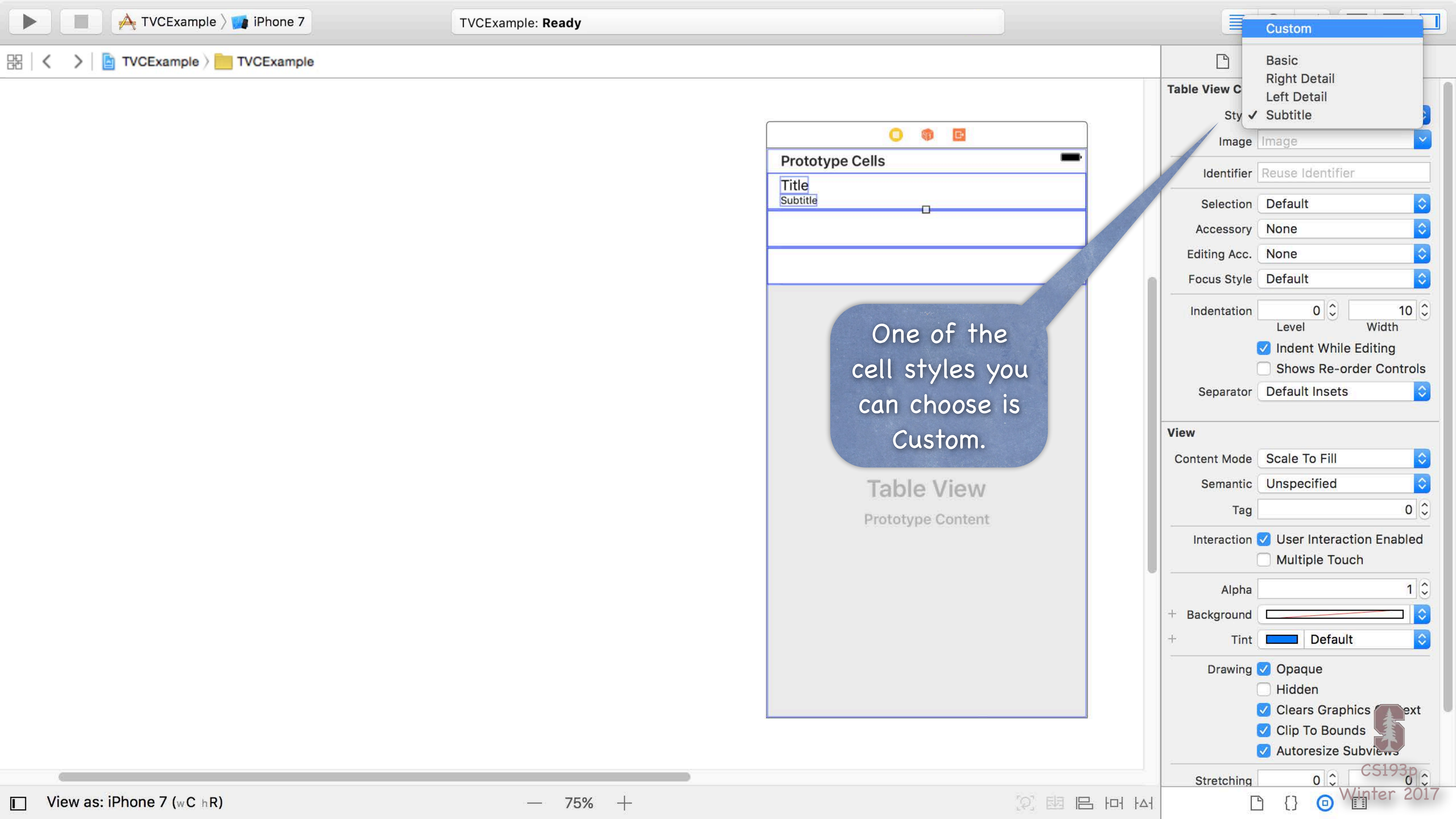
[✓] Clip To Bounds

[✓] Autoresize Subviews

Stretching: 0 0

View as: iPhone 7 (wC hR)

75%



One of the cell styles you can choose is Custom.

TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Prototype Cells

Title
Subtitle

Table View
Prototype Content

View as: iPhone 7 (wC hR)   75%

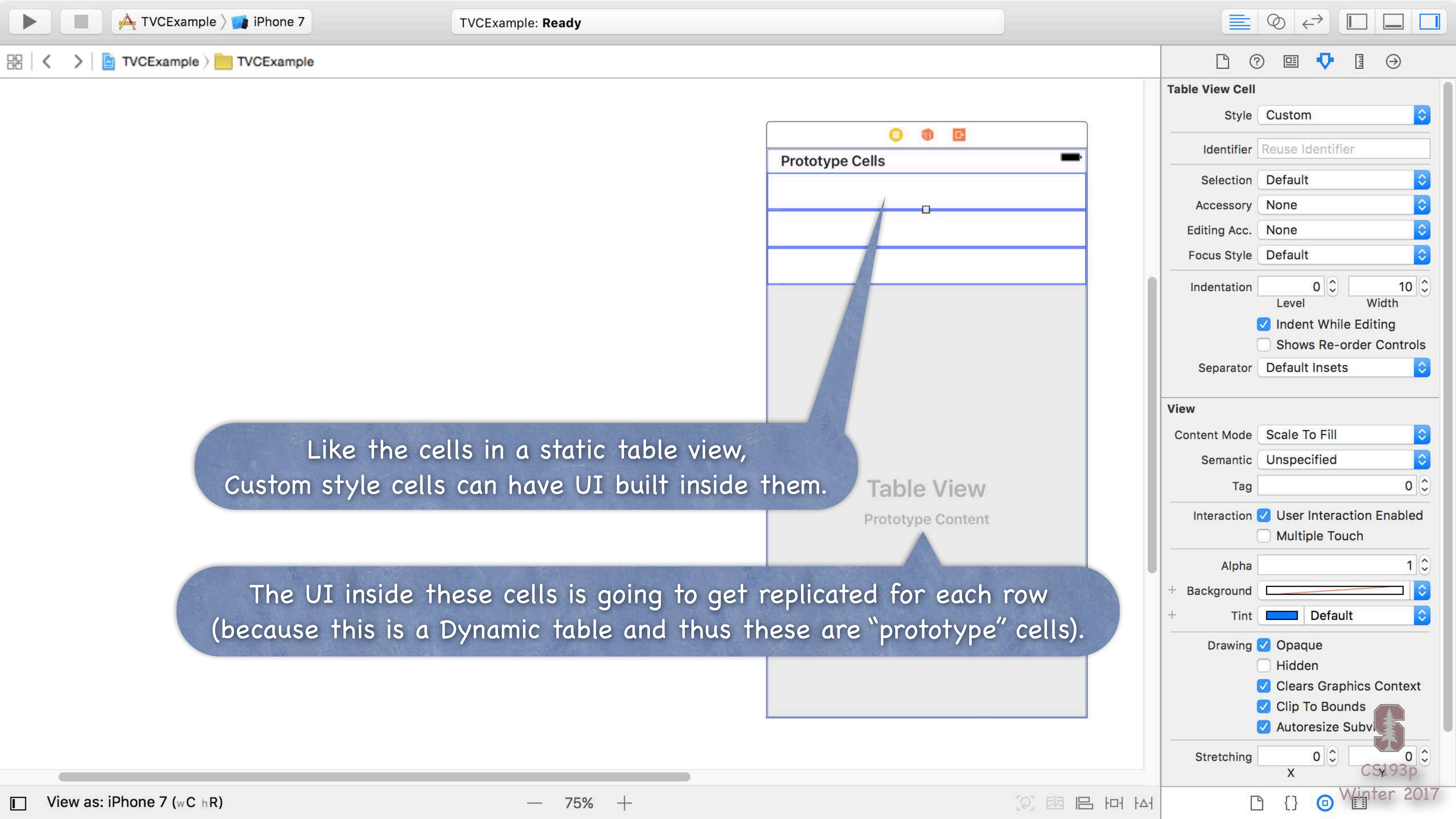
Table View C

Sty
- Custom
- Basic
- Right Detail
- Left Detail
- ✓ 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
Tint: Default
Drawing: ✓ Opaque
☐ Hidden
✓ Clears Graphics Context
✓ Clip To Bounds
✓ Autoresize Subviews
Stretching: 0 0



Like the cells in a static table view,
Custom style cells can have UI built inside them.

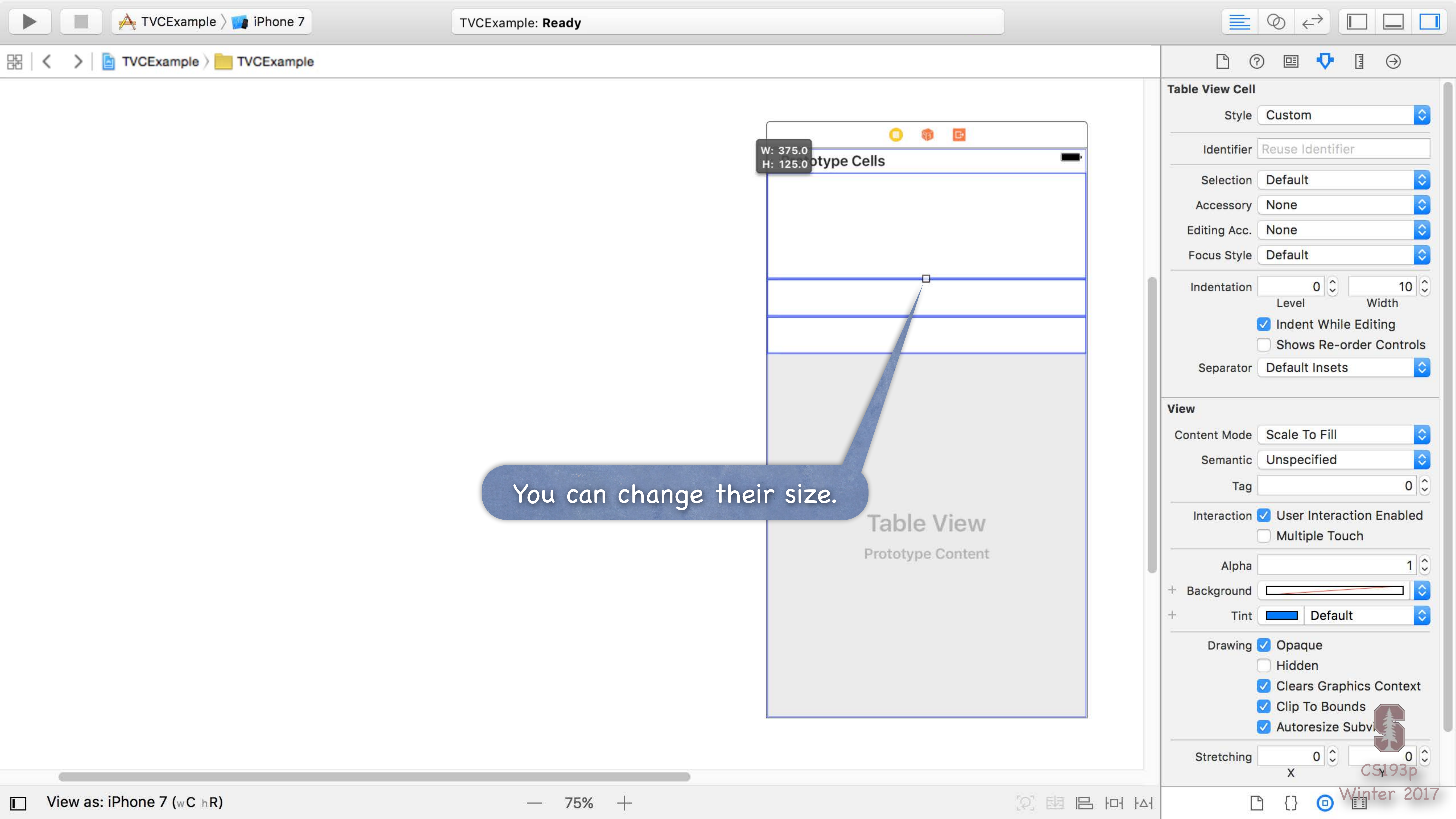
The UI inside these cells is going to get replicated for each row
(because this is a Dynamic table and thus these are "prototype" cells).

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
Tint: Default
Drawing: ✓ Opaque
Hidden
✓ Clears Graphics Context
✓ Clip To Bounds
✓ Autoresize Subvi
Stretching: 0 X, 0 Y


TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

W: 375.0
H: 125.0

Prototype Cells

Table View
Prototype Content

You can change their size.

Table View Cell

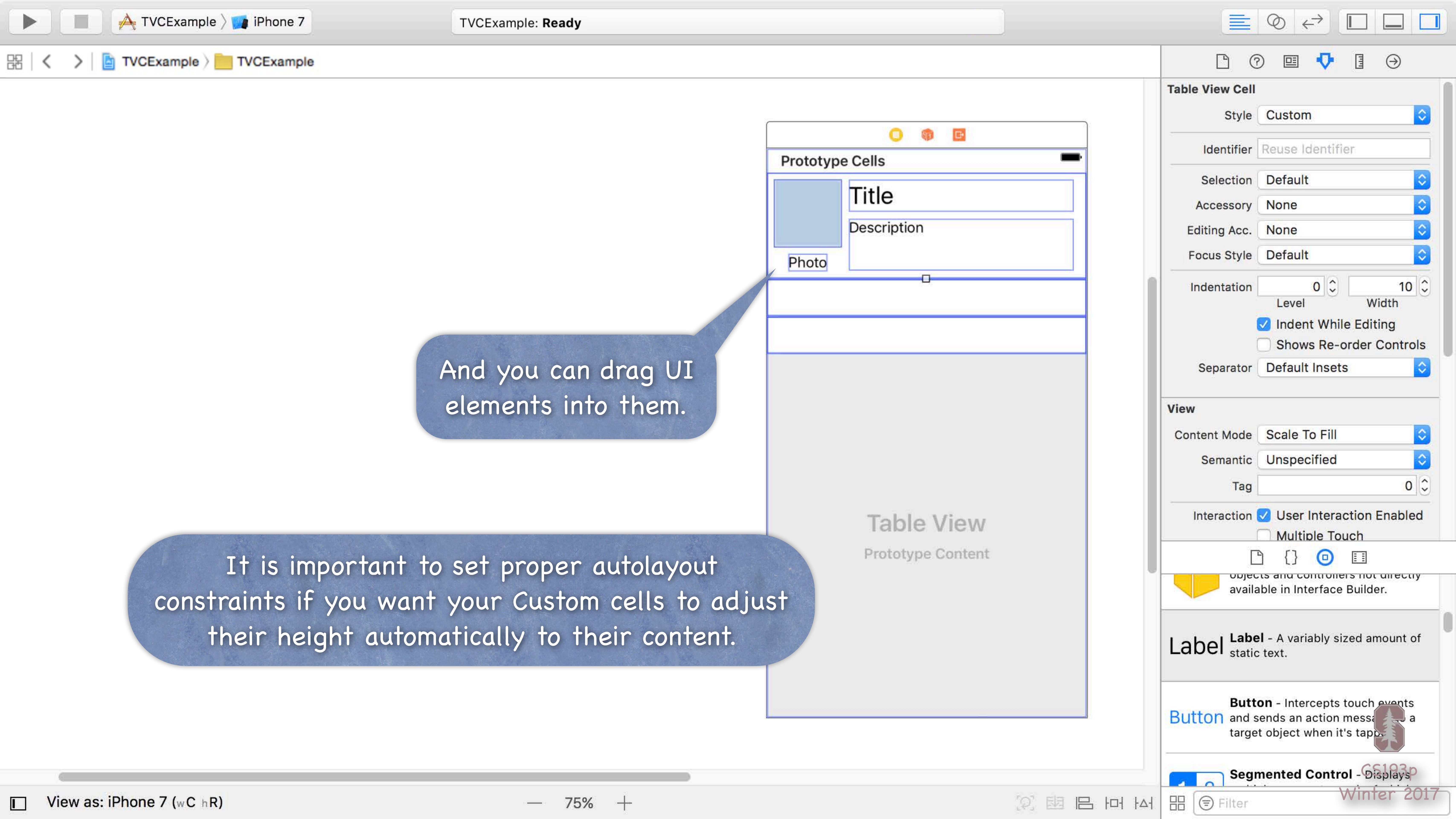
Style: Custom
Identifier: Reuse Identifier
Selection: Default
Accessory: None
Editing Acc.: None
Focus Style: Default
Indentation: 0 Level, 10 Width
☑ Indent While Editing
☐ Shows Re-order Controls
Separator: Default Insets

View

Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ☑ User Interaction Enabled
☐ Multiple Touch
Alpha: 1
Background
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subvi
Stretching: 0 X, 0 Y

View as: iPhone 7 (wC hR)

75%



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.

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 - Displays

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.

