Intermediate Realm on iOS

Hands-On Challenges

Intermediate Realm on iOS

Marin Todorov

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

3 Challenges: Multiple Realms

By Marin Todorov

Challenge A: Add new exam when the user selects a subject

At this point you have the subject list show up on the screen. But currently when the user selects a subject from the list nothing happens. You will fix that in this challenge.

Open up **SubjectsViewController.swift** and scroll to `didSelectRowAtIndexPath:`. In this method you have the opportunity to grab the relevant `SubjectName` object and use its date to create a new `Exam` object in your main Realm file.

Insert inside the empty method body:

```
guard let subject = subjects?[indexPath.row] else {
  return
}

let realm = try! Realm(configuration: RealmConfig.main.configuration)
let exam = Exam(subject.name, date: Date(timeIntervalSinceNow: 7 * 24 *
60 * 60))
```

You fetch the currently selected subject from `subjects` and use its name property to create a brand new `Exam` object. In the end you set the exam's date to next week (in a real life project you will provide a date picker or other UI for the user to select the correct date but we won't focus too much on UI code right now).

Next you need to write the new object to your main Realm file and you're done! Add:

```
try! realm.write {
  realm.add(exam)
}
```

Notice how you read the data from an object from your static Realm and wrote it back to your main Realm file. Cross-Realm operations for the win!

Finally pop the subjects list view controller out of your navigation stack so that the user can glance over their current selection of outstanding exams:

```
navigationController!.popViewControllerAnimated(true)
```

Challenge B: Filter the subject list to only unused topics

Right now if you create a new exam on subject of Biology (for example) when you want to add another exam – you will see that Biology is still in the list of available subjects.

In the previous challenge you learned how to work with data from both your static and main Realm files so you can pretty easy add a bit of code to filter the list of available subjects.

Back in **SubjectsViewController.swift** scroll to `viewDidLoad`.

First grab the names of all the current exams – insert this code at the top of `viewDidLoad`:

```
let mainRealm = try! Realm(configuration:
  RealmConfig.main.configuration)
let examNames = Array(mainRealm.objects(Exam.self)
  .map { $0.name })
```

Here you fetch all existing `Exam` objects and store their names in `examNames`.

Now you need to filter the list of `SubjectName` objects to the ones that have a name not found in `examNames`. Adjust the line where you query `SubjectName` like so:

```
subjects = staticRealm.objects(SubjectName.self)
  .filter("NOT name IN %@", examNames)
  .sorted(byKeyPath: "name")
```

Thanks to the powerful `NSPredicate` syntax you simply say you want the name property not in a list of values and you're off to the races.

Test the app again and you will notice the subjects list doesn't include the items you've already used to add exams.