INTERMEDIATE

CORE DATA

HANDS-ON CHALLENGES

Intermediate Core Data

Luke Parham

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.

Challenge #5: Alternative Means

By Luke Parham

In the demo you saw what could happen when multiple contexts started trying to operate on the data in the persistent store at once.

Your challenge this time is to try an alternative method of creating a new context for the edit screen.

Getting a New Context

In the demo, you saw how you could create a new context and set up a parent child relationship so that saving the child would move the changes into the parent.

Alternatively, you could just have the app's persistent container generate a new background context for you.

Go to `prepare(for:sender:)` and replace the `if let` binding with

```
if let cell = sender as? TargetCollectionViewCell,
   let indexPath = collectionView.indexPath(for: cell) {

  let destination = segue.destination as!
      PersonDetailTableViewController

  destination.personID = people[indexPath.row].objectID
  destination.managedObjectContext =
      appDelegate.persistentContainer.newBackgroundContext()
}
```

Now, instead of doing all the work yourself, you're letting the container generate a new context for you. This context has a `.privateQueueConcurrencyType` concurrency type just like yours did and has its parent set to be the persistnet store coordinator.

Since you aren't allowed to change the parent of this background context, you'll need to find another way to synchronize your contexts.

Go to `reloadData()` and right below the line where you set the query generation,

add the following line.

```
  appDelegate.persistentContainer.viewContext.refreshAllObjects()
```

This will allow your view context to refresh any objects that are altered by the new background context when its used on the edit screen!