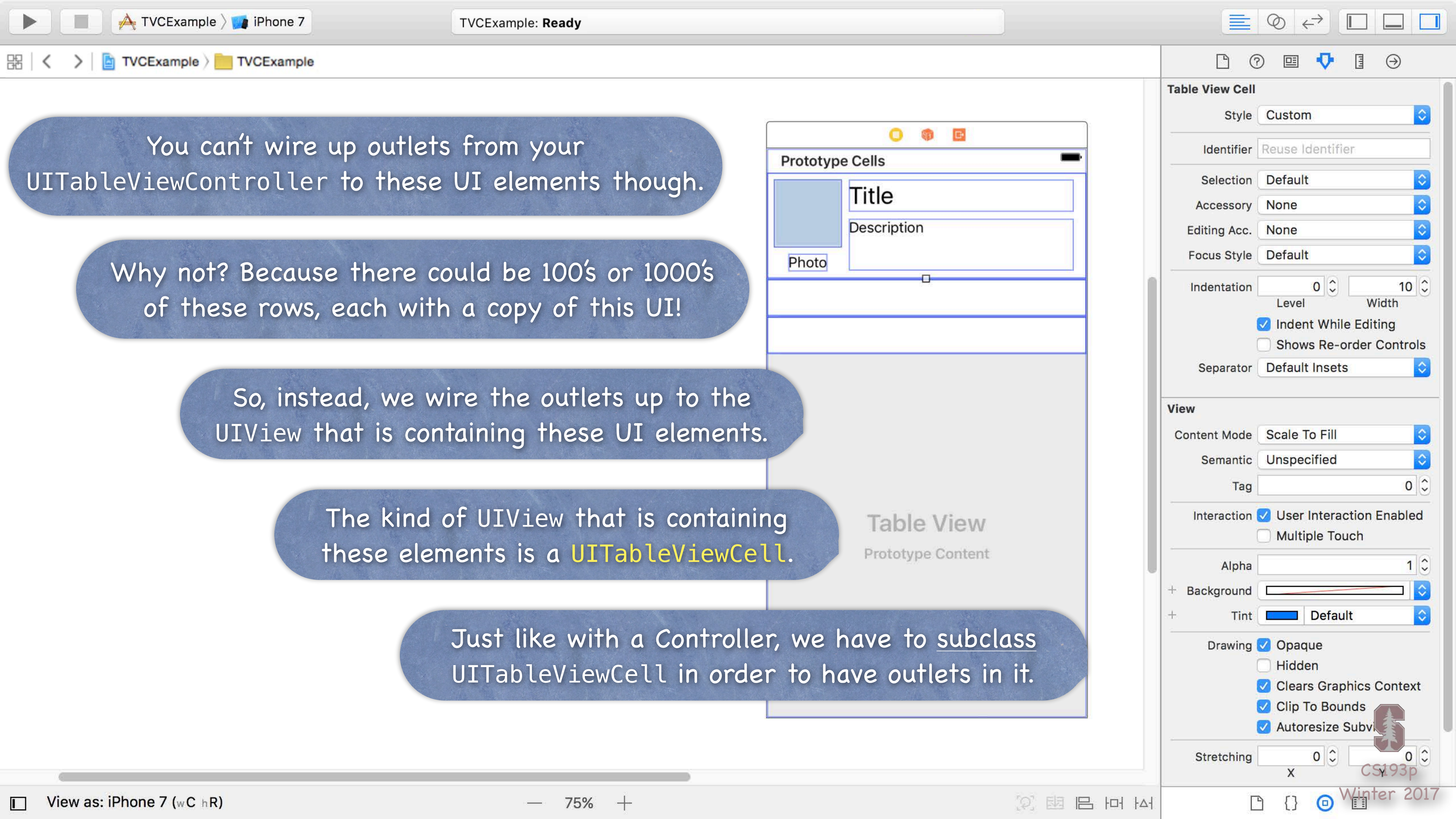
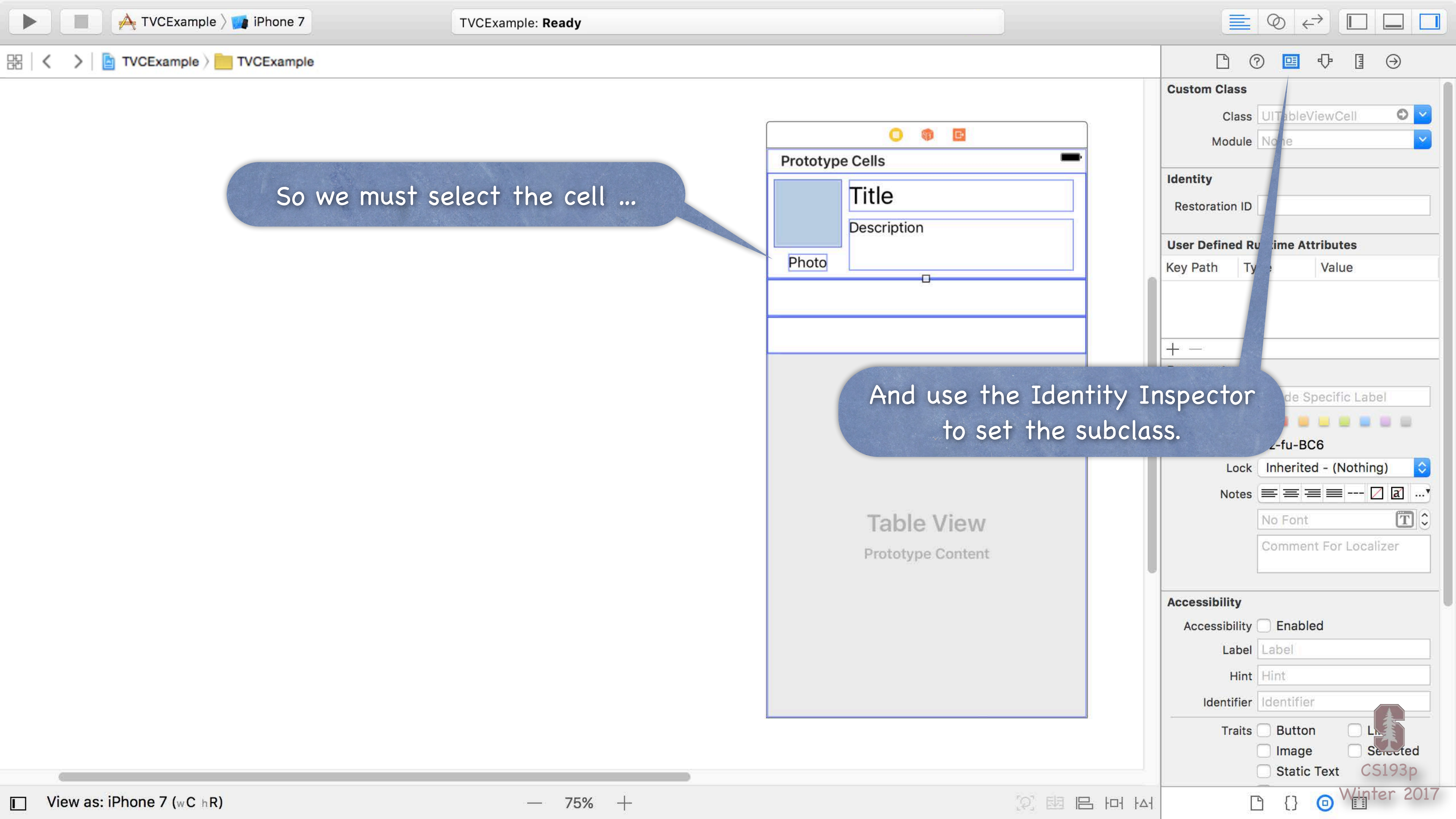


Table View Cell

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:
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subvi
Stretching: 0 (X), 0 (Y)



So we must select the cell …

And use the Identity Inspector to set the subclass.

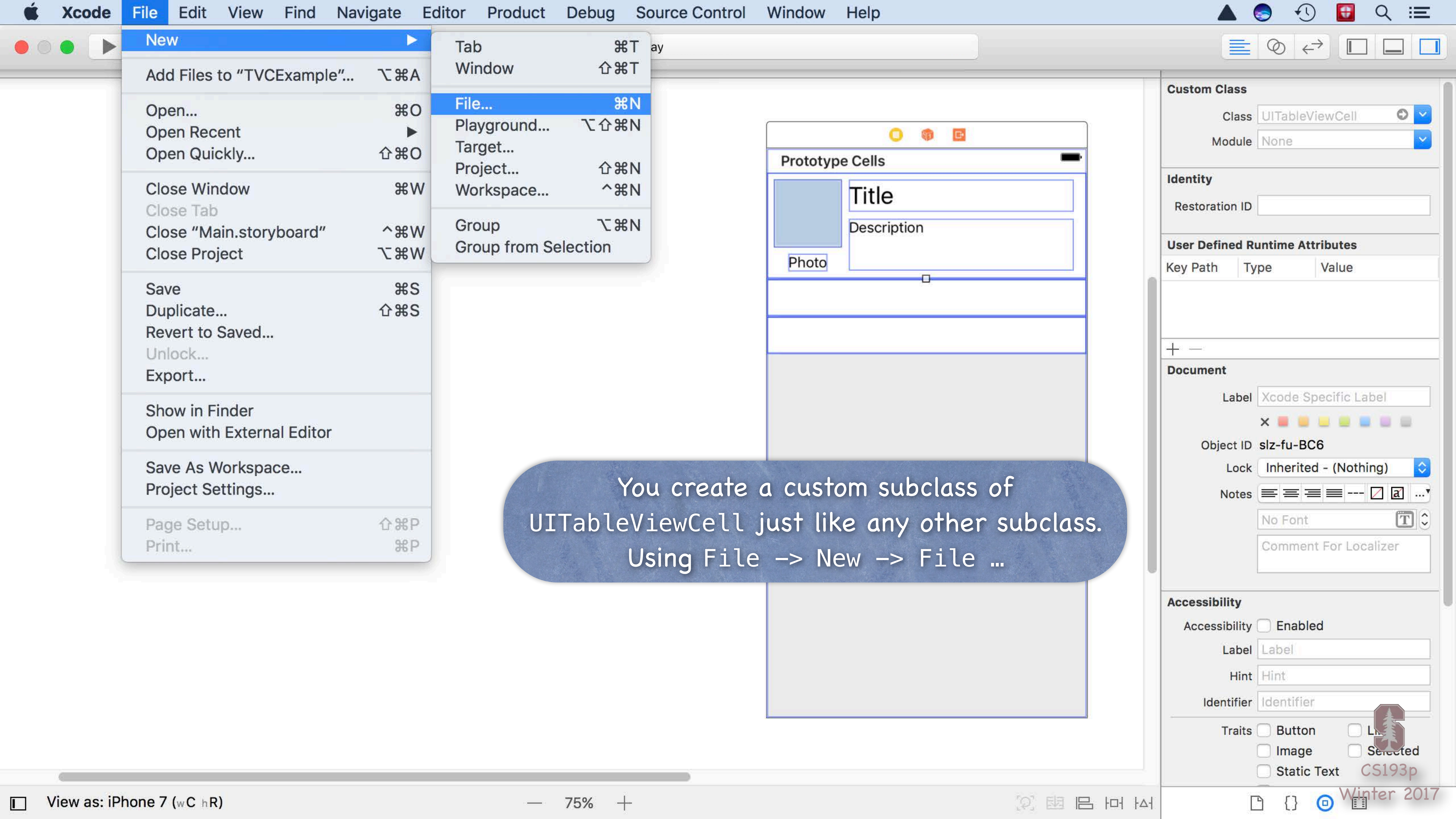
Custom Class
Class: UITableViewCell
Module: None

Identity
Restoration ID:

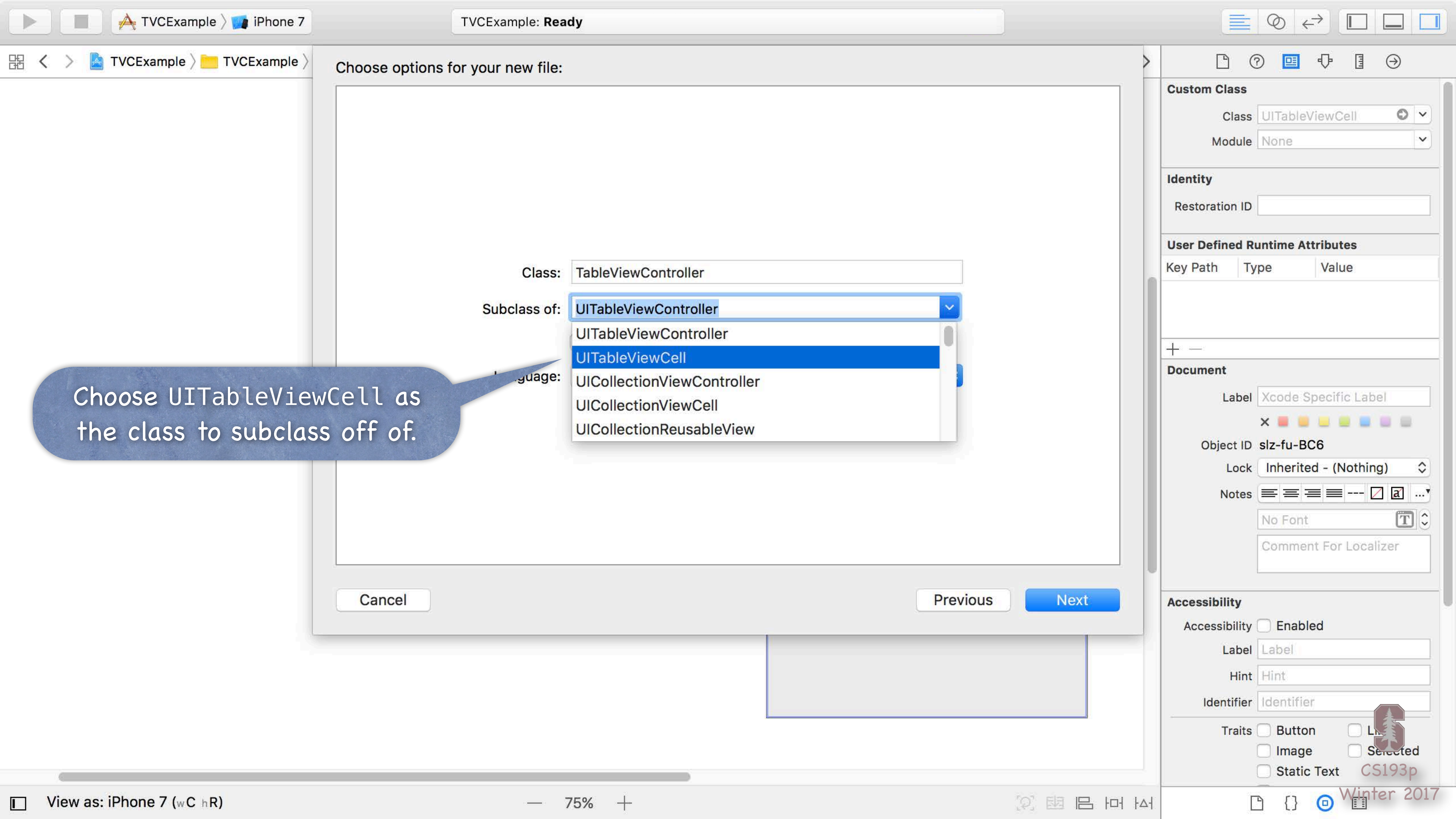
User Defined Runtime Attributes
Key Path | Type | Value

Lock: Inherited - (Nothing)
Notes: No Font, Comment For Localizer

Accessibility
Accessibility: Enabled
Label: Label
Hint: Hint
Identifier: Identifier
Traits: Button, Image, Static Text, Selected



You create a custom subclass of UITableViewCell just like any other subclass.
Using File -> New -> File …



TVCExample | iPhone 7 | TVCExample: **Ready**

TVCExample > TVCExample

Choose options for your new file:

Class: TableViewController

Subclass of: UITableViewController

- UITableViewController
- UITableViewCell
- UICollectionViewController
- UICollectionViewCell
- UICollectionReusableView

Language:

Cancel | Previous | Next

Choose UITableViewCell as the class to subclass off of.

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

Accessibility: Enabled

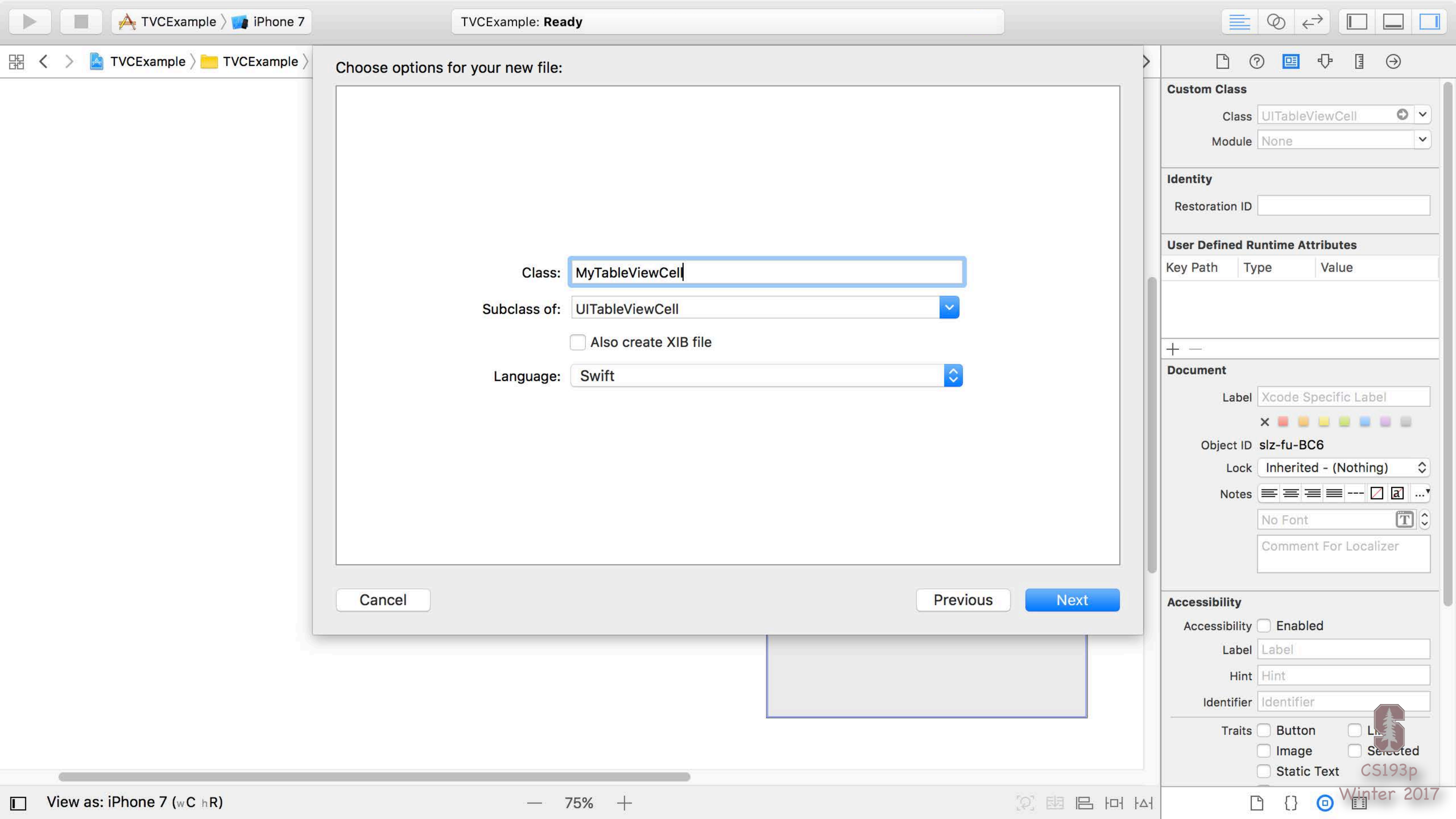
Label: Label

Hint: Hint

Identifier: Identifier

Traits: Button, Link, Image, Selected, Static Text

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



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample 〉

Choose options for your new file:

