Introducing

Stack Views

Hands-On Challenges

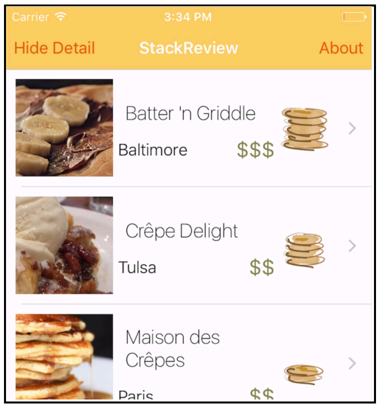
Introducing Stack Views Hands-On Challenges

Challenge 6: Final Challenge

Your final challenge in this stack view series will bring everything you've learned across the entire series together.

In the very first tutorial you created a custom layout for a table cell, and in this final challenge you're going to head right back there to customize it further.

Your challenge is to update the layout of the table view cell to match the following design:



Notice the addition of the city name and price indicator to the cell.

Although this would be a challenge in itself, since this is the final challenge you're going to push yourself. Take a look at the navigation bar, and you'll see a new button on the left hand side – **Hide Detail**.

When the user taps this button, its title should switch to Show Detail, and the additional elements you added to the table cells should disappear. Obviously you'll want to animate this disappearance.

Good luck!

Hints

• You'll need **two** new stack views to create the correct layout within the table cell.

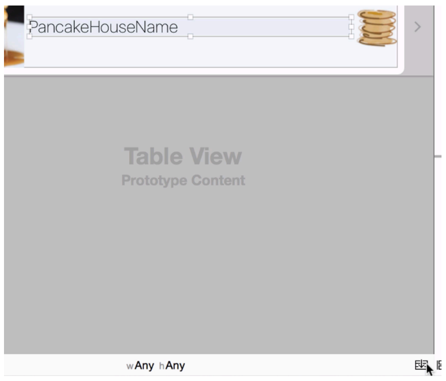
• Implement the animation within the custom cell subclass, and then update each cell from the master view controller when the user taps the button.

Solution

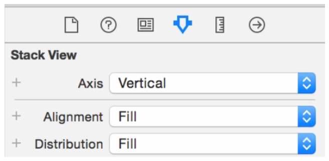
First up, you're going to create the new layout. Once that's done you can turn your attention to implementing the animation.

The new labels both appear underneath the pancake house name label in the table cell. That sounds exactly like you need a new stack view. Guess what… you do!

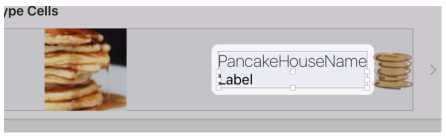
Open **Main.storyboard** and locate the master view controller scene. Select the **pancakeHouseName** label and use the **Stack** menu to create a new stack view:



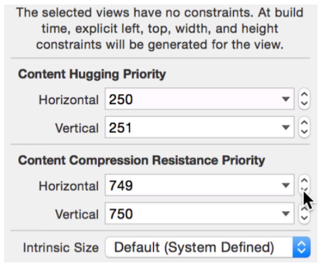
Check that this new stack view has a **Vertical** axis:


Stack View
Axis: Vertical
Alignment: Fill
Distribution: Fill

Now you can add the first of the new labels. Drag a label from the object catalog into this new stack view.



Notice that the layout of the cell is now broken. This is because the outer stack view doesn't think that it's allowed to stretch this new label to fill the cell. To fix this, open the **size inspector** and set the **Horizontal Content Hugging Priority** to **250** and the **Horizontal Content Compression Resistance Priority** to **749**:

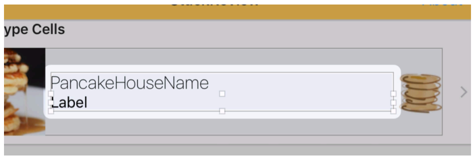

The selected views have no constraints. At build time, explicit left, top, width, and height constraints will be generated for the view.

Content Hugging Priority
Horizontal: 250
Vertical: 251

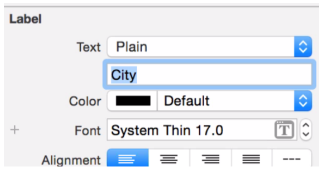
Content Compression Resistance Priority
Horizontal: 749
Vertical: 750

Intrinsic Size: Default (System Defined)

You'll see the layout update to match what you expect:

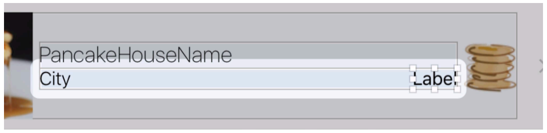


Update the content of the label to say **"City"** and the font to be **System Thin 17.0**.

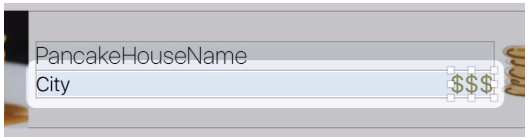

Label
Text: Plain
City
Color: Default
Font: System Thin 17.0
Alignment

Now you can move your attention to the second new label – the price indicator. This label sits alongside the city label – so you need another stack view.

Select the **City** label, and use the **Stack** menu to create a new stack view. Change the **axis** of this new stack view to be **Horizontal**. Then drag a new label from the object catalog to the right hand side of this new stack view:

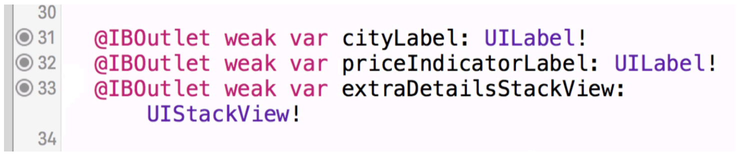


Update this label to contain **"$$$"**, change the font to **System 20.0** and the color to hex value **#989461**:



Now that you've created these new labels, you can wire them up to the underlying class.

