iOS DESIGN PATTERNS

HANDS-ON CHALLENGES

iOS Design Patterns

Joshua Greene

Copyright ©2017 Razeware LLC.

Notice of Rights

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.

Notice of Liability

This challenge and all corresponding materials (such as source code) are provided on an "as is" basis, without warranty of any kind, express of 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 of other dealing in the software.

Trademarks

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

Table of Contents: Overview

Table of Contents: Extended

4

MVVM - Challenge

By Joshua Greene

`ProductDetailsViewController` is looking slim!

But what about `ProductsViewController`? Can ProductViewModel help there too? You bet!

Challenge

First, update **ProductsViewController** to use a **ProductViewModel** array instead of a **Product** array.

This isn't very useful in itself-- it's basically the same, and there's still duplicate code between `ProductDetailsViewController` and `ProductsViewController`.

You can eliminate this duplication by creating a **ProductViewModelView protocol** at the top of **ProductViewModel**. This will represent a "view" that's configurable by a `ProductViewModel`.

Make both **ProductCollectionViewCell** and **ProductDetailsViewController** conform to this protocol.

Add a new method with signature `func configure(_ view: ProductViewModelView)` in an extension on **ProductViewModel**.

Finally, update **ProductDetailsViewController** and **ProductsViewController** to use this method instead.

Challenge Solution

Open **ProductsViewController.swift** and replace the `var products` declaration with this:

```
internal var productViewModels: [ProductViewModel] = []
```

This will result in several compiler errors, which you'll fix next.

Replace **strongSelf.products = products** with this:

```
strongSelf.productViewModels =
  products.map { ProductViewModel(product: $0) }
```

Replace **let product = products[indexPath.row]** with this:

```
let productViewModel = productViewModels[indexPath.row]
```

Replace **prepare(for:, sender:)** with this:

```
public override func prepare(for segue: UIStoryboardSegue, sender: Any?)
{
  guard let viewController = segue.destination
    as? ProductDetailsViewController else { return }
  let indexPath = collectionView.indexPathsForSelectedItems!.first!
  viewController.productViewModel = productViewModels[indexPath.row]
}
```

Replace **return products.count** with

```
return productViewModels.count
```

Finally, replace **collectionView(_:, cellForItemAt:)** with this:

```
public func collectionView(
  _ collectionView: UICollectionView,
  cellForItemAt indexPath: IndexPath)
  -> UICollectionViewCell {

  let cellIdentifier = "ProductCell"
  let productViewModel = productViewModels[indexPath.row]

  let cell = collectionView.dequeueReusableCell(
    withReuseIdentifier: cellIdentifier,
    for: indexPath) as! ProductCollectionViewCell

  cell.label.text = productViewModel.titleText
  cell.imageView.rw_setImage(url: productViewModel.imageURL)

  return cell
}
```

Build and run and verify everything works as expected.

Next, add the following to the top of **ProductViewModel.swift**:

```
@objc public protocol ProductViewModelView {****
  @objc optional var productImageView: UIImageView { get }
  @objc optional var productPriceLabel: UILabel { get }
  @objc optional var productDescriptionLabel: UILabel { get }
  @objc optional var productTitleLabel: UILabel { get }
}
```

And the following to the bottom of **ProductViewModel.swift**:

```
extension ProductViewModel {

  public func configure(_ view: ProductViewModelView) {
    _ = view.productImageView?.rw_setImage(url: imageURL)
    view.productPriceLabel?.text = priceText
    view.productDescriptionLabel?.text = descriptionText
    view.productTitleLabel?.text = titleText
  }
}
```

Add the following to the bottom of **ProductDetailsViewController.swift**:

```
// MARK: - ProductViewModelView
extension ProductDetailsViewController: ProductViewModelView {

  public var productImageView: UIImageView {
    return imageView
  }

  public var productPriceLabel: UILabel {
    return priceLabel
  }

  public var productDescriptionLabel: UILabel {
    return descriptionLabel
  }
}
```

Add the following to the bottom of **ProductCollectionViewCell.swift**:

```
// MARK: - ProductViewModelView
extension ProductCollectionViewCell: ProductViewModelView {

  public var productImageView: UIImageView {
    return imageView
  }

  public var productTitleLabel: UILabel {
    return label
  }
}
```

With all this in place, you can finally eliminate the code duplication in **ProductDetailsViewController** and **ProductsViewController**.

Replace **collectionView(_:, cellForItemAt:)** on **ProductsViewController** with this:

```
public func collectionView(
  _ collectionView: UICollectionView,
  cellForItemAt indexPath: IndexPath)
  -> UICollectionViewCell {

  let cellIdentifier = "ProductCell"
  let productViewModel = productViewModels[indexPath.row]

  let cell = collectionView.dequeueReusableCell(
    withReuseIdentifier: cellIdentifier,
    for: indexPath) as! ProductCollectionViewCell

  productViewModel.configure(cell)

  return cell
}
```

Lastly, replace **viewDidLoad()** on **ProductDetailsViewController** with this:

```
public override func viewDidLoad() {
  super.viewDidLoad()
  productViewModel.configure(self)
}
```

Build and run and verify everything still works as expected.

Über challenge

Whoa, this was already hard enough! Take a breather, you've earned it. ;]