

BEGINNING

# CORE DATA



HANDS-ON CHALLENGES

## Beginning 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 or 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 #3: Fixing the Sort

By Luke Parham

As you might have noticed, the names are sorted, but it seems lowercase and uppercase names are weirdly split into two sections.

As you can see, the lowercase "michael" is put at the end. How strange...

## Changing the Sort Method

It turns out that by default, `NSSortDescriptor` uses the `compare(_:)` method to compare the values of the attribute you're sorting by. In this case, the result is what you see.

What you actually want to see is people ordered by the first letter of their first name, regardless of the casing.

To accomplish this, go to `reloadData(_:)` and replace

```
let sort = NSSortDescriptor(  
    key: #keyPath(Person.name), ascending: true)
```

with

```
let sort = NSSortDescriptor(  
    key: #keyPath(Person.name),  
    ascending: true,  
    selector: #selector(  
        NSString.caseInsensitiveCompare(_:))
```

Build and Run to see people sorted the way nature intended.

## Where to Go From Here

There are a ton of options when it comes to sorting strings. For instance, for your app, even `caseInsensitiveCompare(_:)` might not be specific enough. If you need

to worry about localized strings or numbers being involved in the string you could use `localizedStandardCompare(_:)` or even something very specific via the `compare(options:range:locale:)` method.

To learn more about String sorting, check out the [String Programming Guide](#).