BEGINNING

CORE DATA

Hands-On Challenges

Beginning Core Data

Luke Parham

Notice of Rights

Notice of Liability

Trademarks

Challenge #2: Adding an Image

By Luke Parham

As you saw in the demo, the app allows you to add quite a few attributes to each Person you add to your list. The only thing you still can't do is add a picture via the UI.

Adding an Image Picker

In order to pick an image for each `Person`, you'll need to add a `UIImagePickerController`.

The first thing you need to do is add a property to keep track of which person you're picking an image for.

Add the following at the top of the **ViewController.swift** with the other property definitions.

```
var imagePickerIndexPath = IndexPath(item: 0, section: 0)
```

Next, make `ViewController` the `collectionView`'s delegate by adding the following line to `viewDidLoad()` after the line that makes it up as the dataSource.

```
collectionView.delegate = self
```

Now, add an extension where so that you can implement a collection view delegate method.

```
extension ViewController: UICollectionViewDelegate {

}
```

The only delegate method you need is for selecting a cell. Add the following method to the delegate extension.

```
func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {
    //1
    imagePickerIndexPath = indexPath

    //2
    let pickerController = UIImagePickerController()
    pickerController.delegate = self
    self.navigationController?.present(pickerController,
        animated: true,
        completion: nil)
}
```

1. When a user selects a cell, first you save off the chosen index path.
2. Then you display an image picker controller.

```
if let pictureData = person.picture {
  cell.pictureImageView.image = UIImage(data: pictureData as Data)
}
```

Implementing the Picker Delegate

Now that you're showing the image picker, you need to react to an image being chosen.

First, add the following extension conforming `ViewController` to the `UIImagePickerControllerDelegate` protocol.

```
extension ViewController: UIImagePickerControllerDelegate,
                          UINavigationControllerDelegate {

}
```

The only method you need to impelement is `imagePickerController(_:didFinishPickingMediaWithInfo:)`.

Add the following method to the extension

```
func imagePickerController(_ picker: UIImagePickerController,
    didFinishPickingMediaWithInfo info: [String : Any]) {
    guard let appDelegate =
      UIApplication.shared.delegate as? AppDelegate else {
      return
    }
}
```

As usual, you need to add a guard statement to make sure you can access the context in the app delegate.

Next, retrive the chosen image from the `info` dictionary and retrieve the chosen

`Person` from the `people` array using the cached index path.

```
let image = info[UIImagePickerControllerOriginalImage] as! UIImage
let person = people[imagePickerIndexPath.row]
```

As you've seen, all you need to do to save the image is set it do the picture property of the person.

```
person.picture = UIImagePNGRepresentation(image)! as NSData
```

Now that the image has been set on the managed object, you can save the context to persist it.

```
do {
  try appDelegate.persistentContainer.viewContext.save()
} catch let error {
  print(error)
}
```

Finally, reload the altered person's cell and dismiss the picker controller.

```
collectionView.reloadItems(at: [imagePickerIndexPath])
picker.dismiss(animated: true, completion: nil)
```

Showing The Image

If you built and ran right now, you would technically be able to save images, but you haven't actually added any code to show the images in the cells.

Fixing this problem is a snap. Just go to `collectionView(_:cellForItemAt:)` located in the `UICollectionViewDataSource` extension.

Before the return statement, add

```
if let pictureData = person.picture {
  cell.pictureImageView.image = UIImage(data: pictureData as Data)
}
```

Now, build and run to see that any `Person` object that has a saved image will display that image in the corresponding cell.