

Saving Data in iOS

Hands-On Challenges

Saving Data in iOS Hands-On Challenges

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 permission 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 noninfringement. 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 A: Laying the Foundation

Throughout this series, you will be doing a lot of saving. You will not only saving lots of NSDatas and an NSStrings, you will also be reading a lot of them as well. This challenge is to get you comfortable with the process of using both of them so that will have no problems using them in future challenges.

Challenge 1: Saving User Data

The app provided in the challenge is a very simple app. The user can write some custom text in a text view, then they can select a photo from their photo library. All the code to do this has already been written.

The user also has the ability to choose where they would like to save their data. They can save it in the their Document folder, their Library folder, and also, in their Temporary Directory.

Your challenge is to provide this saving logic inside of `saveToDisk:`. For the sake of simplicity create one text file (`text.txt`) and one data file (`image.png`).

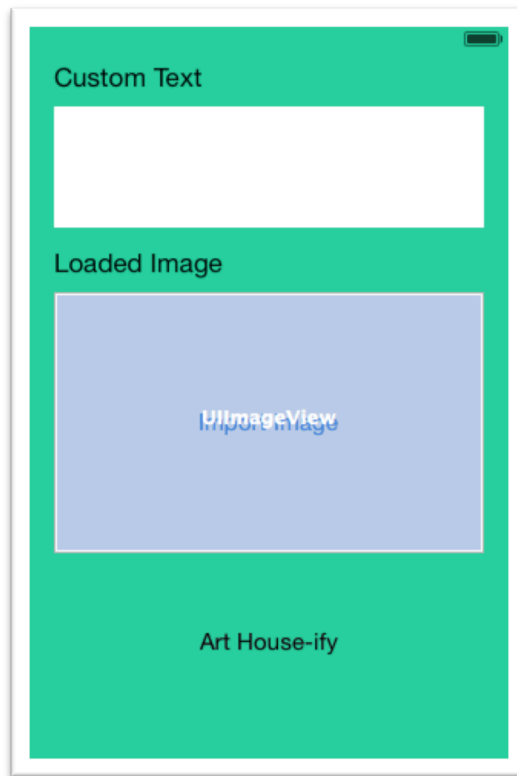
For retrieving the files, look in each directory and if you find the files, then populate the actual user interface with their values.

Uber Haxx0r Challenge: Bring a Little Art to your Life.

Your going to make some alterations to the sample app to save some “artistic alterations”. First, open **Main.Storyboard** and removed the segmented control that allows users to select their saving location.

Next add a UIButton in its place and call the “Art House-ify”.





Two things will happen when the user presses Art-House-ify. First the image will turn black and white, and second, the text will reverse. Once the user presses the button again, the text will return to normal and the picture will return in color.

Once the app makes a change, you should save the change in the NSCachesDirectory, that way, if the user presses the button again, you can fetch the files from the Cache directory and present them to the screen without having to do any additional processing.

To get started, open **ViewController.m**, and underneath the imports, make sure to add the following:

```
@import CoreImage;
```

Next, in your code, you're going to add the following Core Image code to turn your images black and white:

```
CIImage * tempImage = [CIImage  
imageWithCGImage:self.imageView.image.CGImage];  
  
CIImage * blackAndWhite = [CIFilter  
filterWithName:@"CIColorControls" keysAndValues:kCIInputImageKey,  
tempImage, @"inputBrightness", [NSNumber numberWithInt:0.0],
```



```

@"inputContrast", [NSNumber numberWithFloat:1.1],
@"inputSaturation", [NSNumber numberWithFloat:0.0],
nil].outputImage;

CIImage * output = [CIFilter filterWithName:@"CIExposureAdjust"
keysAndValues:kCIInputImageKey, blackAndWhite, @"inputEV",
[NSNumber numberWithFloat:0.7], nil].outputImage;

CIContext * context = [CIContext contextWithOptions:nil];
CGImageRef cgiimage = [context createCGImage:output
fromRect:output.extent];

UIImage * newImage = [UIImage imageWithCGImage:cgiimage];

CGImageRelease(cgiimage);

```

Your challenge is to re-write the `saveToDisk:`. If the app is art-ifying the image and text, it will first look in the `NSCacheDirectory` to see if those elements already exist. If they do not exist, the app create them and save them into that directory.

At this point, the button name should change to "Normal-fy" at which point it should display the original contents.

When the app is loaded, the original contents should display as well.

