

Collection View Custom Layouts

Challenge Instructions

Copyright © 2014 Razeware LLC.

All rights reserved. No part of this book or corresponding materials (such as text, images, or source code) may be reproduced or distributed by any means without prior written per- mission of the copyright owner.

This book and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and non-infringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

All trademarks and registered trademarks appearing in this book are the property of their respective owners.

## Challenge: On The Right Track!

You may have noticed that the calendar still lacks one important piece of information you’d need for a conference schedule, the Track. You’re going to add this now, drawing on everything you’ve just learnt.

Add the following to layoutAttributesForElementsInRect(\_:) in **ScheduleLayout.swift** to get started, just above where you declared hourHeaderViewIndexPaths:

let trackHeaderViewIndexPaths =

dataSource.indexPathsOfTrackHeaderViews()

for indexPath in trackHeaderViewIndexPaths {

let trackHeaderViewAttributes =

layoutAttributesForSupplementaryViewOfKind("TrackHeader",

atIndexPath: indexPath as NSIndexPath)

attributes.addObject(trackHeaderViewAttributes)

}

Just like before, this asks the data source for an array containing the index paths for the track header views, and then iterates over that array, creating a set of layout attributes for each track header by calling layoutAttributesForSupplementaryViewOfKind(\_:atIndexPath:). The attributes returned are then added to the attributes array, which if you remember, contains the attributes for all the elements you need to lay out. This time around the *kind* you use is TrackHeader.

Next add the following convenience method:

private func frameForTrackHeaderViewAtIndexPath(indexPath: NSIndexPath) -> CGRect {

let heightPerTrack = (collectionViewContentSize().height –

dataSource.hourHeaderHeight) /

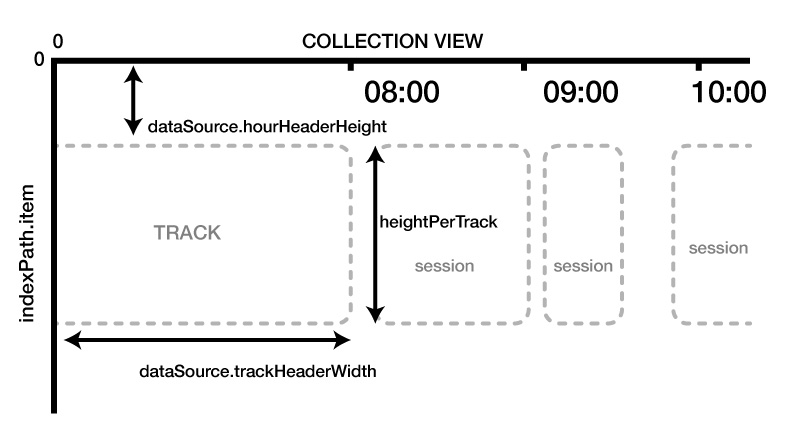
CGFloat(dataSource.numberOfTracksInSchedule)

let frame = // TODO...

return frame

}

Like frameForHourHeaderViewAtIndexPath(\_:) this should work out where the track header for the given index path should be positioned. Use the diagram below to help you figure out how you calculate the frame, and then replace the // TODO with the necessary code:



Find the following line in layoutAttributesForSupplementaryViewOfKind(\_:atIndexPath:)

attributes.frame = frameForHourHeaderViewAtIndexPath(indexPath)

And replace it with this:

attributes.frame = elementKind == "HourHeader" ?

frameForHourHeaderViewAtIndexPath(indexPath) :

frameForTrackHeaderViewAtIndexPath(indexPath)

This makes sure the proper frame-calculating convenience method is called based on the *kind* of the current supplementary view.

1. There are three remaining changes you need to make to **ScheduleLayout.swift** in order for the track headers to be displayed properly. Since the track headers will be displayed down the left-hand side of the collection view, their width needs to be taken into account when laying out both the cells and the hour header views. Therefore, you need to update the following:
   1. The width variable in collectionViewContentSize()
   2. The x value of the frame variable in frameForHourHeaderViewAtIndexPath(\_:)
   3. The x variable in frameForSession(\_:atIndexPath:)

The width of a track header view has been predetermined for you. Take a look at **ScheduleDataSource.swift** to find out which property to use.

Once you’ve made the above changes, open **ScheduleViewController.swift** and register the headerNib with the collection view a second time, but this time passing TrackHeader as the kind.

Finally, remove the contents of the header configuration block, replacing them with the following:

if kind == "HourHeader" {

header.titleLabel.text =

dataSource.titleForHourHeaderViewAtIndexPath(indexPath)

} else if kind == "TrackHeader" {

header.titleLabel.text =

dataSource.titleForTrackHeaderViewAtIndexPath(indexPath)

if let verticalSeparatorView = header.verticalSeparatorView {

verticalSeparatorView.hidden = true

}

}

This sets the title of the track header view using a convenience method on the data source, and since you’re reusing the same nib as the hour header, you also hide the vertical separator.

Build and run. If you’ve got everything working you should see something like this:

