Errata for Creating iOS 5 Apps: Develop and Design / 0-321-76960-0

For a complete and up-to-date list of errata, please see the author's companion page at:

http://www.freelancemadscience.com/creating-ios-apps-faq/errata/

ERRATA

Page 37: viewWillLoad

On page 37, viewWillLoad should be changed to viewWillAppear (matching the sample code below it).

Page 51: {'A', '\o'} Typo

In the note on page 51, "A" should be equal to the array {'A', '\0'} not {'A', '/o'} .

Page 77, 296, 420: Unnecessary retains

There are unnecessary retain calls on page 77, 296 and 420.

The sample code at the top of the page 77 calls `retain`. This is not needed when using ARC. In fact, as written, the code will not compile.

```
if (self) {

    _firstName = [firstName retain];

    _lastName = [lastName retain];

    _id = id;

}
```

should be

```
if (self) {

    _firstName = firstName;

    _lastName = lastName;

    _id = id;

}
```

Page 156: misplaced semicolon

At the bottom of page 156, there's a semicolon in the header of removeWeightAtIndex: when there shouldn't be.

The following code:

```
- (void)removeWeightAtIndex:(NSUInteger)weightIndex;{
```

should be

```
- (void)removeWeightAtIndex:(NSUInteger)weightIndex{
```

Page 163: Typo "viewDidApepar:"

On page 163, "viewDidApepar:" should be "viewDidAppear"

On pages 224, 237 and 431, I use CGFLOAT_MIN to set an initial value for the maxWeight variable. The idea was to assign a value that would be guaranteed to be lower than or equal to all the weight values in the weight history. However, CGFLOAT_MIN is not actually the minimum float (that would be -CGFLOAT_MAX). It's the float value closest to zero (the smallest possible fractional value that a float can express).

The code should, therefore, use -CGFLOAT_MAX (or more simply 0.0f, since all weights are positive values) instead.

```
    CGFloat minWeight = CGFLOAT_MAX;

    CGFloat maxWeight = -CGFLOAT_MAX;

    int monthlyCount = 0;

    CGFloat monthlyTotal = 0.0f;

    for (WeightEntry* entry in self.weightHistory) {
```

Page 296: Missing WeightInLbsKey

Page 296, all the references to WeightInLbsKey should be WeightKey instead.

```
_date = [[decoder decodeObjectForKey:DateKey] retain];
```

should be

```
_date = [decoder decodeObjectForKey:DateKey];
```

Pages 325, 331 and 333 all references to <code>accessHandler</code> should be <code>completionHandler</code> instead.

Pages 329, 350, 353, 354, 406, 433, 435, 498: Removing addObserverForName: observers

Throughout the book (pgs 329, 350, 353, 354, 406, 433, 435, and 498), I use addObserverForName:object:queue:usingBlock: to observe notifications. Typically I then try to remove the observer using code like the following:

```
- (void)viewDidUnload

{
```

```
    [super viewDidUnload];

    // Release any retained subviews of the main view.

    // e.g. self.myOutlet = nil;

    [[NSNotificationCenter defaultCenter]

     removeObserver:self];

}
```

This simply won't work. The view controller (self) isn't the observer. Removing it doesn't do anything at all.

Instead, whenever you call addObserverForName: you need to catch the return value. You then use this value when you want to remove the observer. It's probably easiest to do this using a property.

```
@property(strong, nonatomic) id observer;
```

Then create the observer as shown:

```
self.observer =

[[NSNotificationCenter defaultCenter]

     addObserverForName:NSUserDefaultsDidChangeNotification

     object:[NSUserDefaults standardUserDefaults]

     queue:nil

     usingBlock:^(NSNotification *note) {


         [graphView setWeightEntries:self.weightHistory

                             andUnits:getDefaultUnits()];

     }];
```

And then remove the observer:

```
- (void)viewDidUnload

{

    [super viewDidUnload];

    // Release any retained subviews of the main view.
```

```
    // e.g. self.myOutlet = nil;

    [[NSNotificationCenter defaultCenter]

     removeObserver:self.observer];

}
```

Pages 360, 361: Unnecessary Releases

When compiled under ARC, we no longer need to call release on our objects. In fact, the code will not compile. There are two instances on pages 360 and 361. In both cases, the code should be changed as shown below:

```
    [alert show];
    [alert release];
```

Should be

```
    [alert show];
```

Page 332: Typo

Page 332, @"file, found %d ' , // single quote after %d should be @"file, found %d " , // double-quote

Page 360: No managedObjectContext property

On page 360, there are two references to `self.managedObjectContext.undoManager`. The WeightHistory class does not have a managedObjectContext property. These should both be `self.undoManager` instead.

```
if ([self.managedObjectContext.undoManager canUndo]) {

...

        NSString* message =

        [self.managedObjectContext.undoManager undoActionName];
```

Should be...

```
if ([self.undoManager canUndo]) {

...

        NSString* message =

        [self.undoManager undoActionName];
```

Page 362: didReceiveMemoryWarning implementation

On page 362, we are supposed to override the didRecieveMemoryWarning method to remove all undo actions; however, the WeightHistory does not inherit this method. It's probably best to move this to the TabBarController class instead. The correct implementation (in TabBarController.m) is shown below:

```
- (void)didReceiveMemoryWarning

{

    // Releases the view if it doesn't have a superview.

    [super didReceiveMemoryWarning];

    [self.weightHistory.undoManager removeAllActions];

}
```

Page 387: wrong comparison for "at least 3 children"

In the predicate examples on page 387, I used less-than 3 instead of 3 or greater as shown below:

```
    // Determines if the target has at least 3 children.

    [NSPredicate predicateWithFormat:@"children[size] < 3"];

    Finally, we can combine simple comparisons using AND, OR, or NOT.

    // Determines if the target has at least 3 adult children.

    [NSPredicate predicateWithFormat:

        @"(children[size] < 3) AND (NONE children.age < %@)",

        [NSNumber numberWithInt:18]];
```

should be

```
    // Determines if the target has at least 3 children.

    [NSPredicate predicateWithFormat:@"children[size] > 2"];

    Finally, we can combine simple comparisons using AND, OR, or NOT.

    // Determines if the target has at least 3 adult children.

    [NSPredicate predicateWithFormat:

        @"(children[size] > 2) AND (NONE children.age < %@)",

        [NSNumber numberWithInt:18]];
```

On page 420, we define a property using a `retain`attribute. This doesn't cause any problems when compiling the app, but it probably should be changed to `strong`.

```
@property (nonatomic, retain) NSFetchedResultsController*

fetchedResultsController;
```

should be

```
@property (nonatomic, strong) NSFetchedResultsController*

fetchedResultsController;
```

Page 431: Deleting #import "WeightEntry.h"

Throughout chapter 7, I have you delete and then reinsert the line to #import "WeightEntry.h". This is a side effect of the way I created the code for the chapter, that I simply didn't notice when writing. