Custom Controls in iOS

Hands-On Challenges

Custom Controls in iOS

Catie & Jessy Catterwaul

Challenge #9: Control Reuse

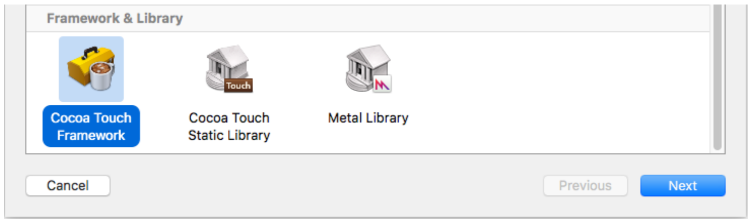
By Catie & Jessy Catterwaul

Create the Three Ring Control framework

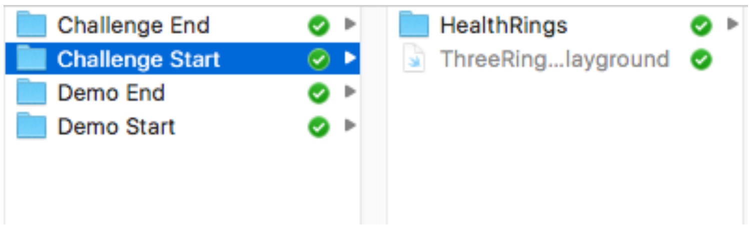
In the Challenge Start folder, you will have the ThreeRingControl.playground file you are familiar with, as well as a demo app we've set up for you to show off the control. It's your job to create a new framework for the Three Ring Control, import it into the app, and make any necessary adjustments to the framework!

Start opening the HealthRings Xcode project. Take a look at ViewController.swift and note that we've set up outlets and actions for the ThreeRingView here. Also note that it's error city in here. The compiler as no idea what a ThreeRingView is. Let's fix that, following the same process as in the demo.

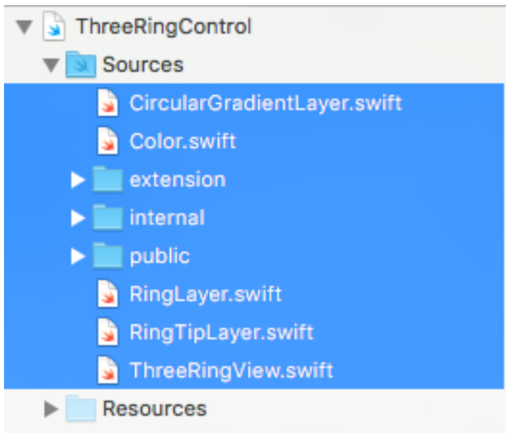
Go to File - New Project and select Cocoa Touch Framework.



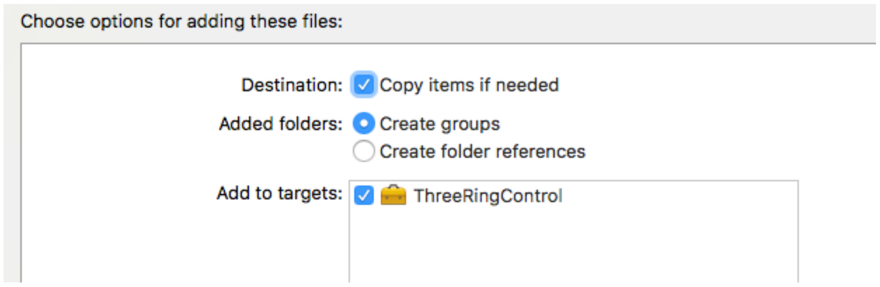
Name the framework ThreeRingControl and add it to the same parent folder that the HealthRings folder resides in.



Back in the finder window, open ThreeRingControl.playground. Open the sources folder, and drag all of the files and folders therein into your shiny new framework.



Make sure to check "Copy items if needed", and to choose "Create groups".



Choose options for adding these files:

Destination: ✓ Copy items if needed

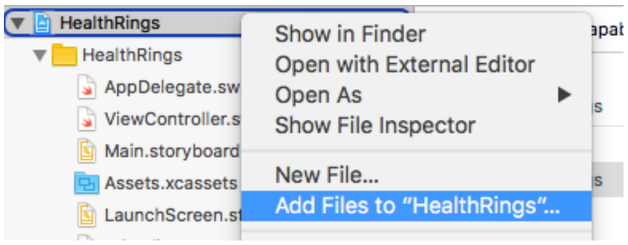
Added folders: ● Create groups
○ Create folder references

Add to targets: ✓ ThreeRingControl

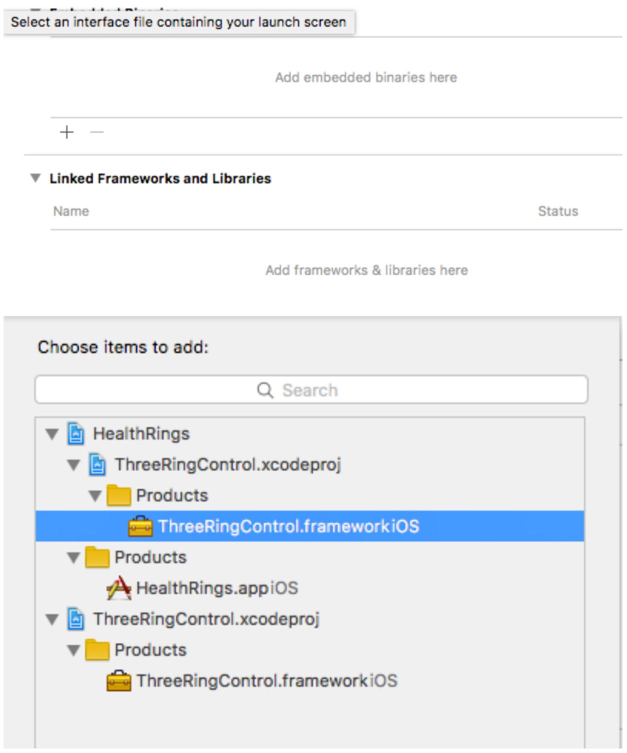
You will need all of these files, except one! Go ahead and delete PlaygroundPage.swift.

Add the framework to the demo app

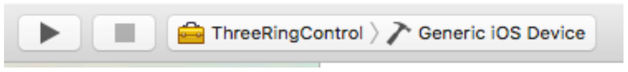
Close the playground and the framework, and head back to the HealthRings project. In the Project Navigator, right click on the HealthRings project, and select "Add Files to 'HealthRings'...". Navigate to the framework you just created, and select ThreeRingControl.xcodeproj.



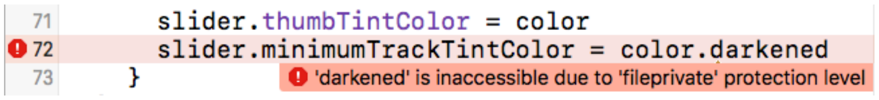
Select the HealthRings project in the Project Navigator, and under the General tab, scroll down to "Embedded Binaries". Click the little plus button, and in the popup, select ThreeRingControl.framework.


Select an interface file containing your launch screen
Add embedded binaries here
Linked Frameworks and Libraries
Name
Status
Add frameworks & libraries here
Choose items to add:
Search
HealthRings
ThreeRingControl.xcodeproj
Products
ThreeRingControl.framework iOS
Products
HealthRings.app iOS
ThreeRingControl.xcodeproj
Products
ThreeRingControl.framework iOS

If you check ViewController.swift, it still doesn't know what a ThreeRingControl is. Change the active scheme to ThreeRingControl and try building the framework itself.



Switch the active scheme back to HealthRings. Now the module is importing without trouble. Excellent! But we do have another error:



```
        slider.thumbTintColor = color
        slider.minimumTrackTintColor = color.darkened
    }
```

'darkened' is inaccessible due to 'fileprivate' protection level

We're trying to use the darkened version of each ring color on their associated value sliders in this app. It looks like now is a good time to look through the Three Ring Control framework, reorganize, and audit access levels.

Command + Click on `darkened` to get to its declaration in CIColor.swift. Aha! The UIColor extension is private. You'll need to mark is as `public` in order to use it elsewhere.

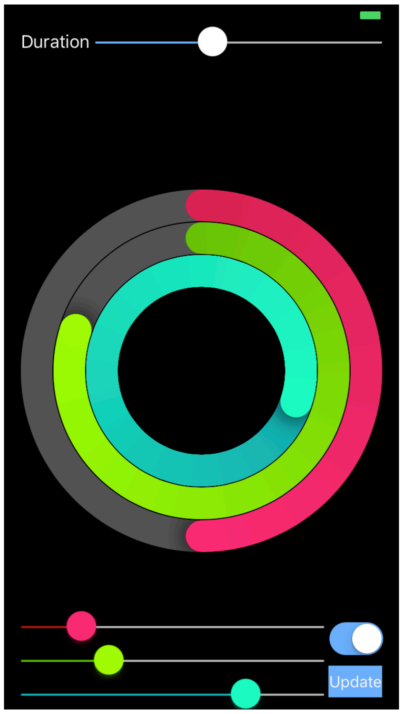
```
public extension UIColor {...
```

In fact, organizationally speaking, this extension should be moved into its own file if you plan to make it publicly accessible. Cut the UIColor extension out of CIColor.swift, create a new Swift file within the public folder called UIColor, and paste the extension in the new file. Don't forget to import UIKit at the top!

```
import UIKit

public extension UIColor {...
```

Try building the app again and you should find success! Go ahead and play around with the demo app and enjoy the very healthful, ring-shaped fruits of your labor.



Audit access levels

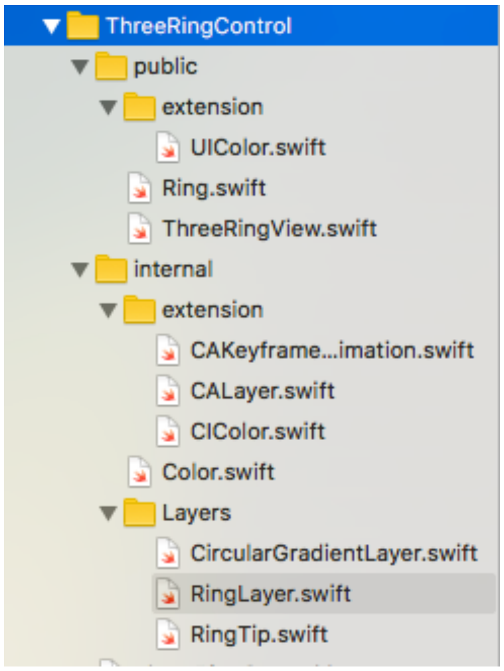
Back to work! Everything you need from ThreeRingControl is at least as accessible as required. But let's go back through and see if anything is *more* accessible than you need.

Currently Public:

- Ring
- UIColor
- ThreeRingView
- RingLayer
- CircularGradientLayer
- CALayer
- Color

Consider which of these are being used directly by this demo app. Only three of these classes and extensions, `UIColor`, `Ring`, and `ThreeRingView`, actually need accessible outside of the framework. Due to the way you built the control up in the playground, less restrictive access levels were sometimes required during development.

Reorganize all of the swift files into helpful groups like so:



Now start taking away access, one file at a time:

You aren't accessing the `RingLayer` class or its members outside of the framework. Delete every `public` in this file! There will be 8 altogether.

```
public final class RingLayer: CALayer {
  public var ringBackgroundColor...

  public var ringColor...

  public var ringWidth...

  public var value...

  public required init?(coder: NSCoder) {...

  public override init() {...

public extension RingLayer {...
```

The same applies to `CircularGradientLayer`! You'll find 5 `publics` there.

```
final class CircularGradientLayer: CALayer {
  var color = UIColor.white.cgColor {...
  required init?(coder: NSCoder) {...
  override init() {...
extension CircularGradientLayer {...
```

The `CALayer` extension doesn't need to be public either.

```
extension CALayer {...
```

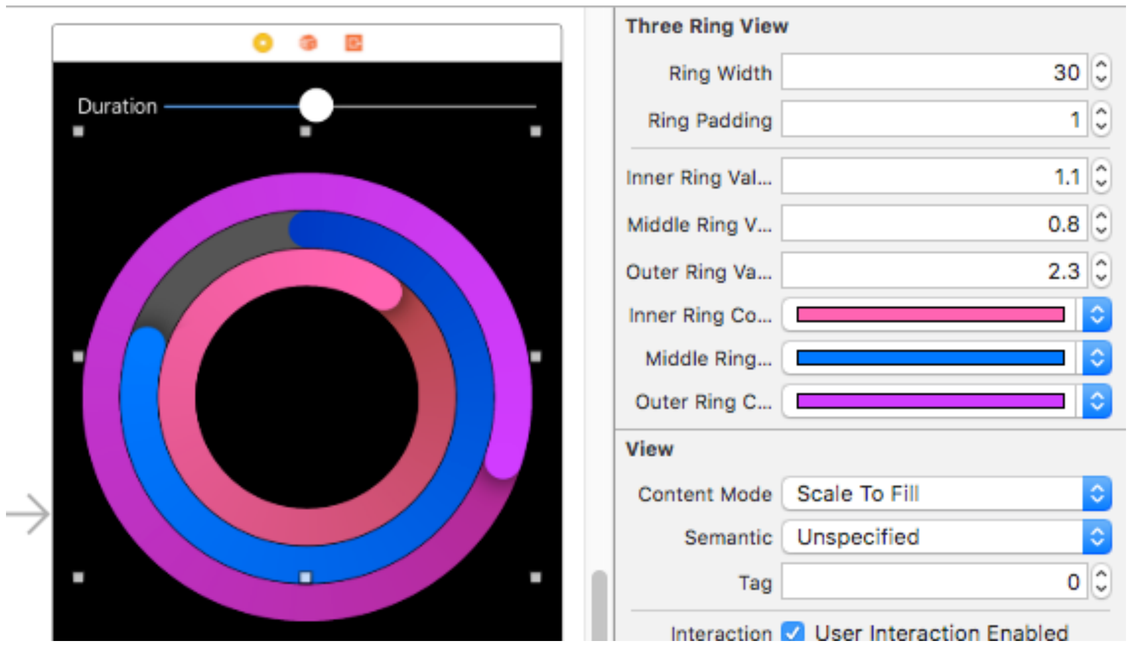
And finally, get rid of 2 `public` marks in the `Color` enum.

```
enum Color {}
extension Color {...
```

Build and run again, and you should see that nothing has changed! The app runs just as before.

Uber Haxxor Challenge

Your Three Ring Control is working perfectly in this demo app, but if you look at the storyboard you'll notice that the control isn't rendering there. You may recall from the Deluxe Button project how to change this! Your Uber Haxxor Challenge is to get the Three Ring Control to display in the storyboard, and add the ability to edit properties on the API directly from Interface Builder!



Three Ring View

Ring Width	30
Ring Padding	1
Inner Ring Val...	1.1
Middle Ring V...	0.8
Outer Ring Va...	2.3
Inner Ring Co...	🟥
Middle Ring...	🟦
Outer Ring C...	🟪

View

Content Mode	Scale To Fill
Semantic	Unspecified
Tag	0
Interaction	✓ User Interaction Enabled