Class: MyTableViewCell

Subclass of: UITableViewCell

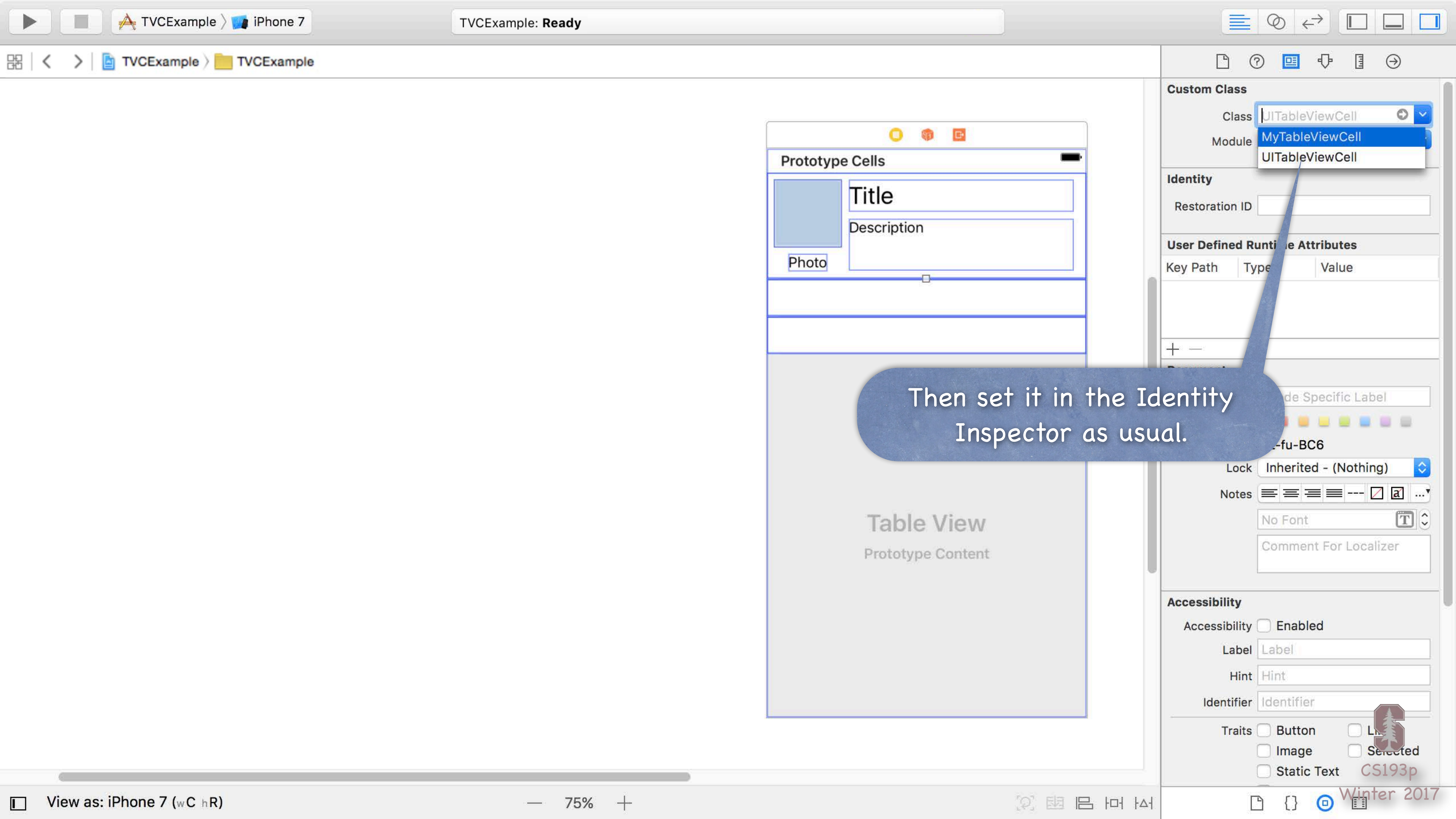
☐ Also create XIB file

Language: Swift

Cancel

Previous

Next



TVCExample: Ready

Custom Class

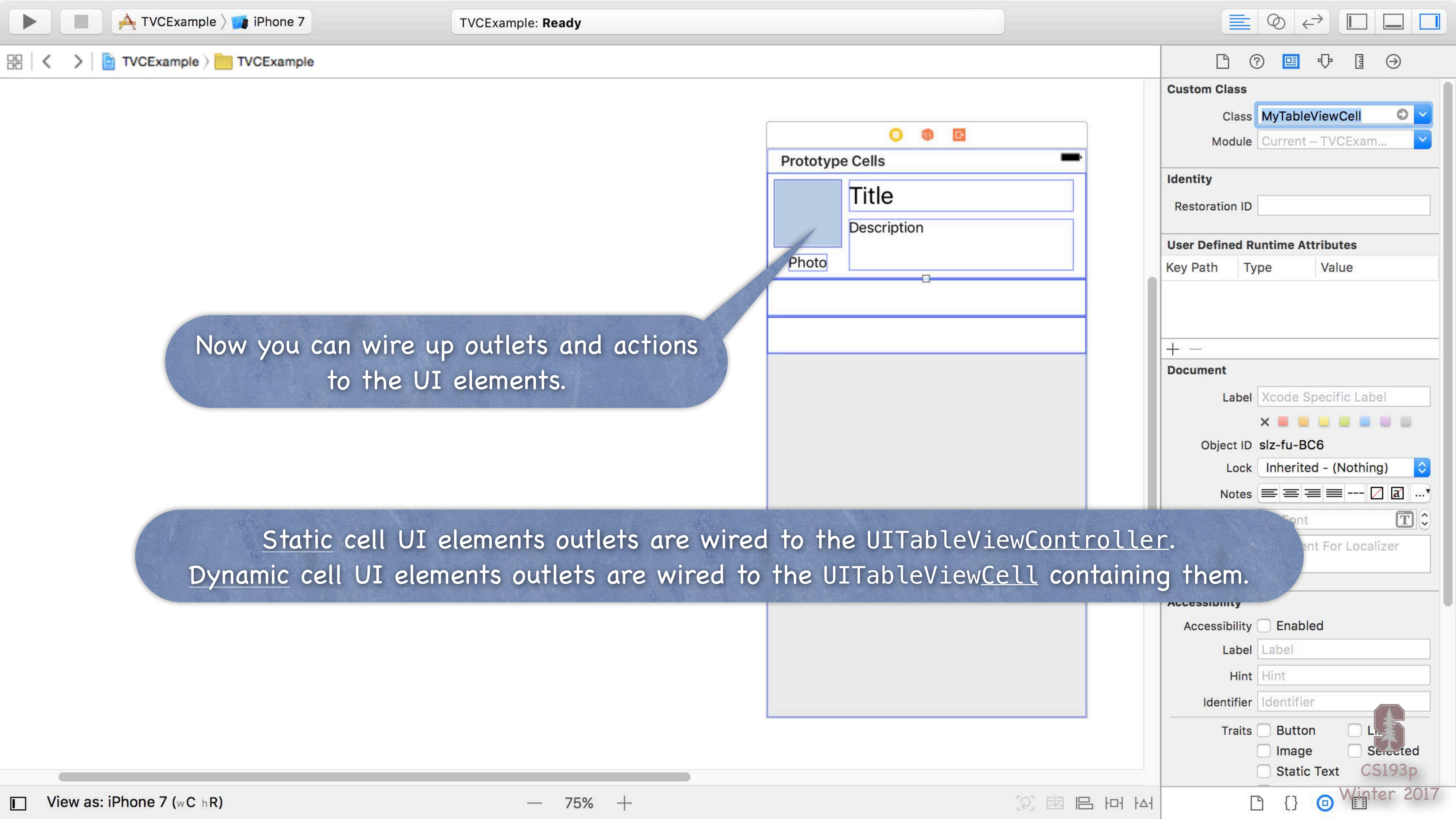
Class: UITableViewCell

Module

MyTableViewCell

UITableViewCell

Then set it in the Identity Inspector as usual.



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Prototype Cells

Custom Class

Class MyTableViewCell

Module Current – TVCExam...

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

Accessibility

Accessibility Enabled

Label Label

Hint Hint

Identifier Identifier

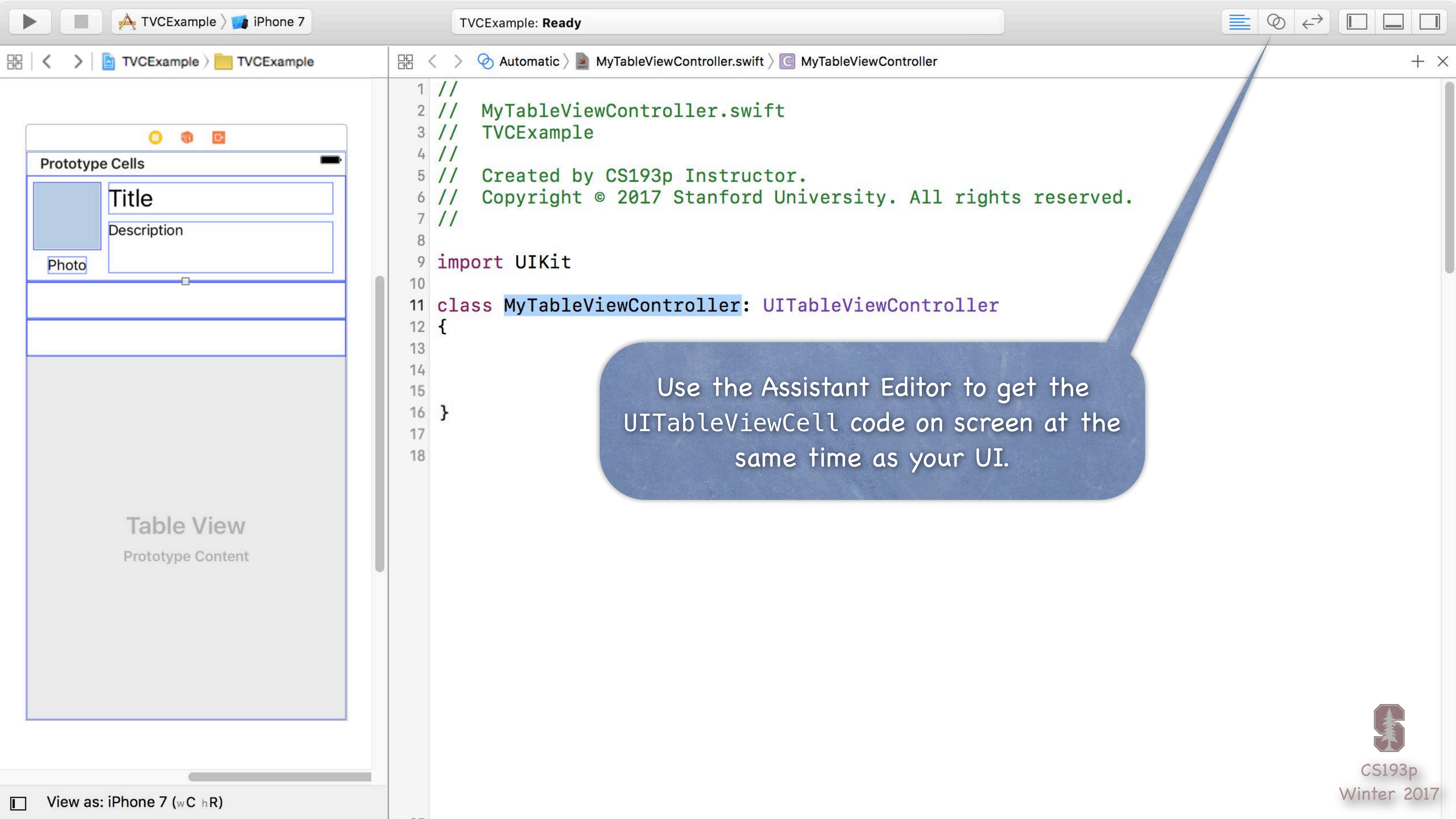
Traits Button, Image, Static Text, Selected

View as: iPhone 7 (wC hR)

75%

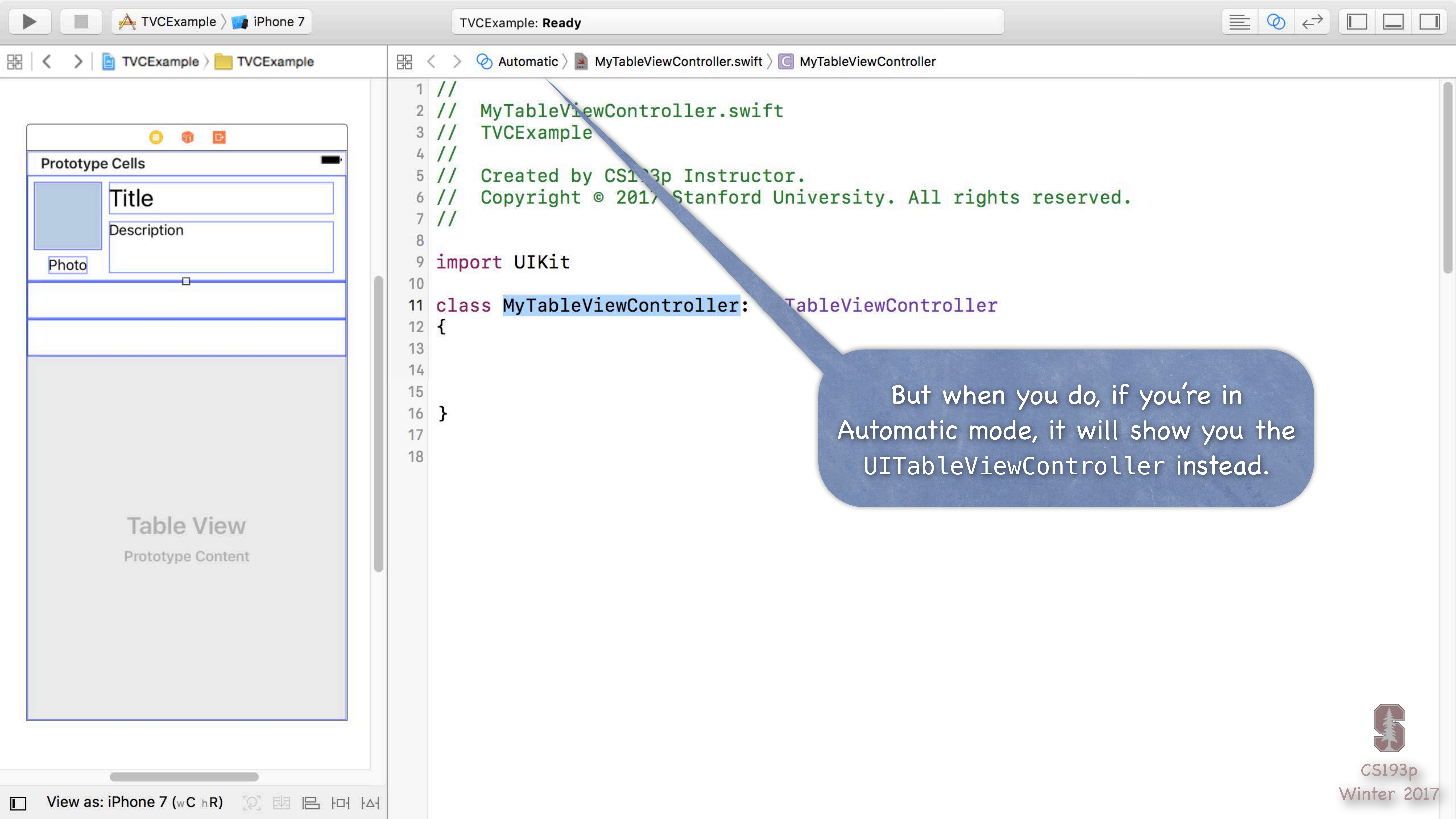
Now you can wire up outlets and actions to the UI elements.

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



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

import UIKit

class MyTableViewController: UITableViewController
{



}
```

Use the Assistant Editor to get the UITableViewCell code on screen at the same time as your UI.



TVCExample > iPhone 7

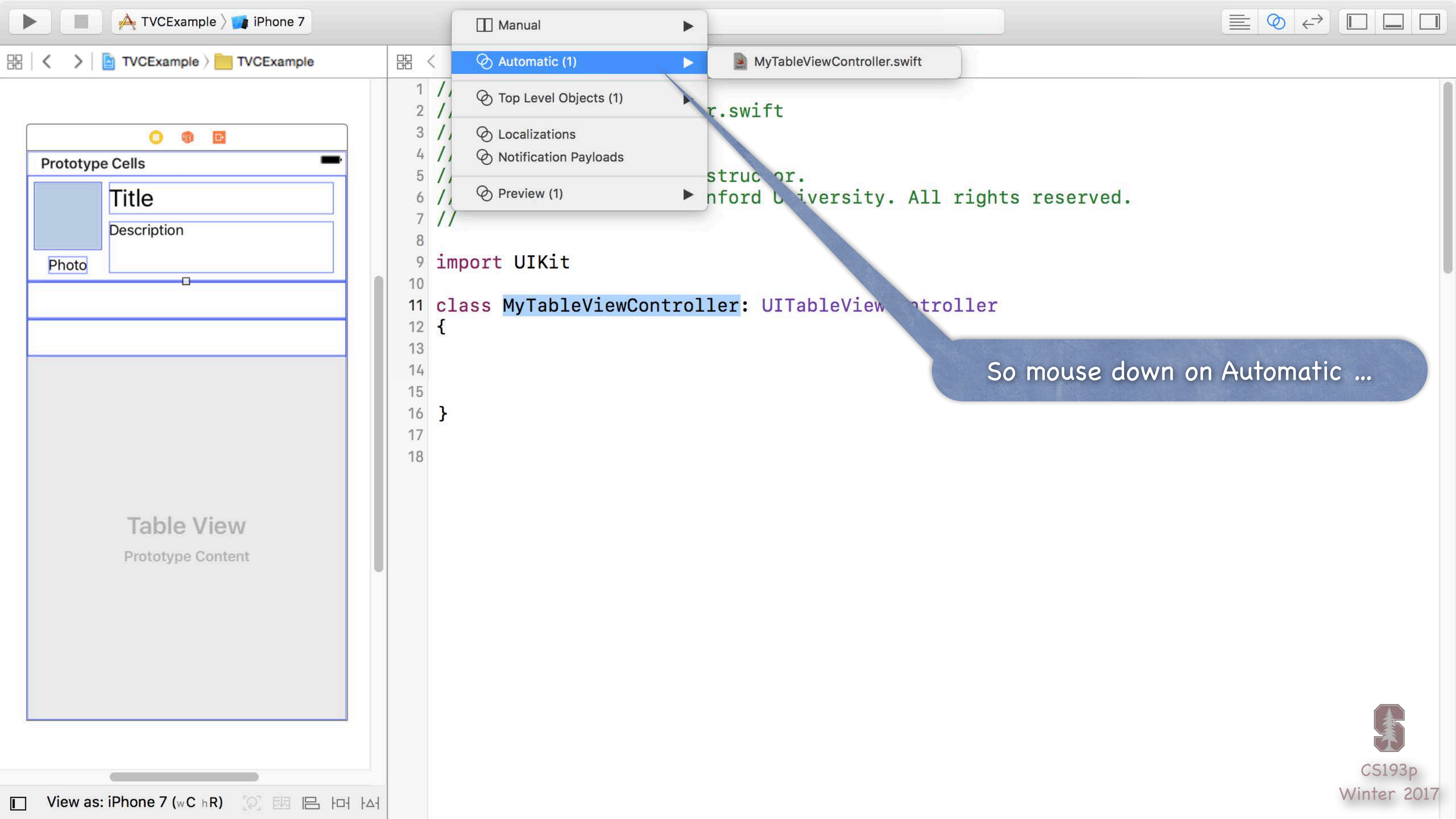
TVCExample: **Ready**

TVCExample > TVCExample

Automatic > MyTableViewController.swift > MyTableViewController

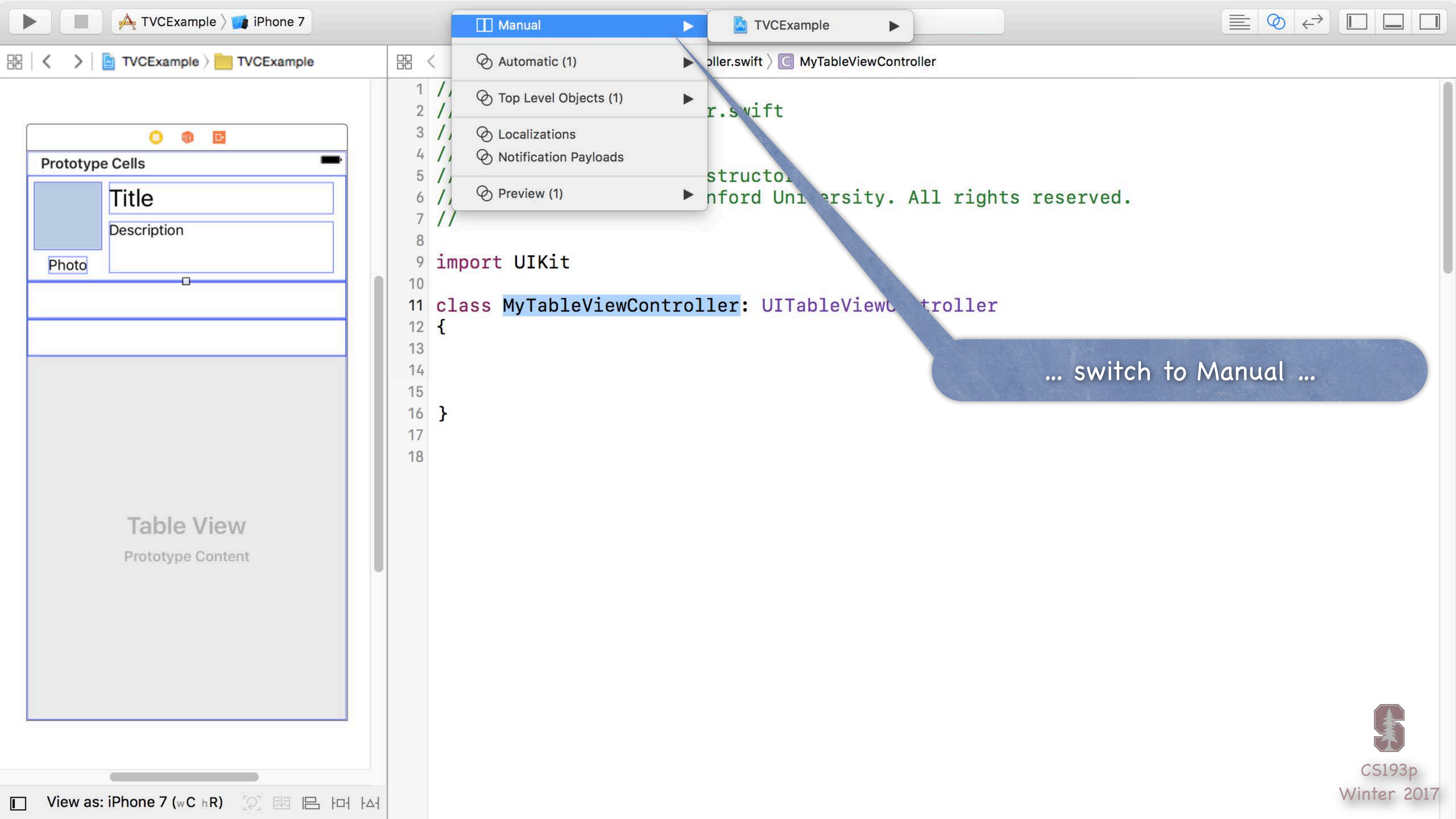
```swift
//
//  MyTableViewController.swift
//  TVCExample
//
//  Created by CS1?3p Instructor.
//  Copyright © 201? Stanford University. All rights reserved.
//

