Custom Collection View Layouts

Hands-on Challenges

Custom Collection View Layouts Hands-on Challenges

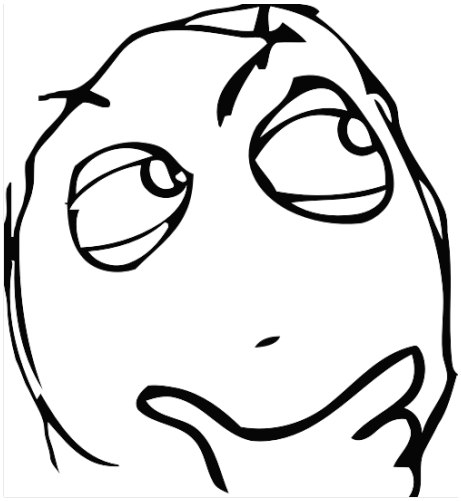
Challenge F: Counter Rotation

Just like with the Ultravisual layout, the Timbre layout is being used to display session details from RWDevCon, and each session has a corresponding background photo that's not yet being displayed. Let's change that now!

Your challenge this time is to add an image view to the cell, and update the `UICollectionViewCell` subclass to populate the image view with the corresponding background photo whenever the `tutorial` property is set.

Debugging the View Hierarchy

However, there is a catch. The photos shouldn't be rotated, so you'll need to figure out how to counter the rotation of cell, and where the best place to do that is.

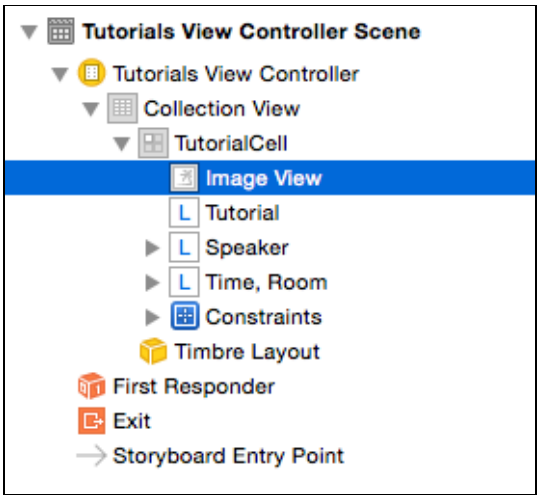


Before you turn the page for our solution, be sure to give it a try for yourself first!

Solution

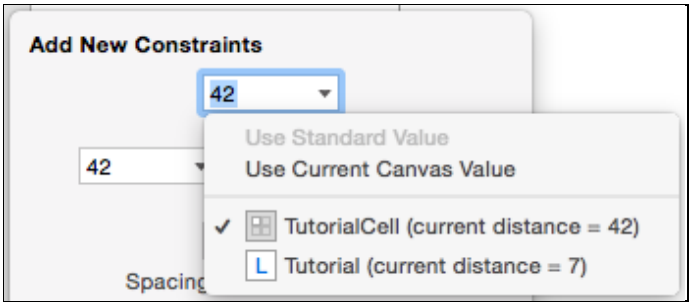
Adding the image view

Open **Main.storyboard** and drag an **Image View** from the Object Library on to **TutorialCell** in the Document Outline, making sure it's the first subview in the list so it's positioned underneath the rest of the subviews:

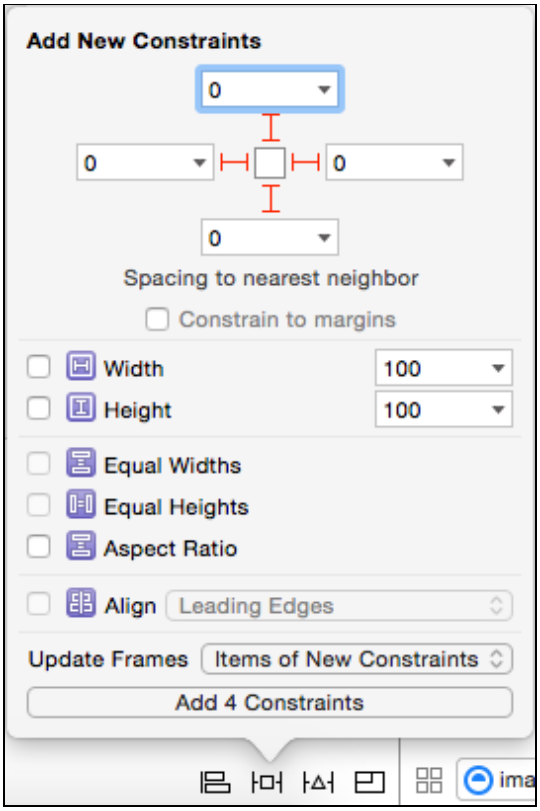


With **Image View** selected, in the Attributes inspector change View\Mode to **Aspect Fill** so the photos don't get stretched and warped when they're displayed.

Next, click the **Pin** button at the bottom of the storyboard canvas. For the Leading, Top, Trailing, and Bottom spacing, make sure they're referencing **TutorialCell** and not Tutorial:



Then uncheck **Constrain to margins** and set all four of their values to `0`. Change Update Frames to **Items of New Contraints** and click **Add 4 Constraints**:



With the image view now set up, it's time to create an outlet so the image can be set programmatically.

Open **TutorialCell.swift** and add the following just below the existing outlets:

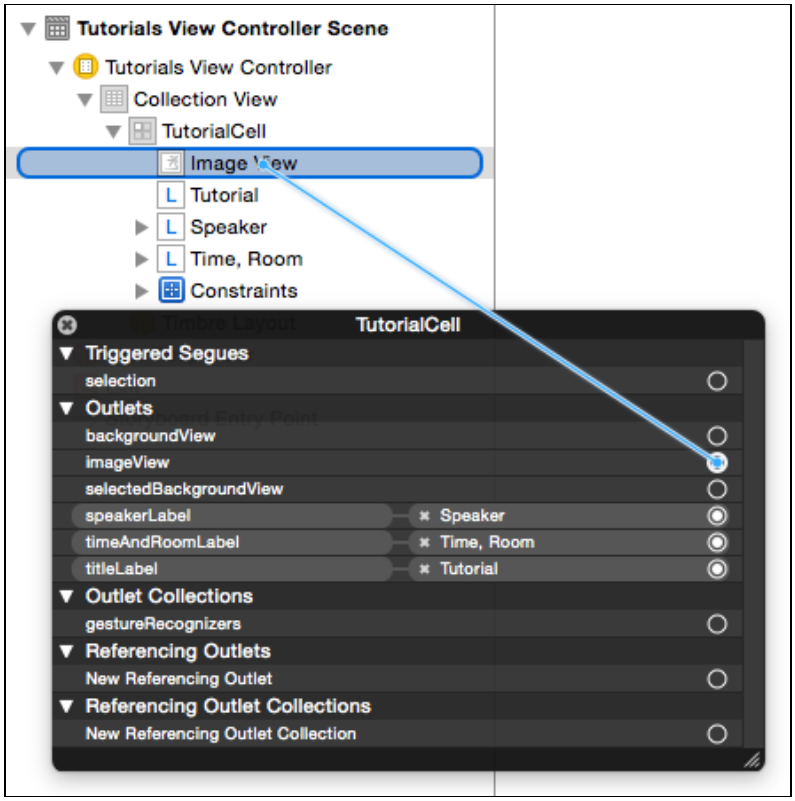
```
@IBOutlet private weak var imageView: UIImageView!
```

And now add the following to the bottom of the `if` block of the `didSet` observer on `tutorial`:

```
imageView.image = tutorial.backgroundImage
```

This simply updates the image being displayed by the image view whenever tutorial is set.

Jump back to **Main.storyboard** and right-click on **TutorialCell** in the Document Outline to invoke the Outlets and Connection popup. Drag from `imageView` to **Image View** to connect the two:



Now the image view will display the corresponding background photo, it's time to get rid of the background colors.

Open **TutorialsViewController.swift** and delete the following constant declaration:

```
let palette = UIColor.palette()
```

And then from `collectionView(_:cellForItemAtIndexPath:)` delete the following line:

```
cell.contentView.backgroundColor = palette[indexPath.item]
```

By removing these two lines, the background color is no longer set on each cell, which is now redundant since you're displaying background photos instead.

Build and run. You'll see the corresponding background photos are now displayed:

However, they are rotated by the same amount as the cell – you'll fix that now!

Fixing the rotation

Open **TutorialCell.swift** and add the following to the bottom of the class:

```
override func applyLayoutAttributes(layoutAttributes:
  UICollectionViewLayoutAttributes!) {
  // 1
  super.applyLayoutAttributes(layoutAttributes)
  // 2
  imageView.transform =
    CGAffineTransformMakeRotation(degreesToRadians(14))
```

```
    }
```

Here's the play-by-play of what's happening:

1. You call through to the `super` implementation to make sure the layout attributes are applied to the cell as normal.
2. Then you apply a transform to the image view that's the opposite to the one applied to the cell, which results in the image view being rotated back to it's usual position. You once again make use of the `degreesToRadians(_:)` utility function you added to the layout class.

Build and run. You'll see that the photos are now rotated the opposite way to the cell so that they appear straight:

Congratulations! You have added an image view to the cell which displays the tutorial's background photo, and you've countered the rotation applied to the cell so that they appear straight.

You may have noticed that the top-left and bottom-right edges of the photos appear to have been cut-off, but rest assured they haven't – it's a side-effect of rotating the image view in the opposite direction to the cell, and you'll fix it in the next video using container views and view clipping.