Saving Data in iOS

Hands-On Challenges

Saving Data in iOS Hands-On Challenges

Challenge A: Unzipping Files

There may be times when you need to download files from a remote source such as additional levels or maybe even new artwork. In a lot of cases, you'll want to compress these files to save time when sending across the network. Of course, once you receive the files, you'll have to uncompress them, then organize them. This is the basis of the challenge.

Challenge: Unzipping and moving

The starter has a zip file called bundle.zip. Your job is to unzip this file into the temp directory, then move the files into the document directory.

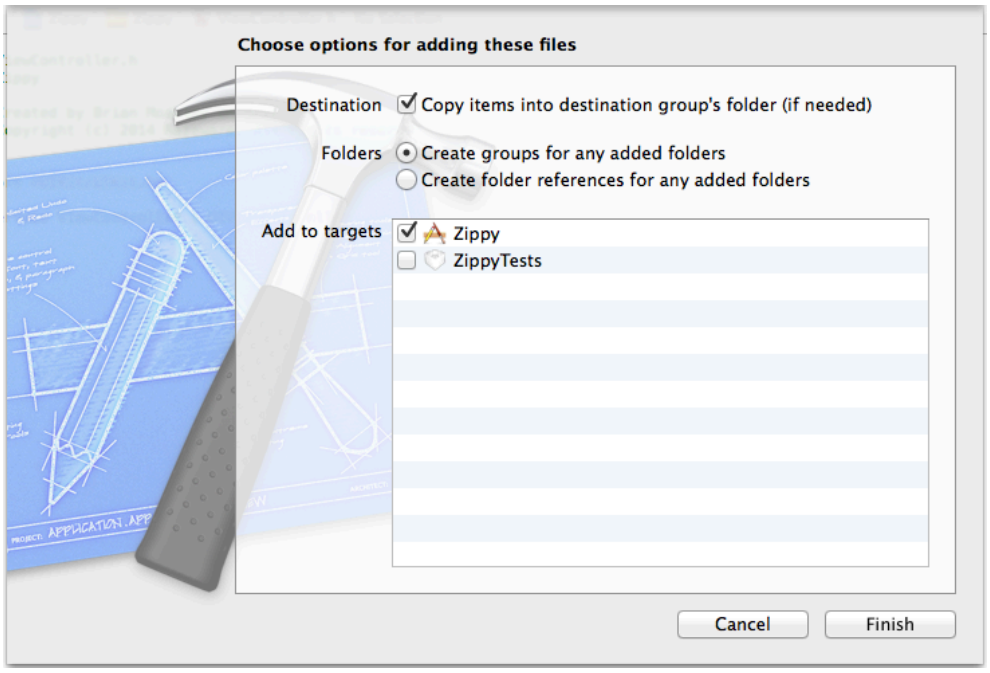
Thankfully, you won't have to write any of the unzipping code. You'll have to get a library to do that for you. Head over to GitHub to download **SSZipArchive**. You can find the link here:

https://github.com/soffes/ssziparchive

Extract the files, then open the starter project.

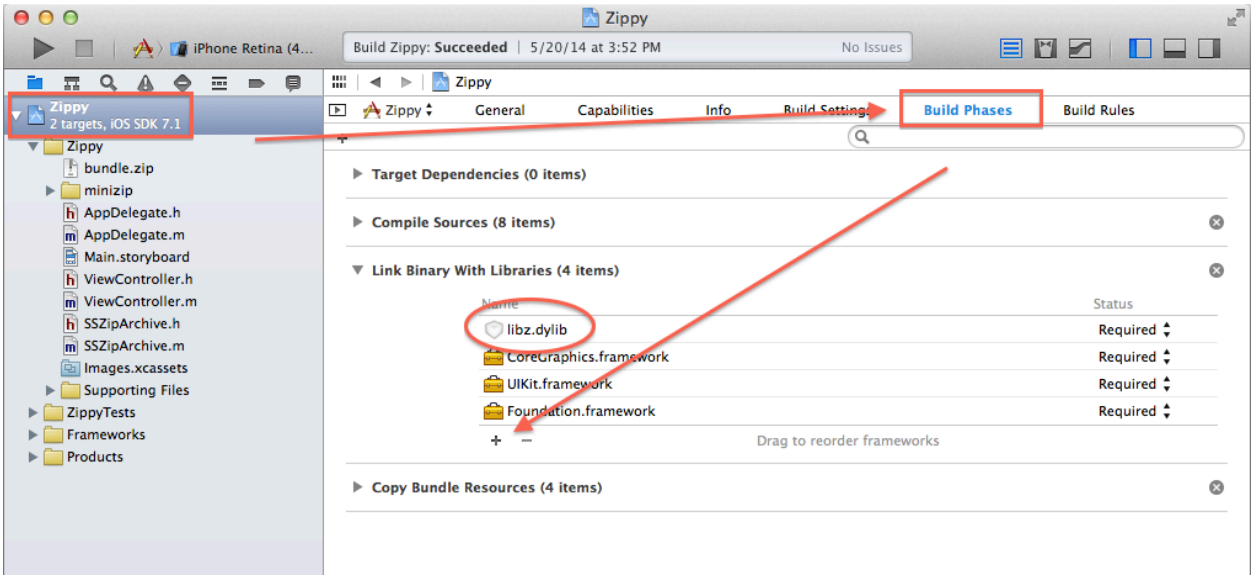
First, add SSZipArchive folder to the starter project by dragging it into the challenge project.

Make sure the "Copy Items in destination project's folder" is checked. You should see the following dialog:



Next you have to link the appropriate library.

Click on the **project**, then select **Build Phases**. Expand the **Link Binary With Libraries** section, then click the **+** button. Add the **libz.dylib**.



Build your project and you should now have the ability to unzip your files.

In your AppDelegate.m, you'll need get a path to the actual zip file inside of your app bundle. You can get the path to the zip file from the following code.

```
[[NSBundle mainBundle] pathForResource:@"bundle" ofType:@"zip"];
```

This is a string path, so you will have to convert it to a url. Once you have it, you will next have to unzip the file into a directory. You will need to provide the source of the zip file and location where you wish to unzip the files.

```
[SSZipArchive unzipFileAtPath:zipfile toDestination:directory];
```

Once you have unzipped the files, you will need to use the file manager to move them to Document directory.

Uber Haxx0r Challenge: The Sorting Code

Moving files is certainly useful, but it also helps to keep them organized. In this challenge, you will unzip the files into the temporary directory. Then, you should

create two additional directories. These directories should be called **png** and **jpg**, and they should be placed inside of the document directory.

Loop though the files and move the png files to the png folder and the jpg files to the jpg folder. If you run into any txt files, then those should be move to the Library directory.

Good luck!

Two things will happen will the user press 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 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 numberWithFloat: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.