import UIKit

class MyTableViewController: ?TableViewController
{



}
```

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



```
//
//                    r.swift
//
//
//                    struc or.
//                    nford U iversity. All rights reserved.
//

import UIKit

class MyTableViewController: UITableView  troller
{



}
```

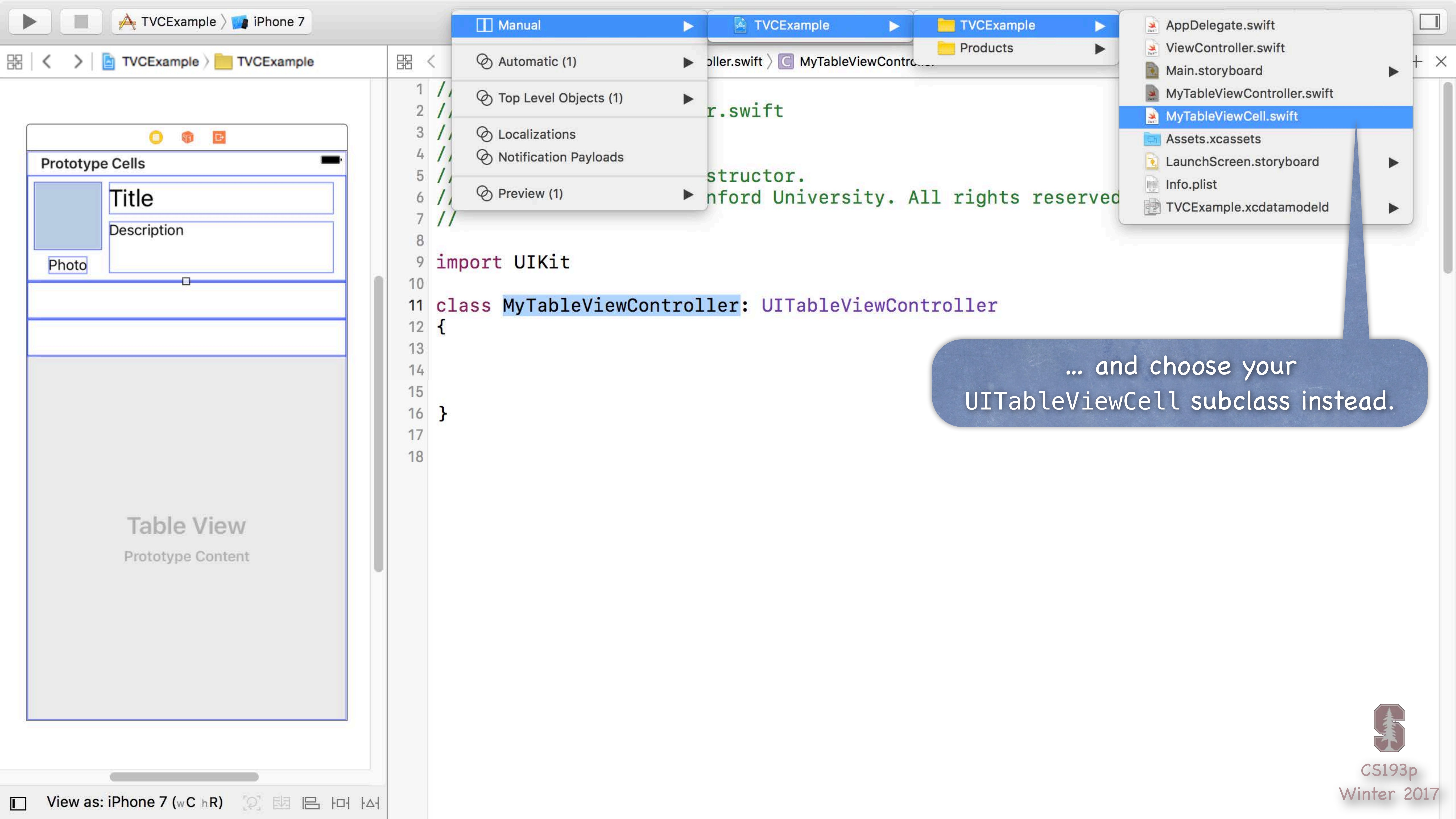
So mouse down on Automatic ...



```
//
//                    r.swift
//
//
//                    structo
//                    nford Uni ersity. All rights reserved.
//

import UIKit

class MyTableViewController: UITableViewC  troller
{



}
```

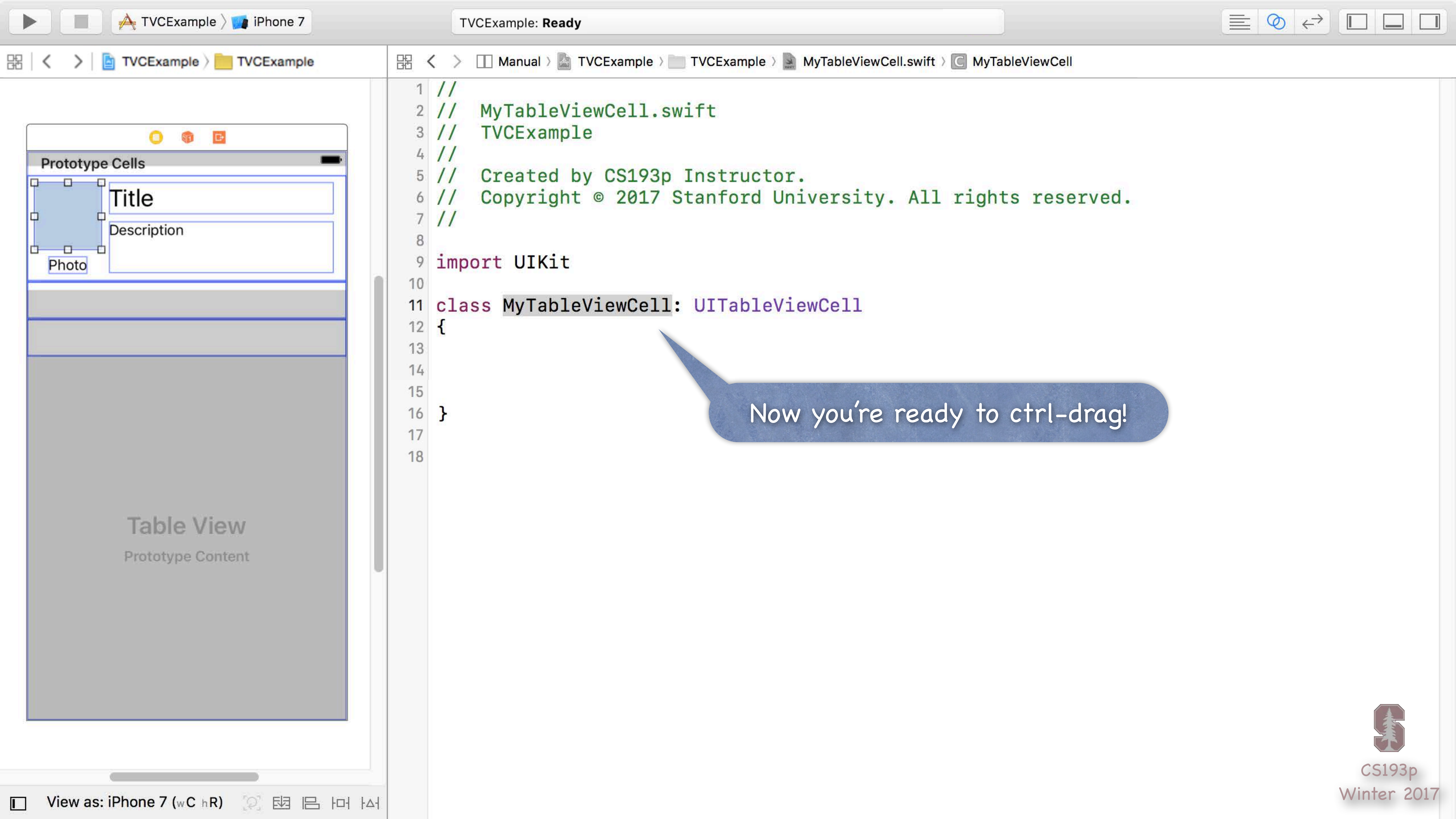
… switch to Manual …

CS193p
Winter 2017



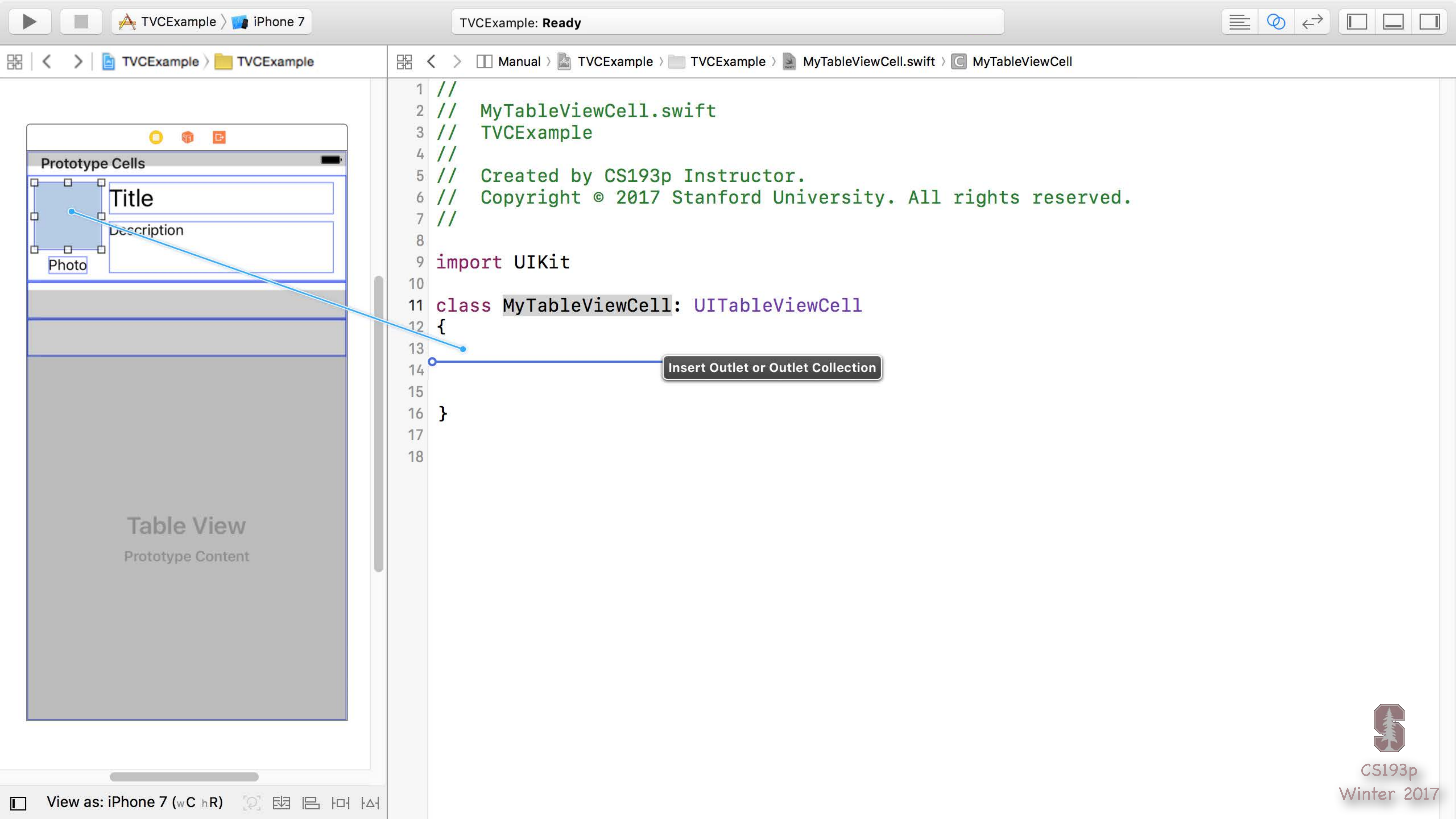
```
//
//                    r.swift
//
//
//                    structor.
//                    nford University. All rights reserved
//

import UIKit

class MyTableViewController: UITableViewController
{



}
```

… and choose your
UITableViewCell subclass instead.

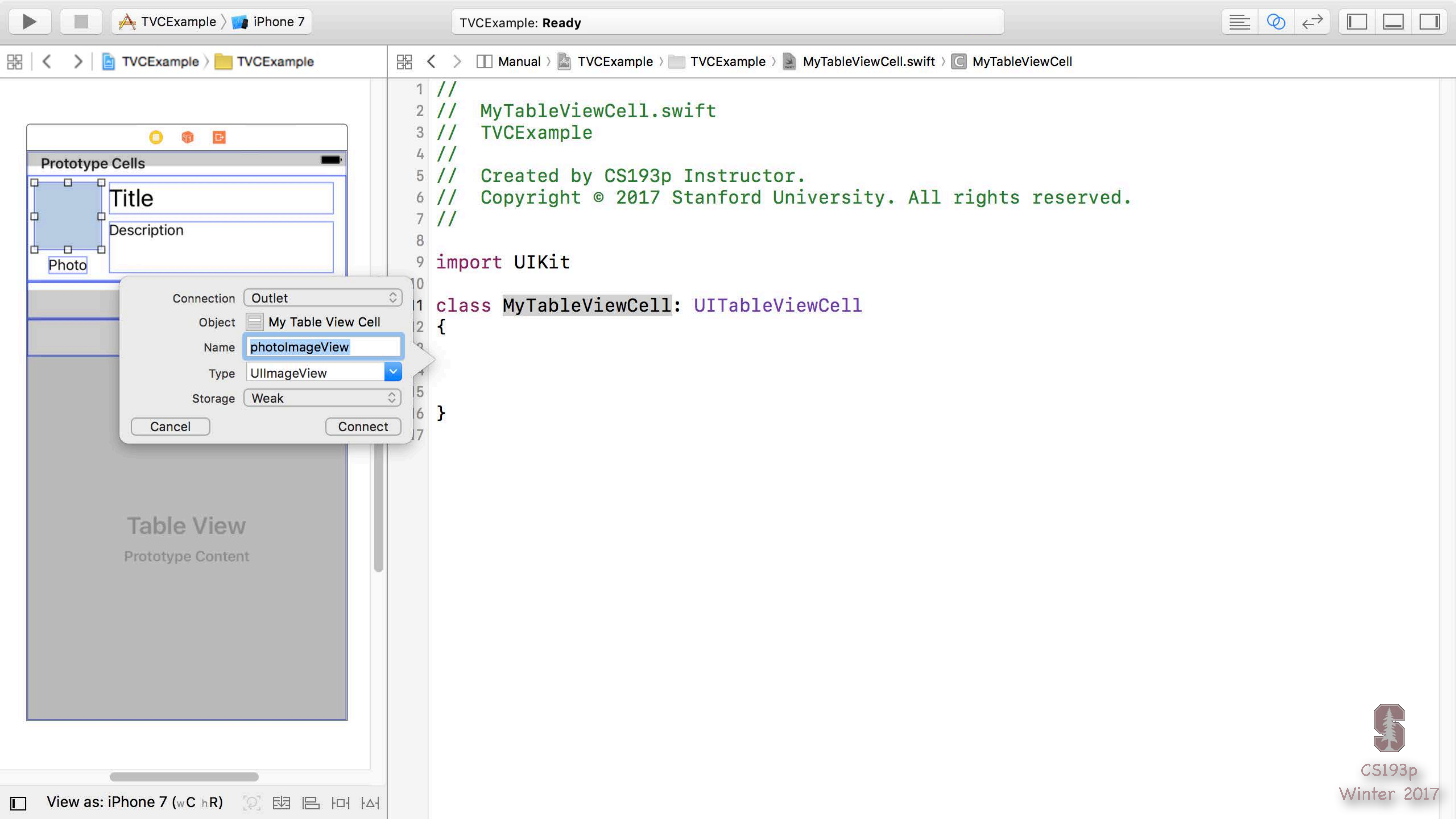


```
//
//  MyTableViewCell.swift
//  TVCExample
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class MyTableViewCell: UITableViewCell
{