Open the assistant editor and select **PancakeHouseTableViewCell.swift** from the jump bar. Then, by control-dragging from the storyboard to the assistant editor, create new outlets for the the two new labels, and the inner (horizontal) stack view. Call them `cityLabel`, `priceIndicatorLabel` and `extraDetailsStackView`:


```
@IBOutlet weak var cityLabel: UILabel!
@IBOutlet weak var priceIndicatorLabel: UILabel!
@IBOutlet weak var extraDetailsStackView:
    UIStackView!
```

To ensure that the new labels get correctly populated with content, add the following inside the `if let` clause of the `didSet` closure for the `pancakeHouse` property:

```
cityLabel?.text = pancakeHouse.city
priceGuideLabel?.text = "\(pancakeHouse.priceGuide)"
```

That's it for creating the new cell layout – it's time to move on to animating the disappearance of these new labels.

Each `PancakeHouseTableViewCell` needs to know whether this `extraDetailsStackView` should be visible or not. To accomplish this, you'll add a new property. Add the following code to the `PancakeHouseTableViewCell` class:

```
var showExtraDetails : Bool = false {
  didSet {
    extraDetailStackView.hidden = !showExtraDetails
  }
}
```

This is a simple boolean property, that includes a `didSet` clause which updates the hidden property of the `extraDetailStackView`.

As well as being able to set this property, you want to be able to animate it nicely. You'll do this by adding a new function.

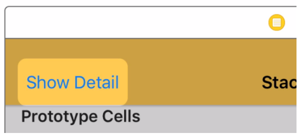
Add the following method to the class:

```
func animateShowExtraDetails(show: Bool) {
  UIView.animateWithDuration(1.0) {
    self.showExtraDetails = show
  }
}
```

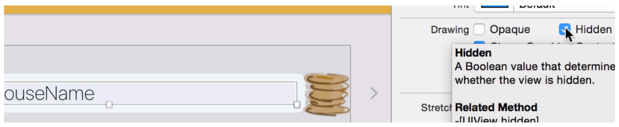
This is a really simple `UIView` animation, which updates the `showExtraDetails` property. Since this involves changing the `hidden` property on a subview of a stack view, it'll animate the appearance and disappearance.

That's all the changes you need to make in the table view cell. You can head back to the storyboard to add the new button to the navigation bar.

Find the master view controller scene in the storyboard and drag a **Bar Button Item** from the object library to the left hand side of the navigation bar. Change its title to "**Show Details**":



Since the default will be to have the extra details hidden when the app starts, locate the Extra Details Stack View and check Hidden in the attributes inspector:



Hidden
A Boolean value that determine
whether the view is hidden.
Related Method
-[UIView hidden]

Using the assistant editor add a new outlet and action for the “Show Details” bar button. Call them `showHideDetailButton` and `handleShowHideDetailTapped` respectively.

You need a way of keeping track of whether or not the extra details are currently visible or not. Add the following property to the `MasterViewController`:

```
private var detailsHidden : Bool = true {
  didSet {
    for cell in tableView.visibleCells {
      if let cell = cell as? PancakeHouseTableViewCell
        where cell.showExtraDetails == detailsHidden {
          cell.animateShowExtraDetails(!detailsHidden)
      }
    }
  }
}
```

This is a boolean property which includes a `didSet` closure which loops through the visible cells animating their `showExtraDetails` property to match the new value.

Changing this new property will handle all the animations for you, so the implementation of the button tap handler is really simple. Update it to match the following:

```
@IBAction func handleShowHideDetailsTapped(sender: AnyObject) {
  if detailsHidden {
    showHideDetailsButton.title = "Hide Detail"
  } else {
    showHideDetailsButton.title = "Show Detail"
  }
  detailsHidden = !detailsHidden
}
```

This method updates the title of the button, and then toggles the new detailsHidden property.

The final thing you need to do is ensure that when new cells are dequed, they have the correct value on their `showExtraDetails` property.

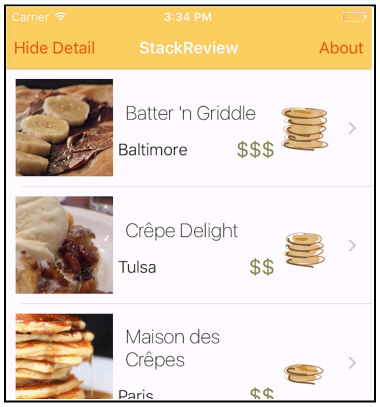
Find the `tableView(_:, cellForIndexPath:)` method, and update the cell type test to match the following:

```
if let cell = cell as? PancakeHouseTableViewCell {
  cell.pancakeHouse = pancakeHouse
  cell.showExtraDetails = !detailsHidden
} else {
  cell.textLabel?.text = pancakeHouse.name
}
```

Notice that you've added a single line to set the `showExtraDetails` property on the cell appropriately.

That's it! You've done it! Well done.

Build and run to check your work. By default you won't see any difference in the cells, but when you tap the new "Show Details" button each cell will animate the new labels in:


Carrier
3:34 PM
Hide Detail
StackReview
About
Batter 'n Griddle
Baltimore
$$$
Crêpe Delight
Tulsa
$$
Maison des Crêpes
Paris
$$