}
```

Now you're ready to ctrl-drag!



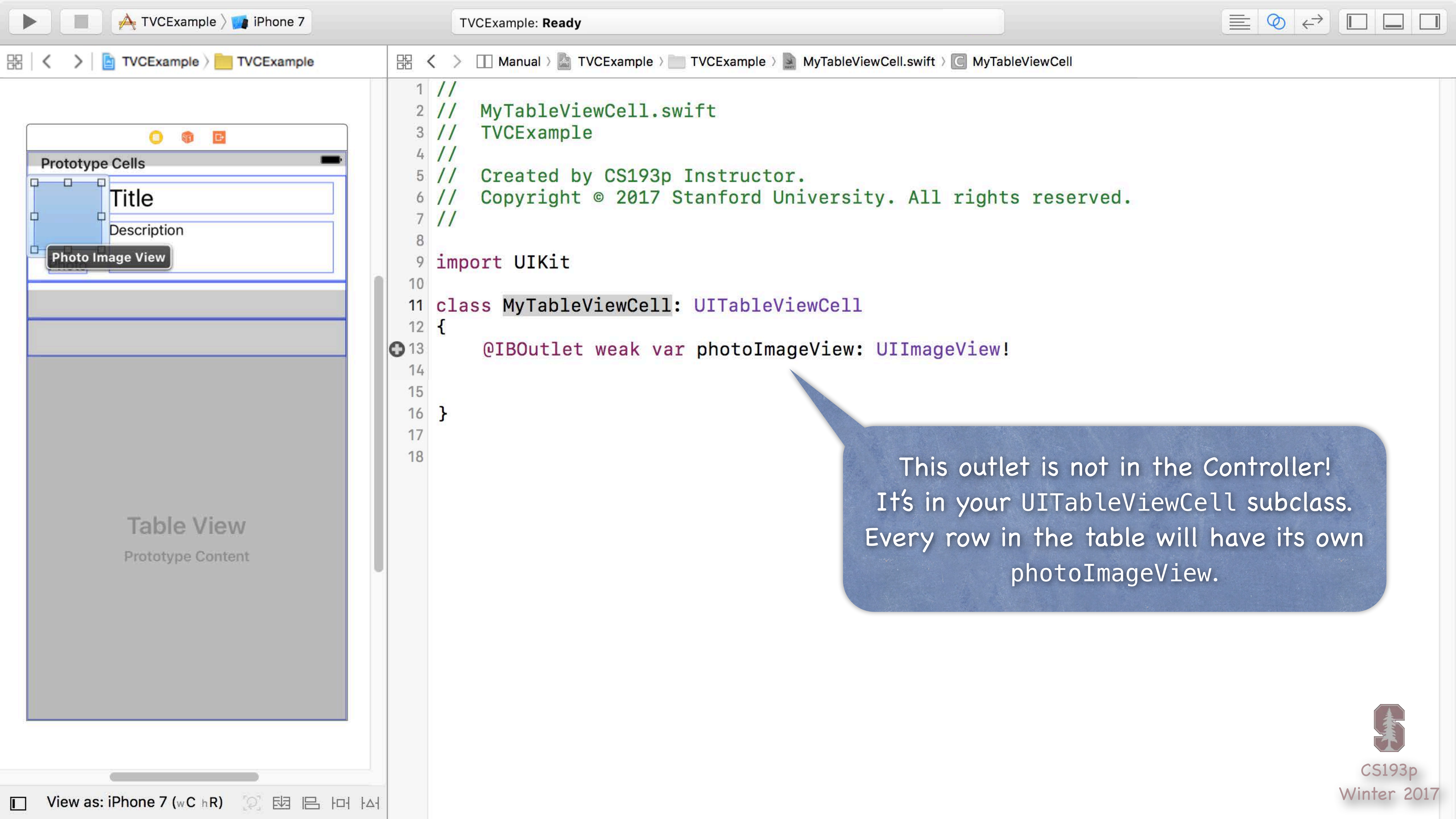
```swift
//
//  MyTableViewCell.swift
//  TVCExample
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit

class MyTableViewCell: UITableViewCell
{


}
```



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

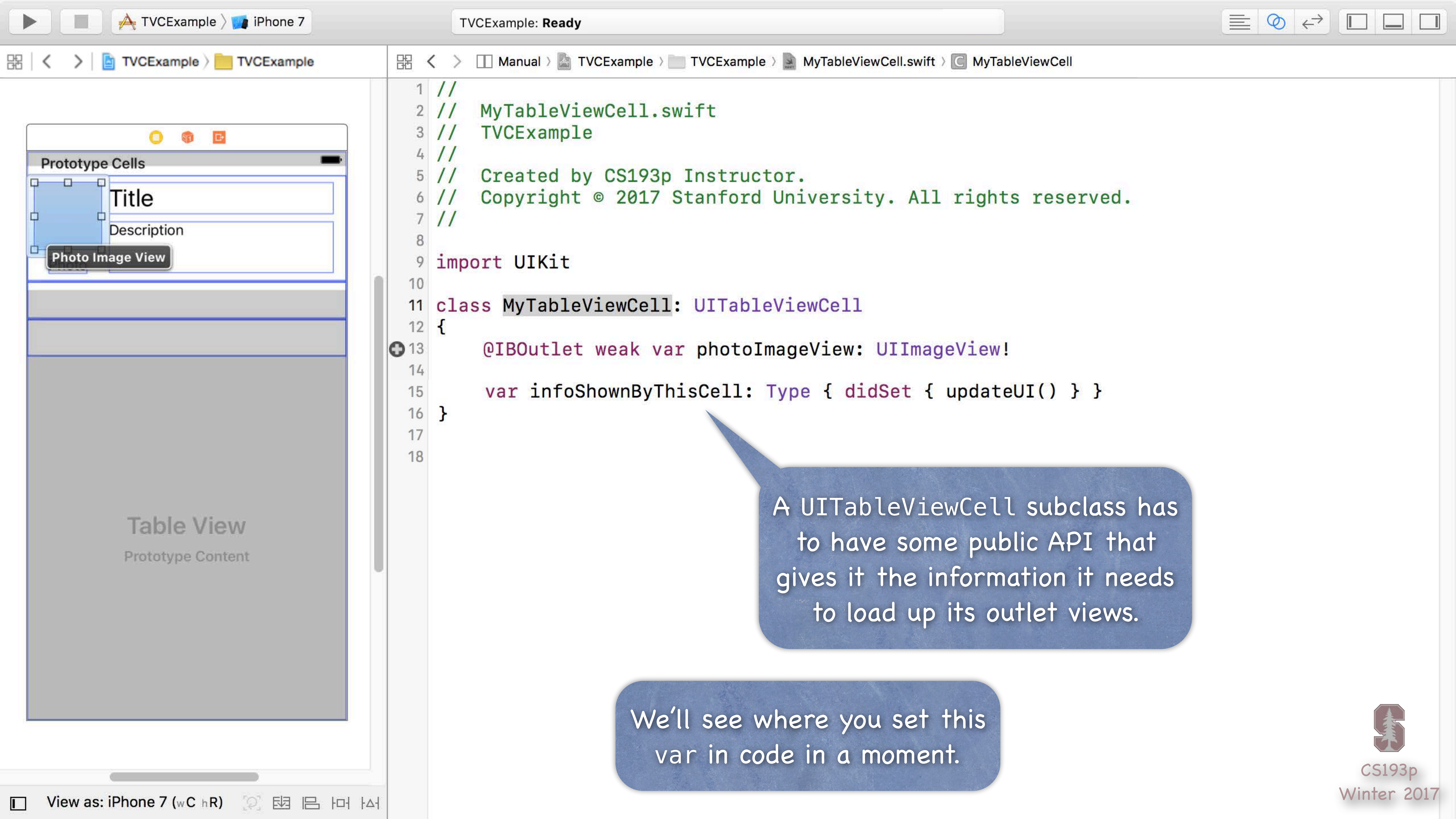
Cancel Connect

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

import UIKit

class MyTableViewCell: UITableViewCell
{


}
```



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

import UIKit

class MyTableViewCell: UITableViewCell
{
    @IBOutlet weak var photoImageView: UIImageView!


}
```

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



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Manual 〉 TVCExample 〉 TVCExample 〉 MyTableViewCell.swift 〉 MyTableViewCell

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

import UIKit

class MyTableViewCell: UITableViewCell
{
    @IBOutlet weak var photoImageView: UIImageView!

    var infoShownByThisCell: Type { didSet { updateUI() } }
}
```

A UITableViewCell 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.

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 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]

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

    return cell
}
```

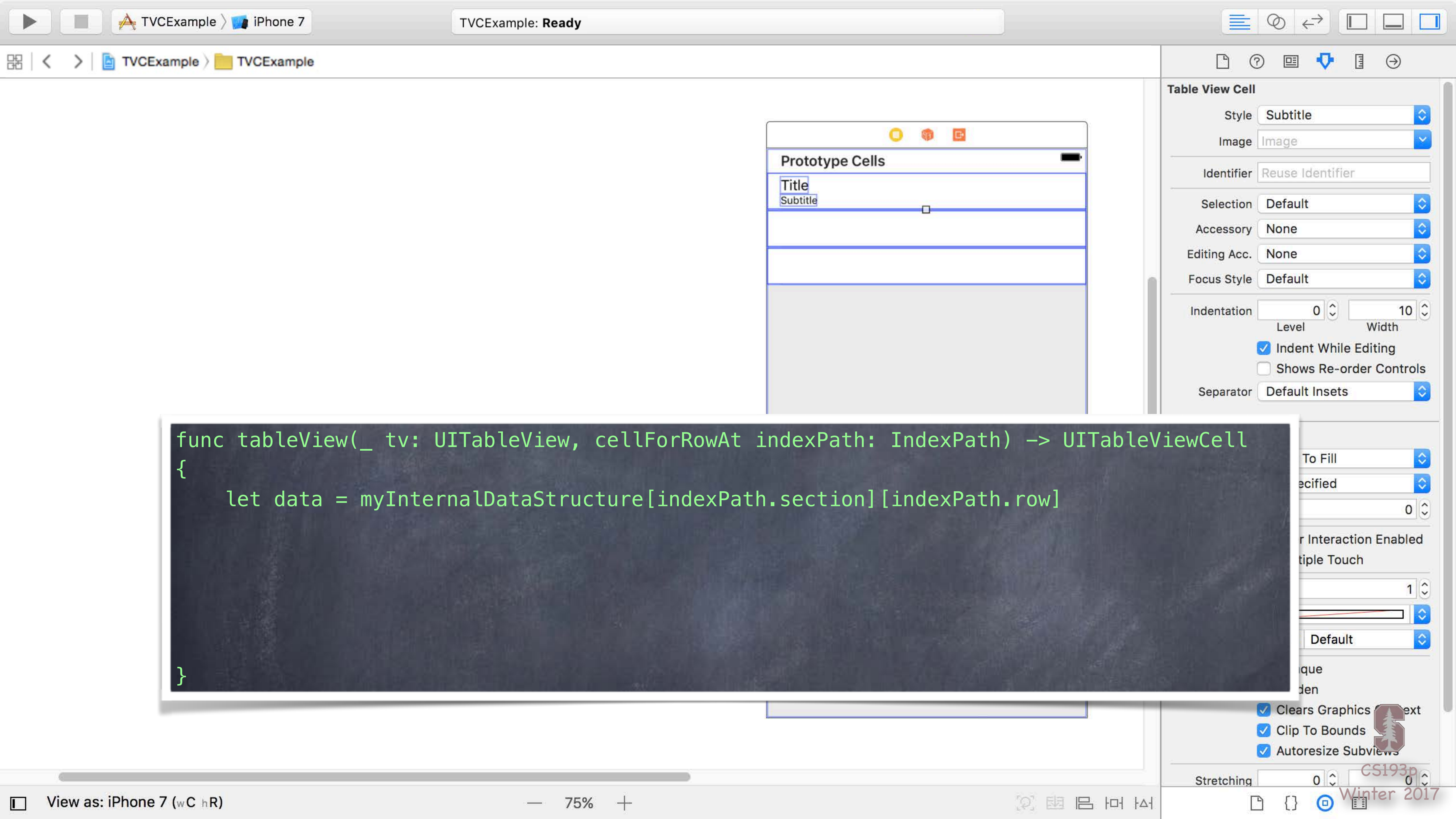
Customizing Each Row

- 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]

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

    return cell
}
}
```


TVCExample › iPhone 7
TVCExample: **Ready**
TVCExample › TVCExample

Prototype Cells
Title
Subtitle

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

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



}
```

View as: iPhone 7 (wC hR)
75%

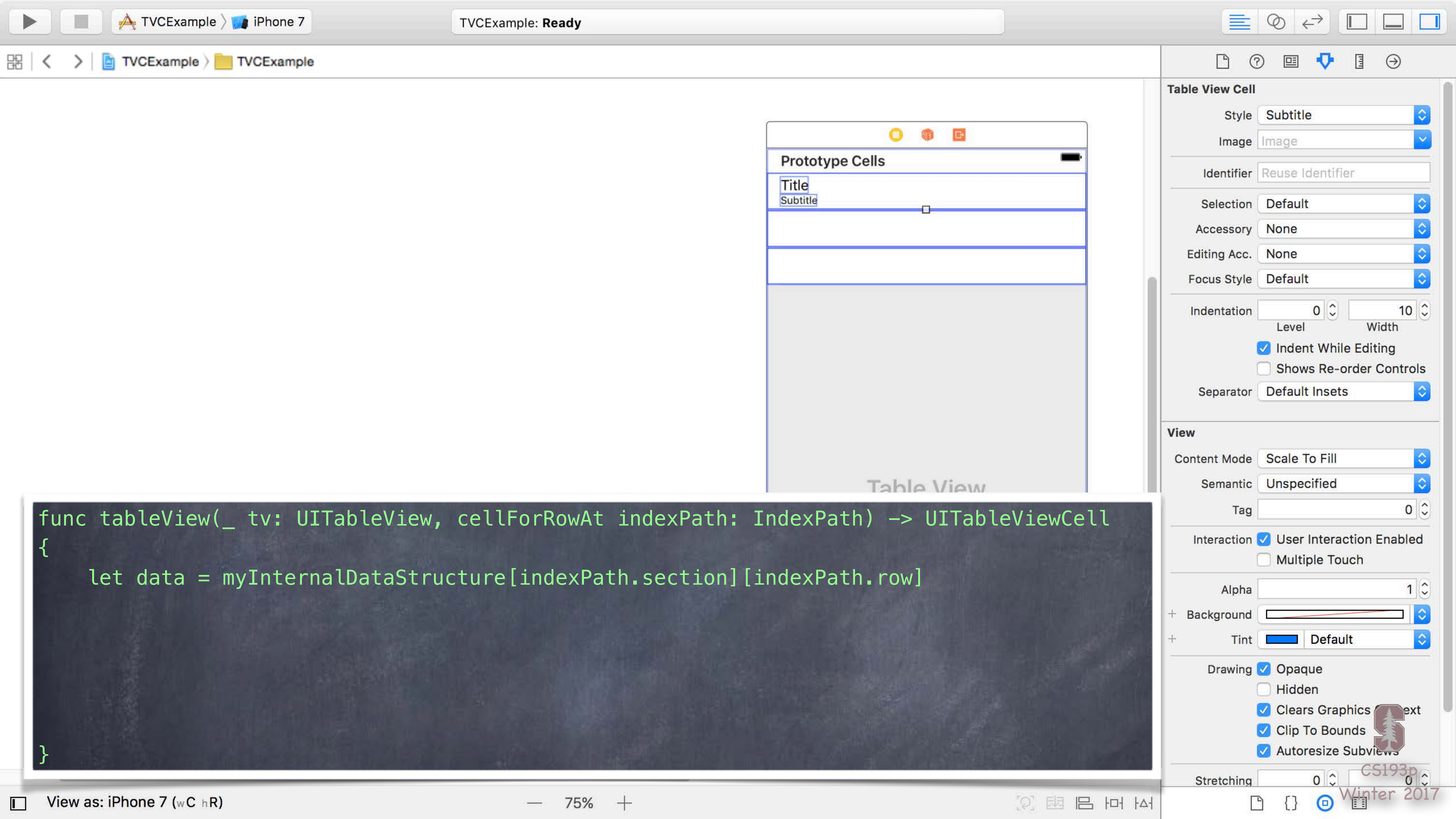


Table View Cell

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
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subviews
Stretching: 0 0

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



}
```

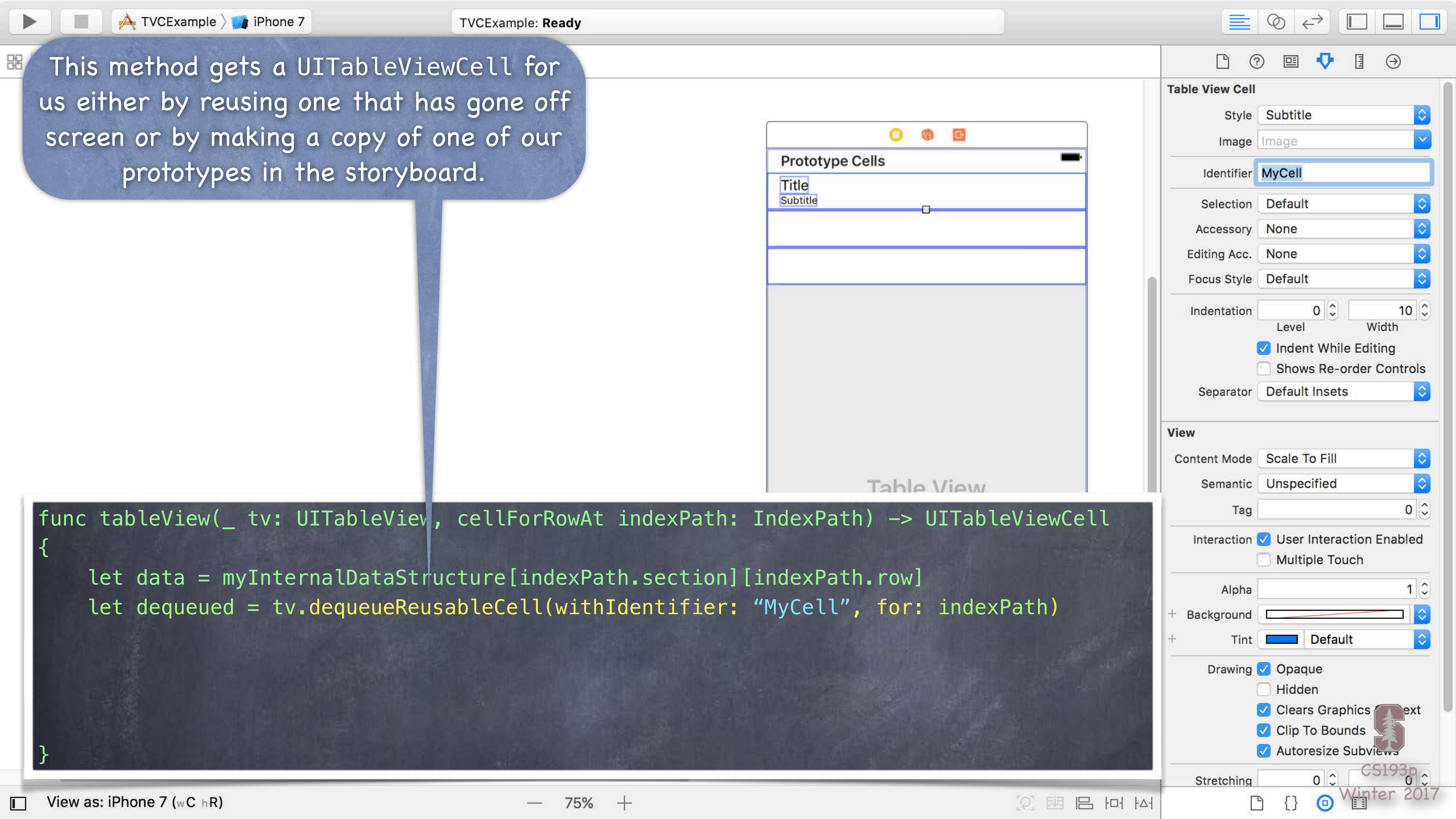
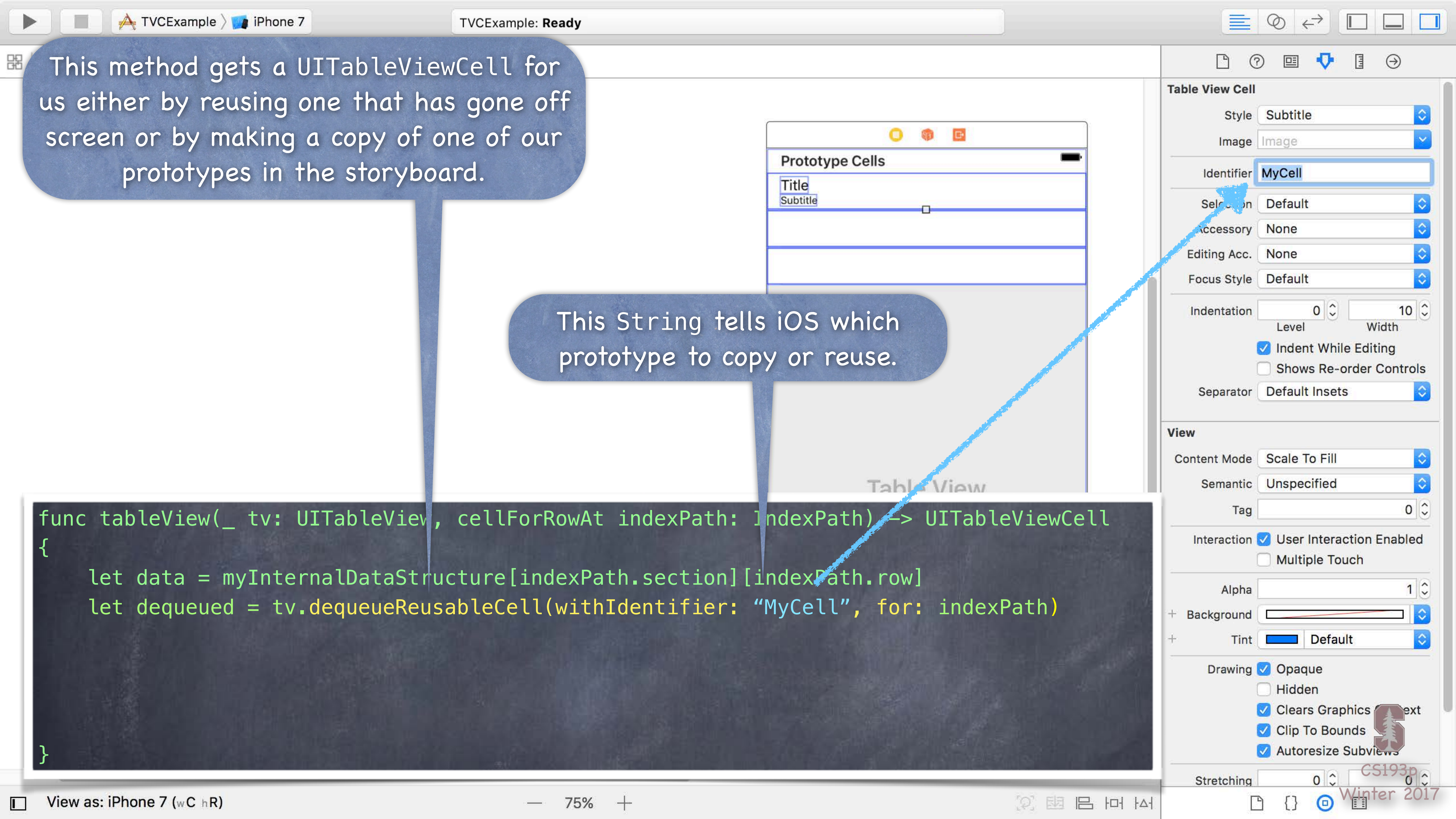


Table View Cell
Style: Subtitle
Image: Image
Identifier: MyCell
Selection: Default
Accessory: None
Editing Acc.: None
Focus Style: Default
Indentation: 0 Level, 10 Width
✓ Indent While Editing
Shows Re-order Controls
Separator: Default Insets

View
Content Mode: Scale To Fill
Semantic: Unspecified
Tag: 0
Interaction: ✓ User Interaction Enabled
Multiple Touch
Alpha: 1
Background
Tint: Default
Drawing: ✓ Opaque
Hidden
✓ Clears Graphics Context
✓ Clip To Bounds
✓ Autoresize Subviews
Stretching: 0 0

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)



}
```



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.

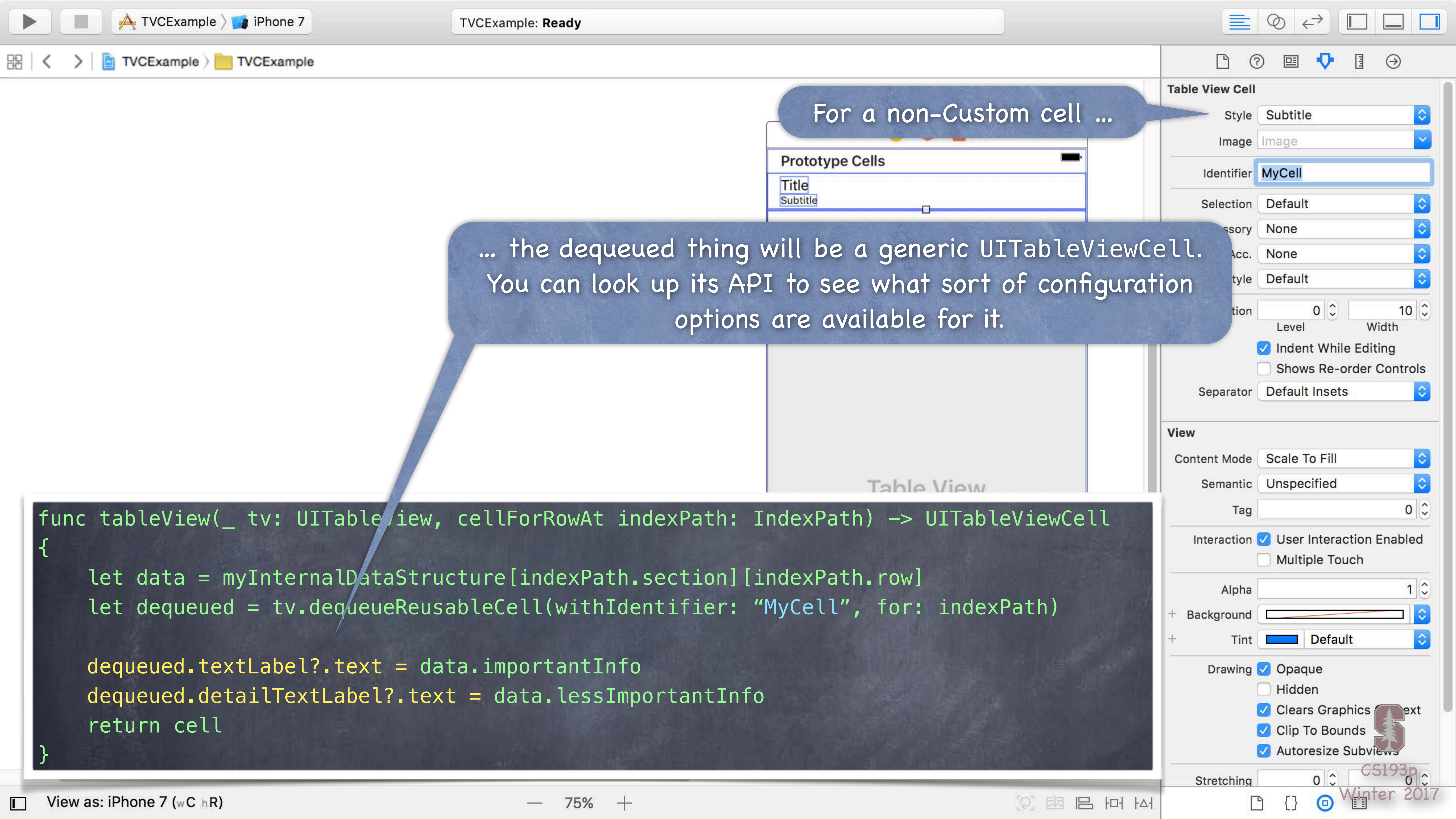
```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: “MyCell”, for: indexPath)


}
```

Table View Cell

Style	Subtitle
Image	Image
Identifier	MyCell
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	
Tint	Default
Drawing	✓ Opaque
	☐ Hidden
	✓ Clears Graphics Context
	✓ Clip To Bounds
	✓ Autoresize Subviews
Stretching	0 0



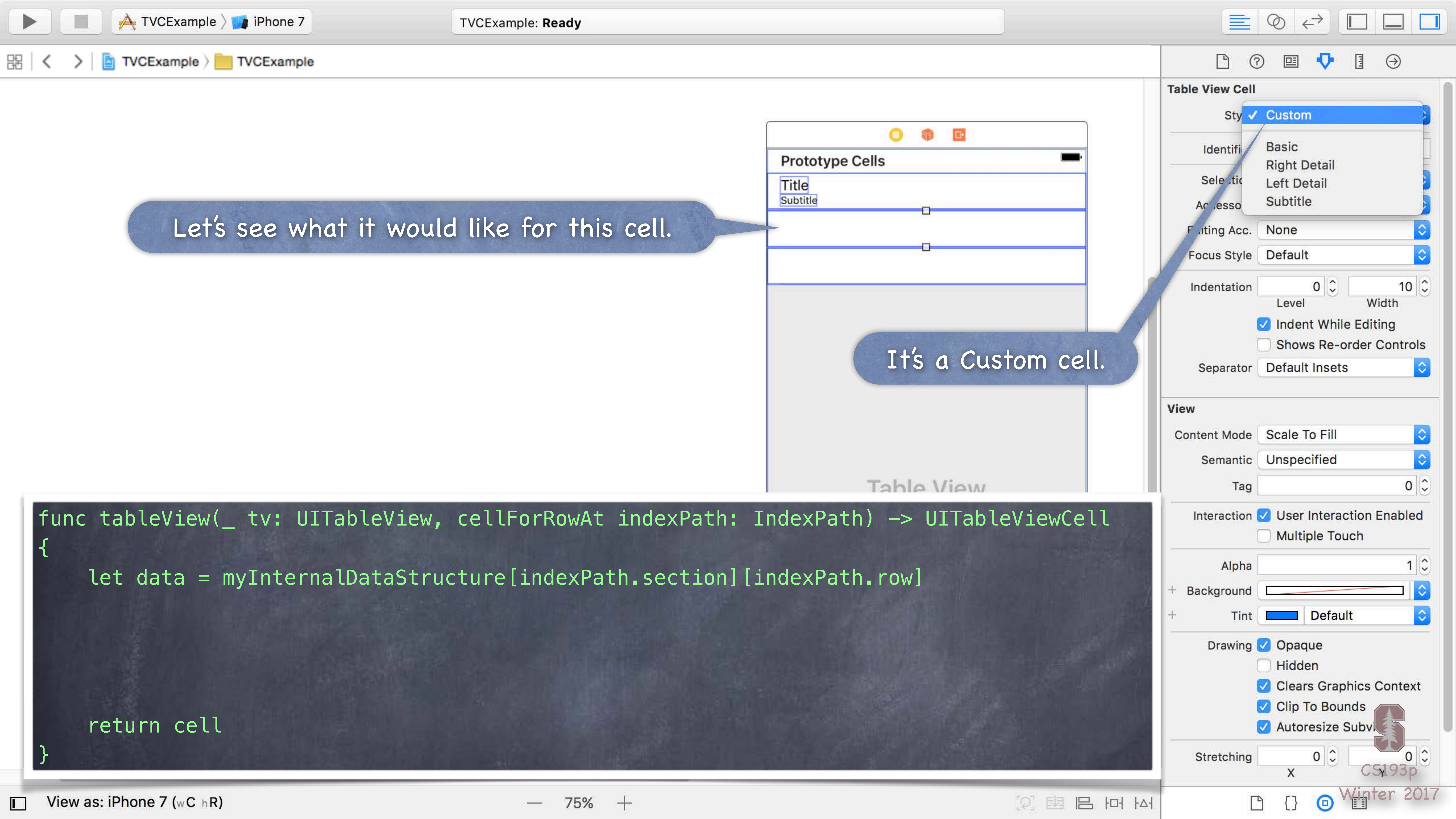
For a non-Custom cell …

… 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
}
```

Table View Cell
Style: Subtitle
Image: Image
Identifier: MyCell
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
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subviews
Stretching: 0 0



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

It's a Custom cell.

Table View Cell

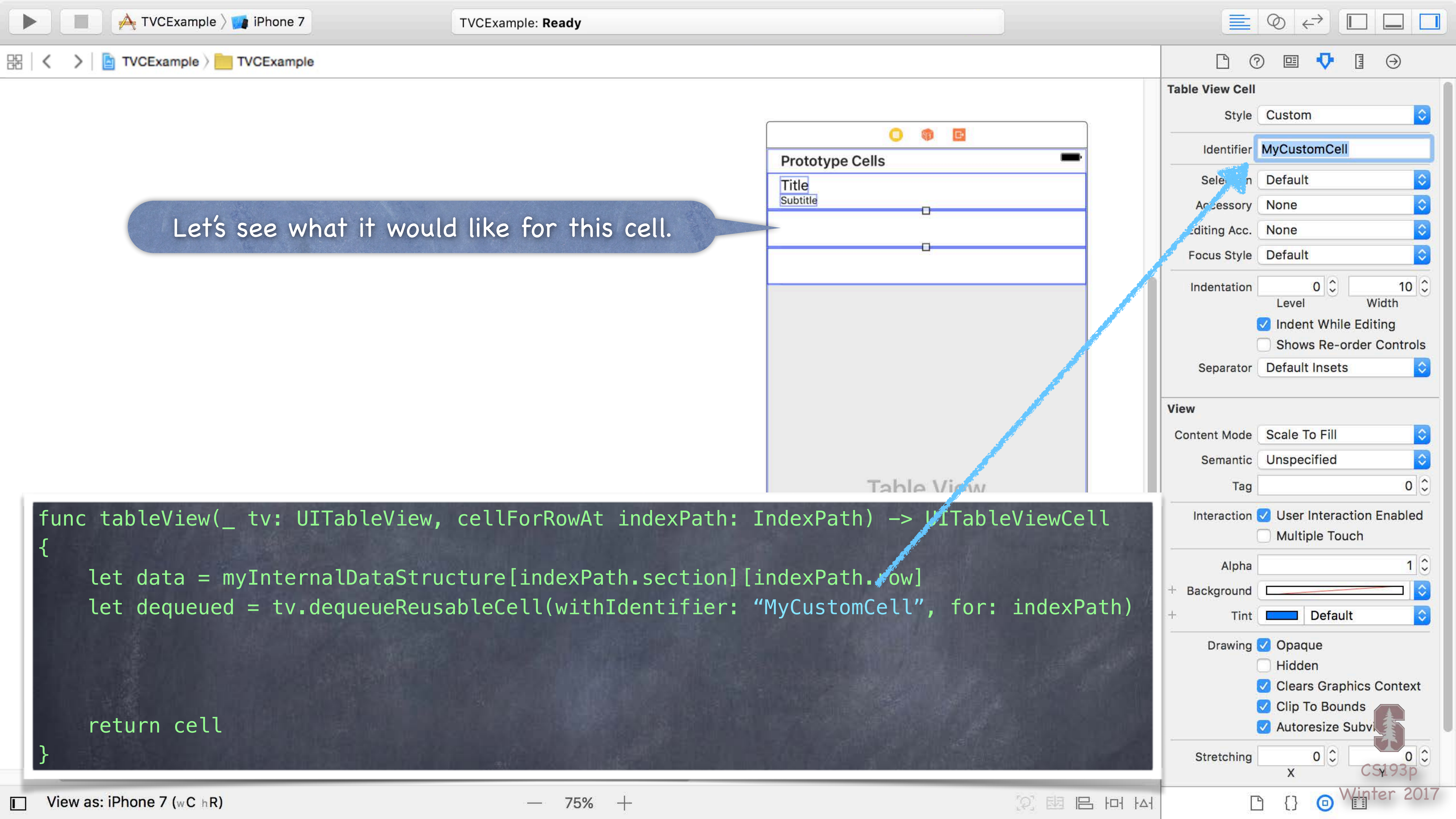
Style: ✓ Custom
Basic
Right Detail
Left Detail
Subtitle

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
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subviews
Stretching: 0 X, 0 Y

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



    return cell
}
```



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

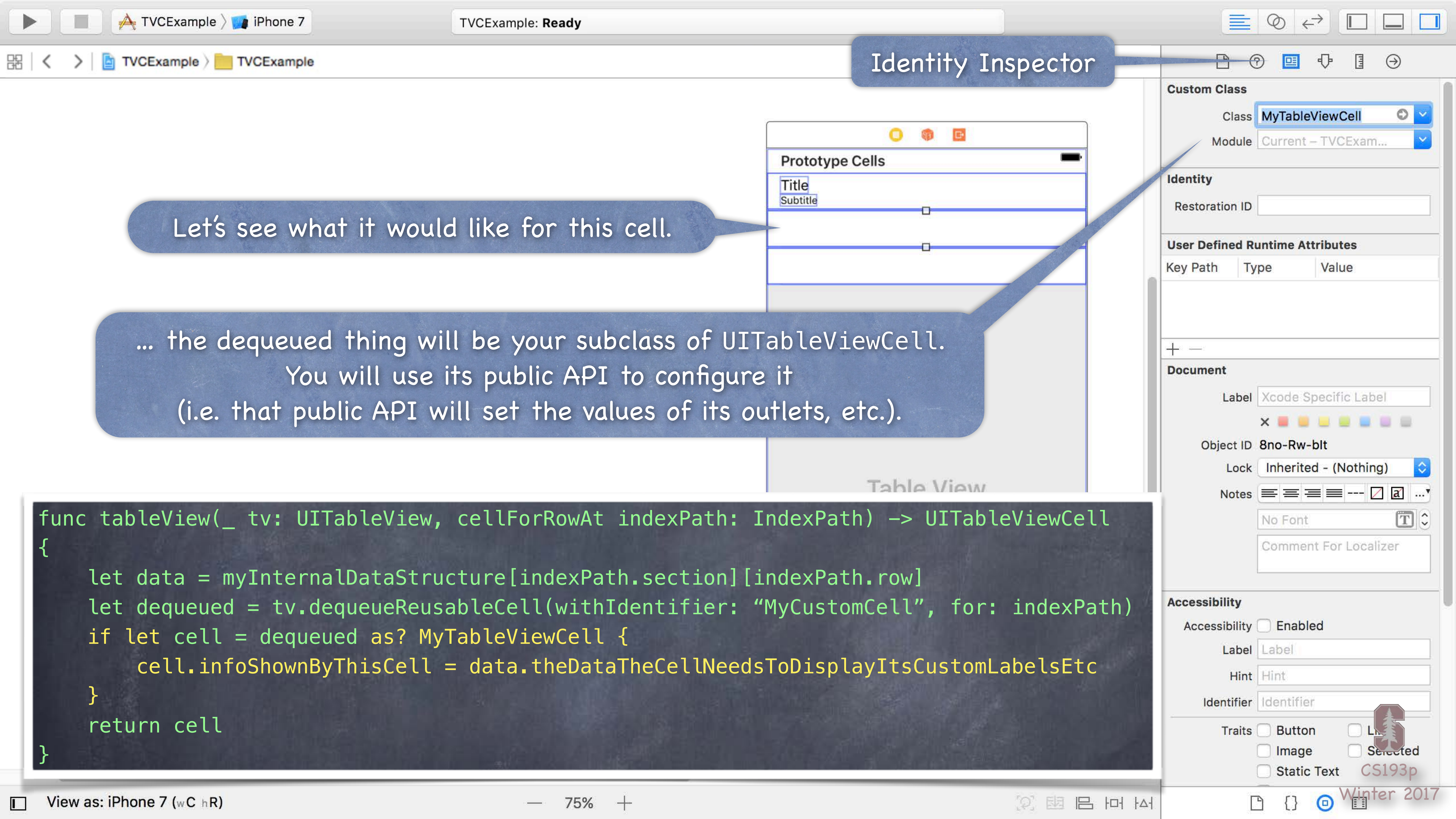
```
func tableView(_ tv: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let data = myInternalDataStructure[indexPath.section][indexPath.row]
    let dequeued = tv.dequeueReusableCell(withIdentifier: “MyCustomCell”, for: indexPath)


    return cell
}
```

Table View Cell

Style: Custom
Identifier: MyCustomCell
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
Tint: Default
Drawing: ✓ Opaque
☐ Hidden
✓ Clears Graphics Context
✓ Clip To Bounds
✓ Autoresize Subvi
Stretching: 0 X, 0 Y



Identity Inspector

Custom Class
Class: MyTableViewCell
Module: Current – TVCExam...

Identity
Restoration ID:

User Defined Runtime Attributes

Key Path	Type	Value

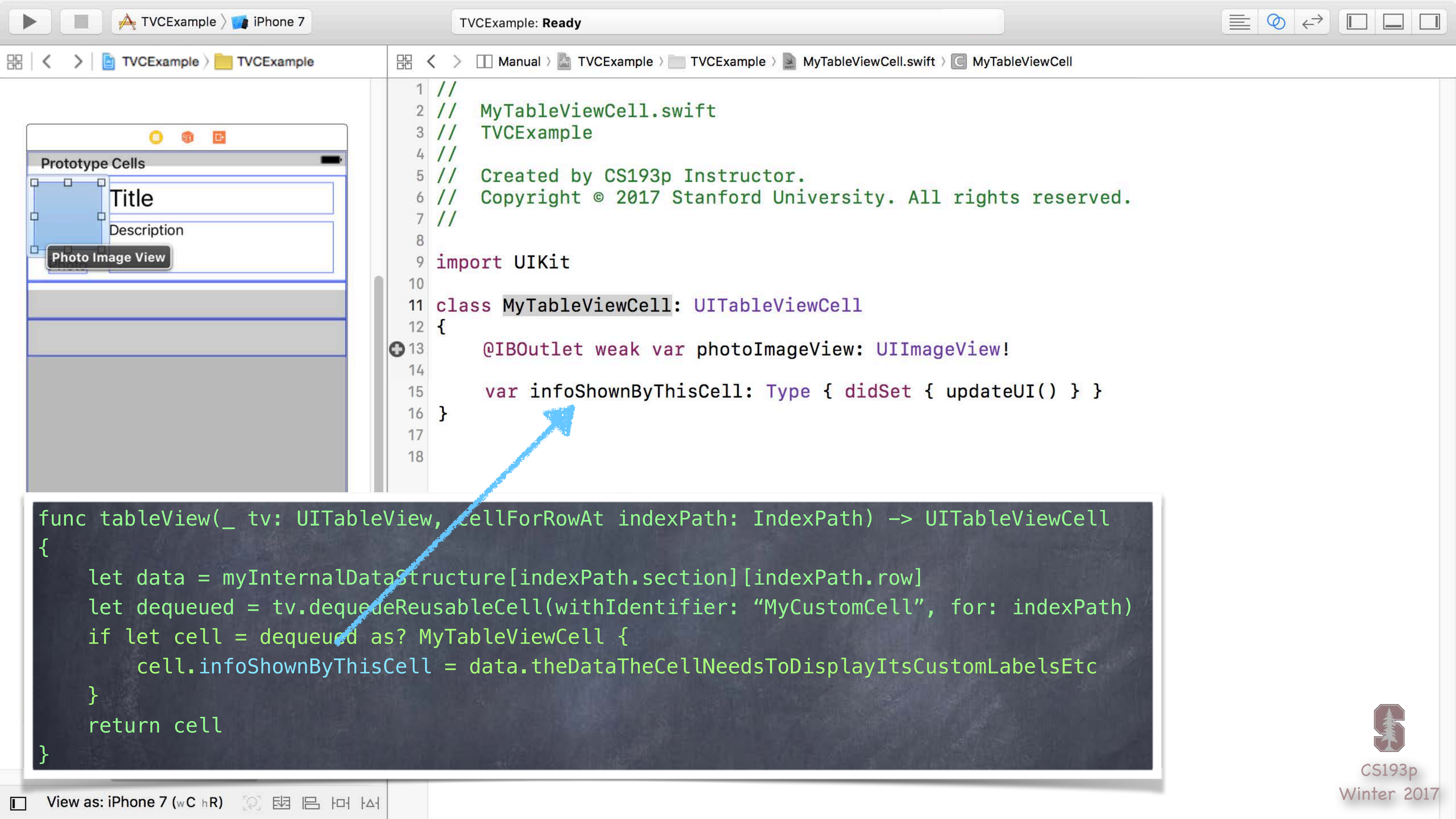
Document
Label: Xcode Specific Label
Object ID: 8no-Rw-blt
Lock: Inherited - (Nothing)
Notes: No Font
Comment For Localizer

Accessibility
Accessibility: Enabled
Label: Label
Hint: Hint
Identifier: Identifier
Traits: Button, Image, Static Text, Selected

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.).

```
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
}
```



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

import UIKit

class MyTableViewCell: UITableViewCell
{
    @IBOutlet weak var photoImageView: UIImageView!

    var infoShownByThisCell: Type { didSet { updateUI() } }
}
```

```swift
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
}
```

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 `cellForRowAt` 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, titleFor{Header,Footer}InSection: Int) -> String`

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

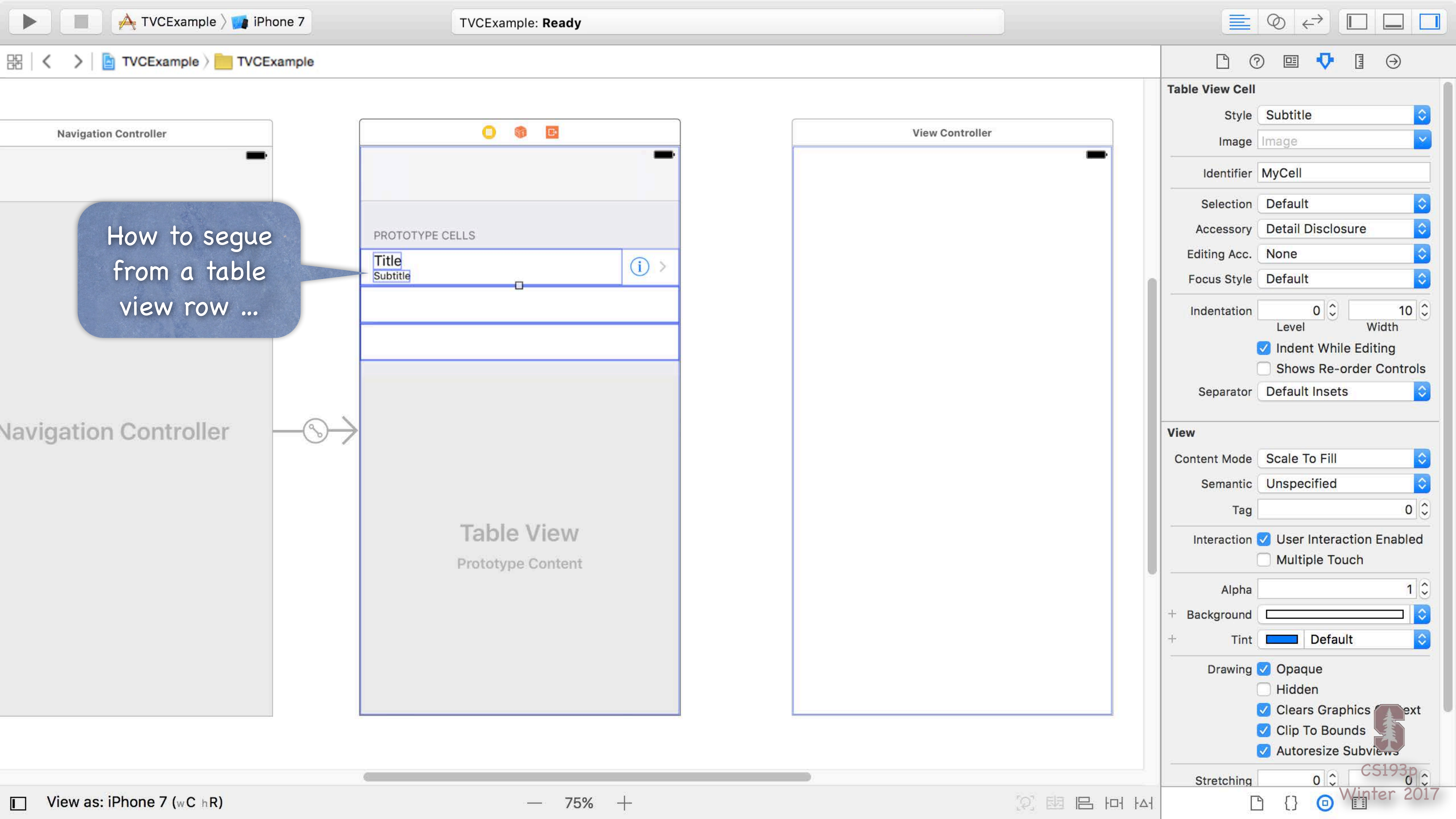
- There are a number of other methods in this `protocol`

  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)



How to segue from a table view row …

TVCExample: **Ready**

TVCExample 〉 TVCExample

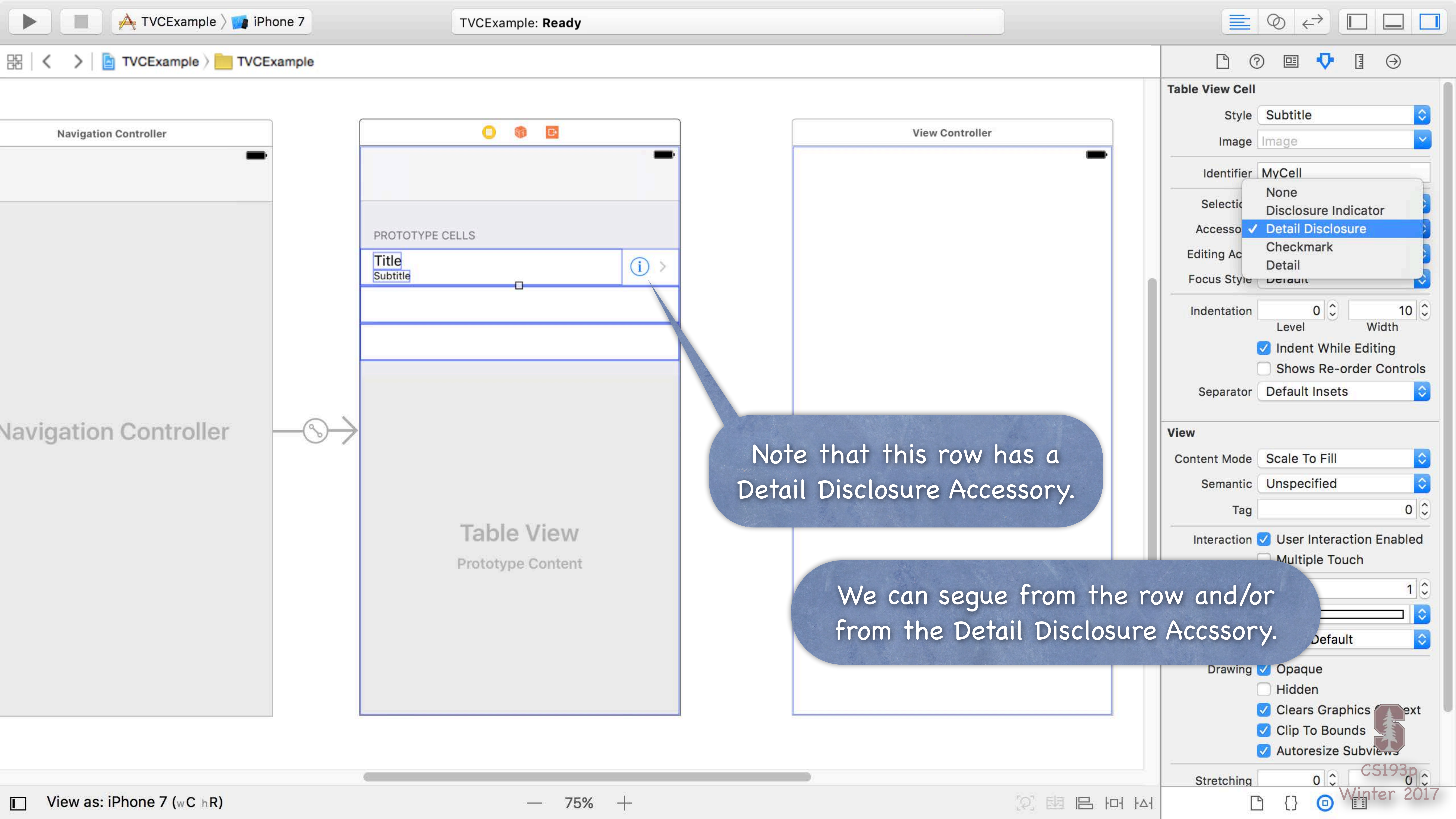
Table View Cell

Style	Subtitle
Image	Image
Identifier	MyCell
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	Default
Drawing	☑ Opaque
	☐ Hidden
	☑ Clears Graphics Context
	☑ Clip To Bounds
	☑ Autoresize Subviews
Stretching	0 0

View as: iPhone 7 (wC hR) 75%



TVCExample 〉 iPhone 7

TVCExample: **Ready**

TVCExample 〉 TVCExample

Navigation Controller

PROTOTYPE CELLS

Title
Subtitle

Table View
Prototype Content

View Controller

Table View Cell

Style	Subtitle
Image	Image
Identifier	MyCell
Selectio	
Accesso	✓ Detail Disclosure
Editing Ac	
Focus Style	Default
Indentation	0 Level, 10 Width
	✓ Indent While Editing
	☐ Shows Re-order Controls
Separator	Default Insets

None
Disclosure Indicator
✓ Detail Disclosure
Checkmark
Detail

View

Content Mode	Scale To Fill
Semantic	Unspecified
Tag	0
Interaction	✓ User Interaction Enabled
	☐ Multiple Touch
	1
	Default
Drawing	✓ Opaque
	☐ Hidden
	✓ Clears Graphics Context
	✓ Clip To Bounds
	✓ Autoresize Subviews
Stretching	0, 0

View as: iPhone 7 (wC hR)

75%

Note that this row has a Detail Disclosure Accessory.

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

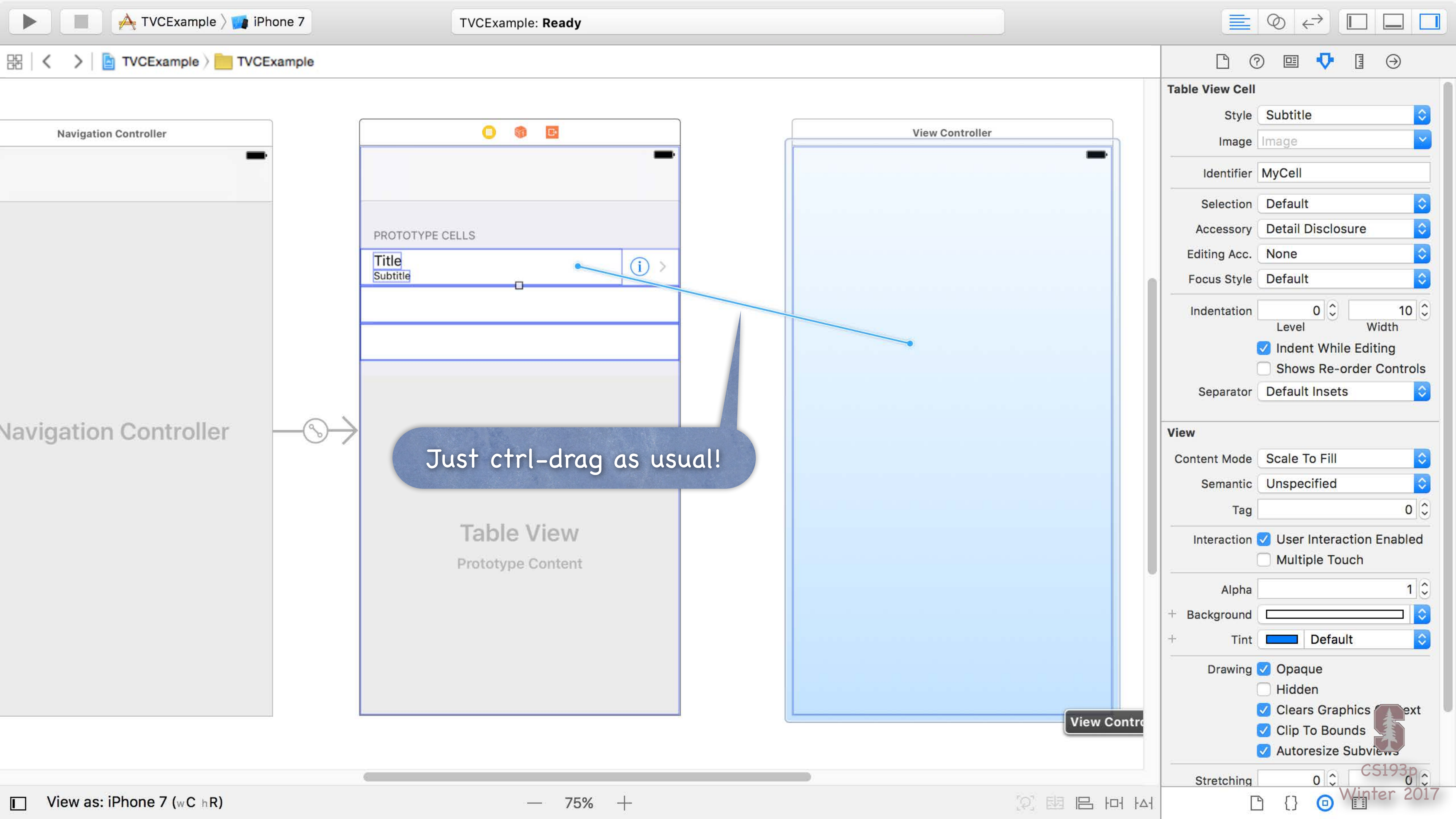
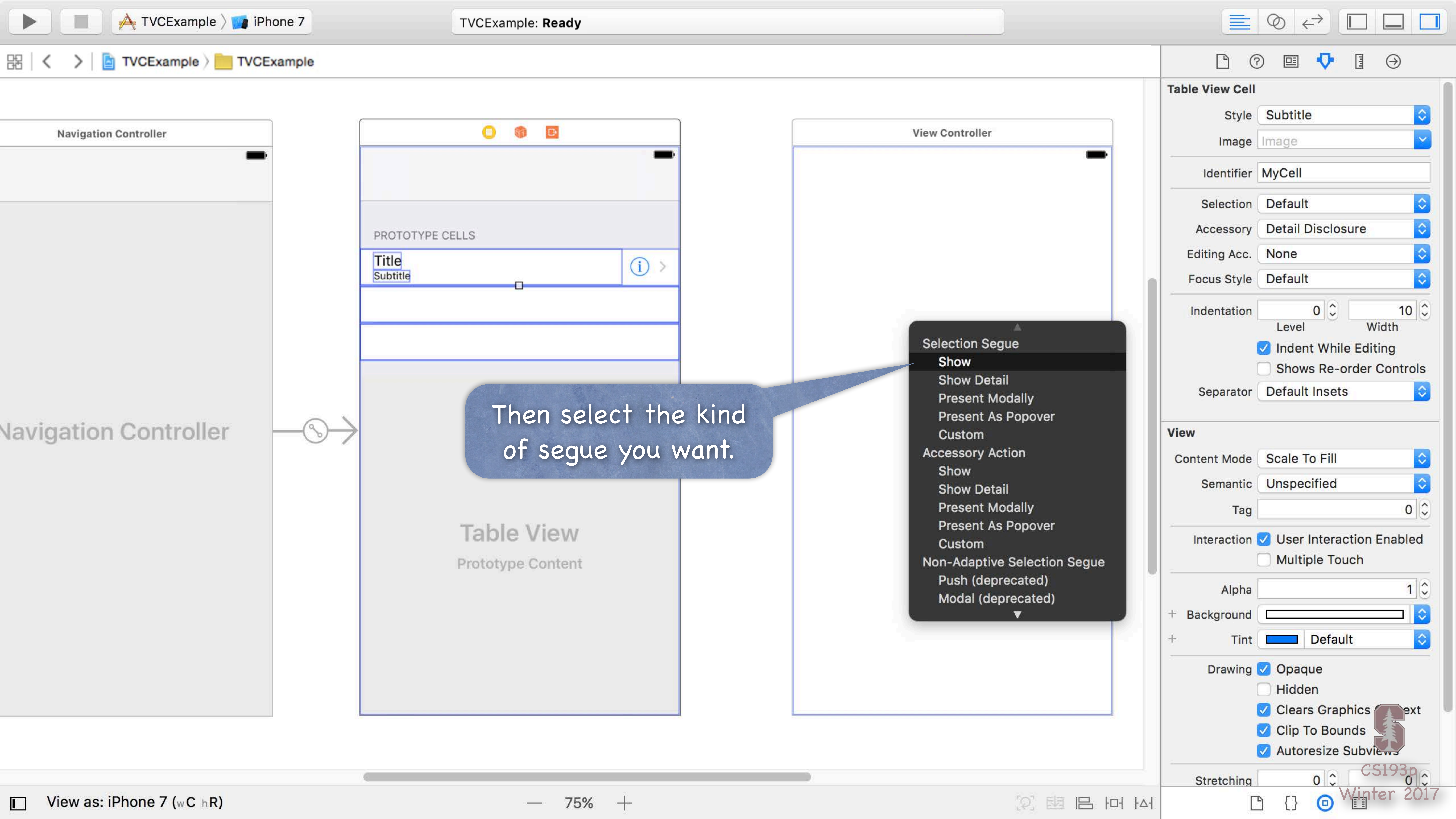


Table View Cell

Style	Subtitle
Image	Image
Identifier	MyCell
Selection	Default
Accessory	Detail Disclosure
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	
Tint	Default
Drawing	✓ Opaque
	☐ Hidden
	✓ Clears Graphics Context
	✓ Clip To Bounds
	✓ Autoresize Subviews
Stretching	0 0



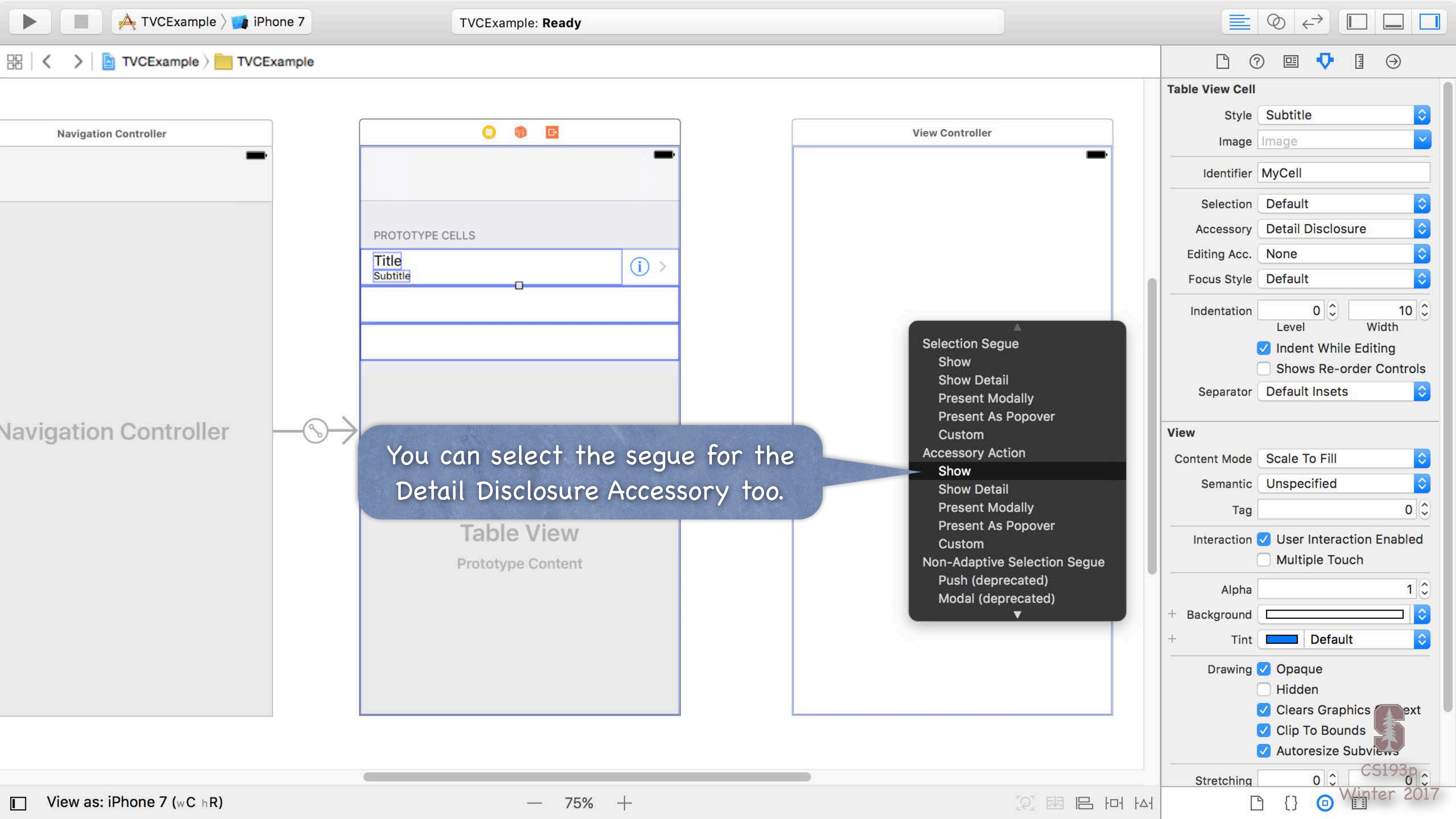
Then select the kind of segue you want.

Table View Cell

Style: Subtitle
Image: Image
Identifier: MyCell
Selection: Default
Accessory: Detail Disclosure
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:
Tint: Default
Drawing: [✓] Opaque
[ ] Hidden
[✓] Clears Graphics Context
[✓] Clip To Bounds
[✓] Autoresize Subviews
Stretching: 0 0



You can select the segue for the Detail Disclosure Accessory too.

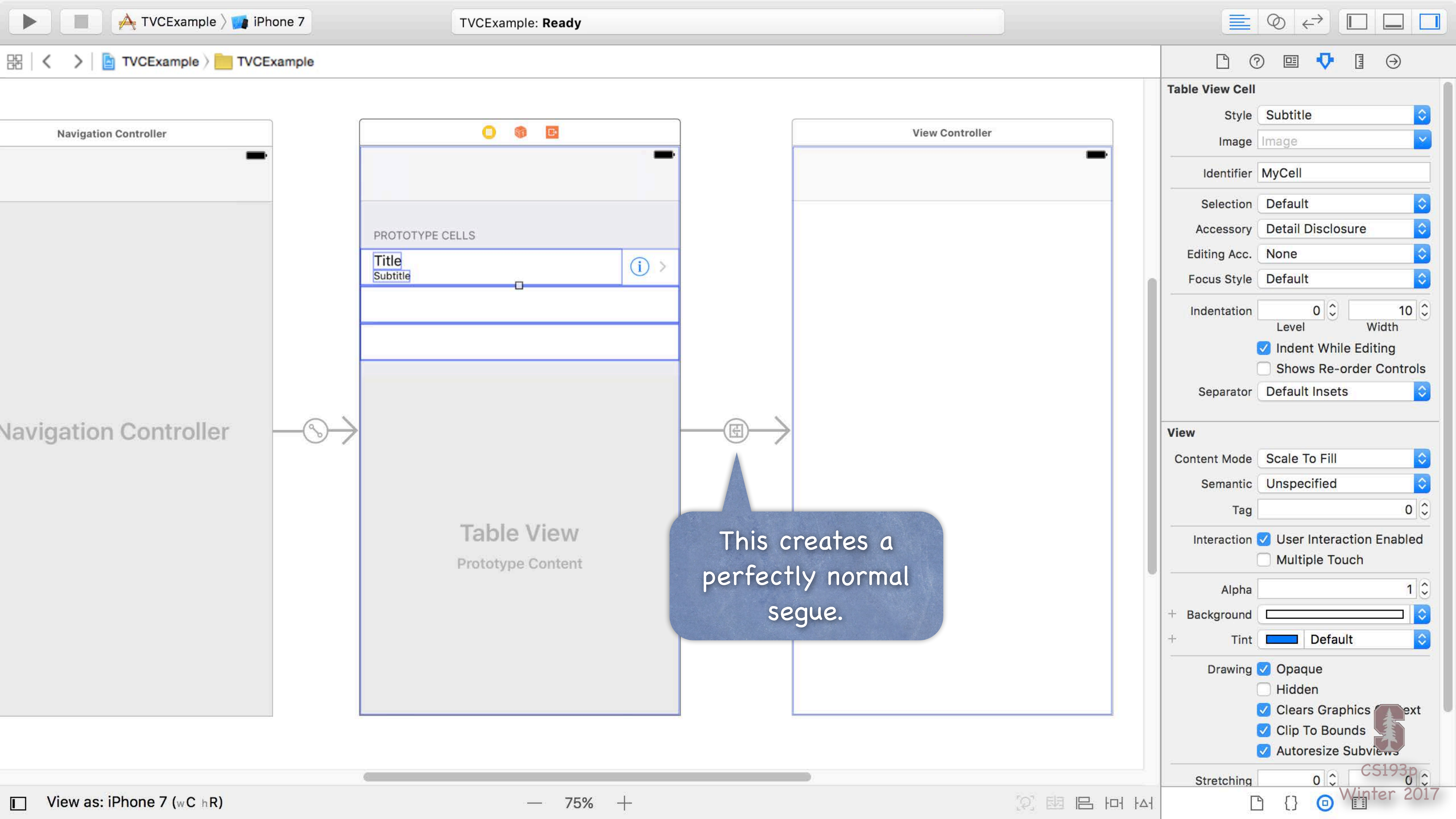
TVCExample: Ready

Table View Cell

Style: Subtitle
Image: Image
Identifier: MyCell
Selection: Default
Accessory: Detail Disclosure
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
Tint: Default
Drawing: ☑ Opaque
☐ Hidden
☑ Clears Graphics Context
☑ Clip To Bounds
☑ Autoresize Subviews
Stretching: 0 0



This creates a perfectly normal segue.

Table View Cell

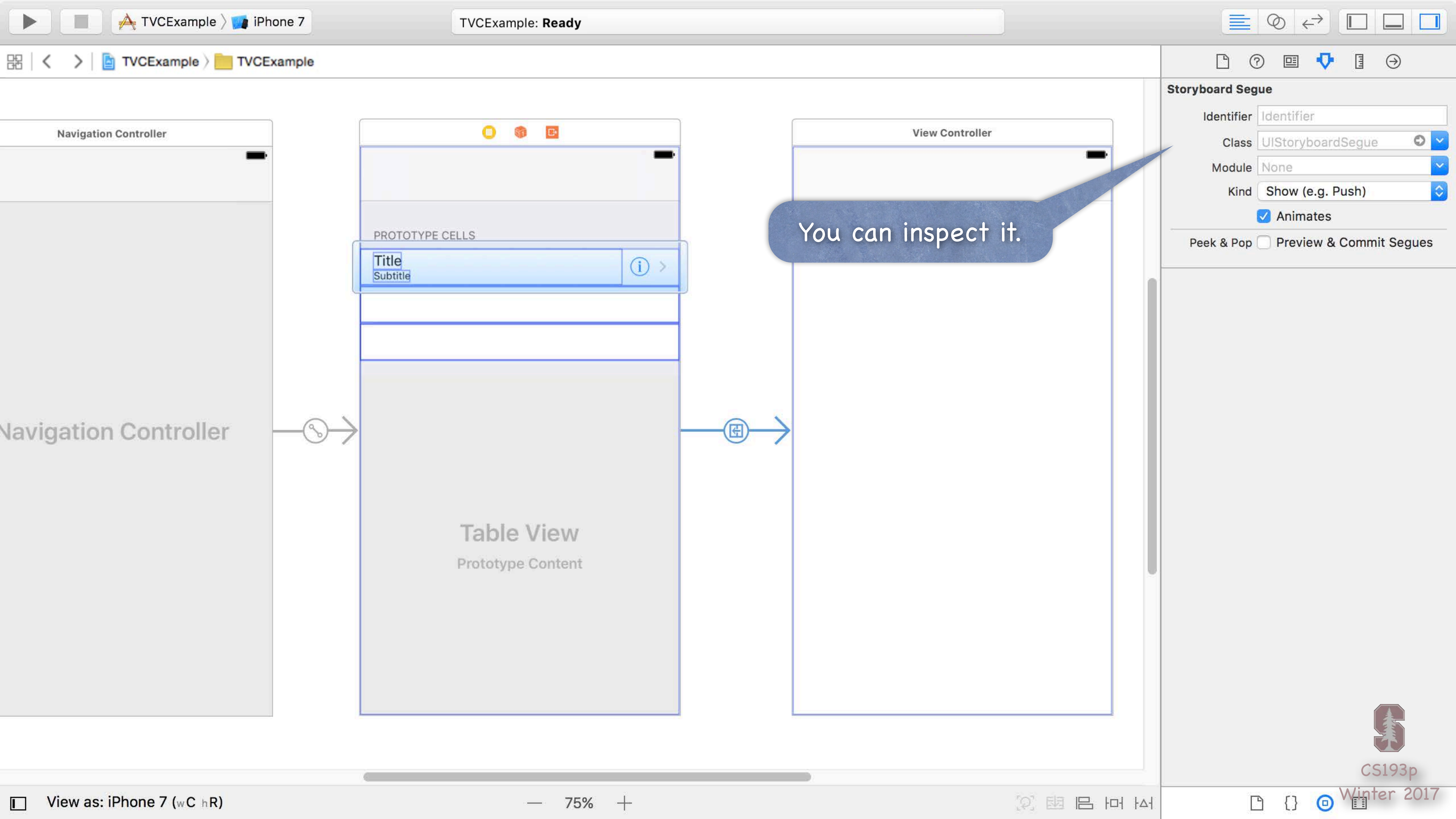
Style	Subtitle
Image	Image
Identifier	MyCell
Selection	Default
Accessory	Detail Disclosure
Editing Acc.	None
Focus Style	Default
Indentation	0 Level, 10 Width

- [✓] Indent While Editing
- [ ] Shows Re-order Controls

Separator	Default Insets

View

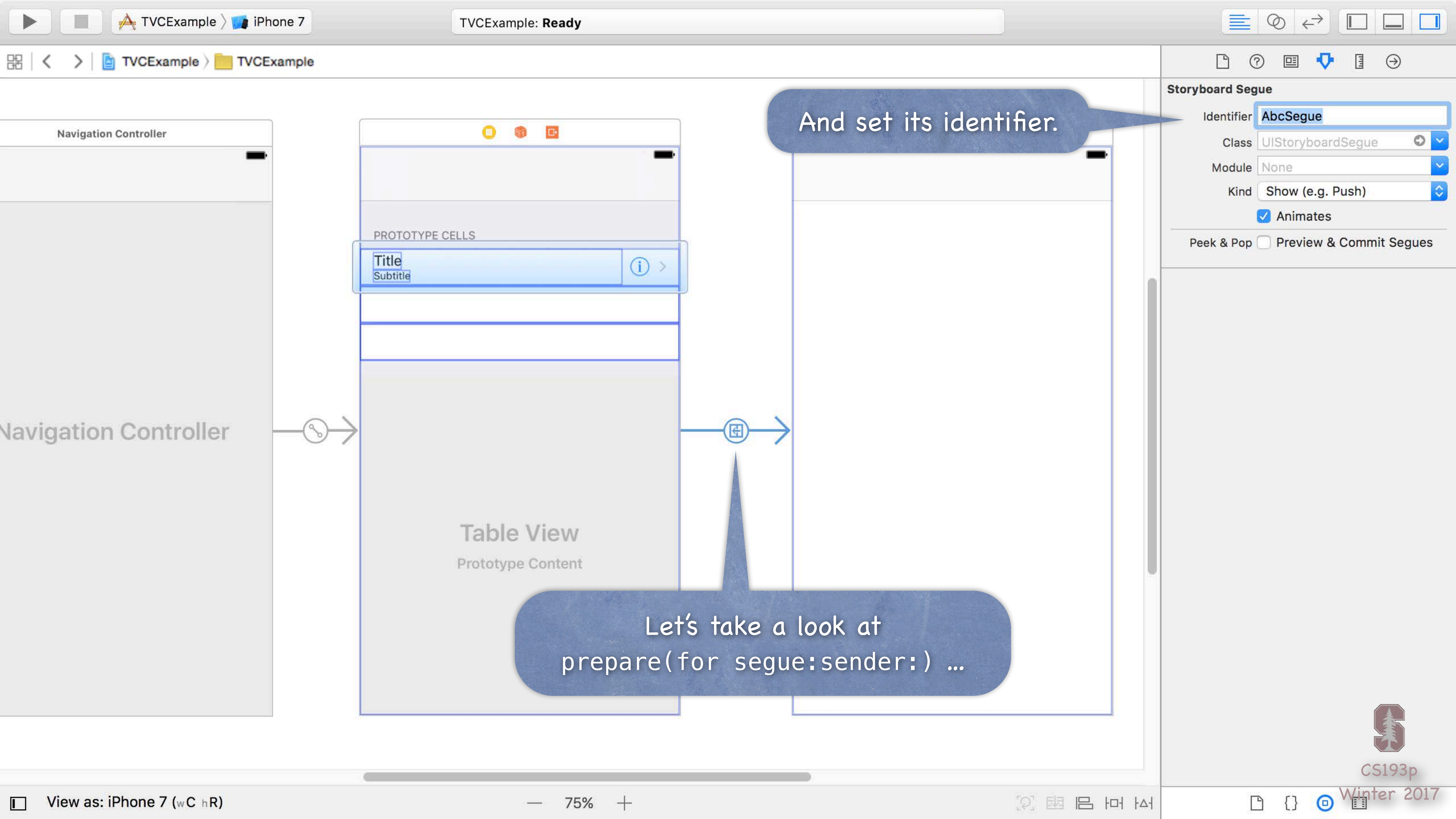
Content Mode	Scale To Fill
Semantic	Unspecified
Tag	0
Interaction	✓ User Interaction Enabled; Multiple Touch
Alpha	1
Background	
Tint	Default
Drawing	✓ Opaque; Hidden; ✓ Clears Graphics Context; ✓ Clip To Bounds; ✓ Autoresize Subviews
Stretching	0 0



You can inspect it.

Storyboard Segue

Identifier	Identifier
Class	UIStoryboardSegue
Module	None
Kind	Show (e.g. Push)
	[✓] Animates
Peek & Pop	[ ] Preview & Commit Segues



And set its identifier.

Storyboard Segue

Identifier: AbcSegue
Class: UIStoryboardSegue
Module: None
Kind: Show (e.g. Push)
[✓] Animates
Peek & Pop: [ ] Preview & Commit Segues

Let's take a look at prepare(for segue:sender:) …

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”:





        default: break
        }
    }
}
```

  You can see now why sender is Any

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

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 {



            }
        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

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) {


            }
        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 …

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

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

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 `cellForRowAt` 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)`

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

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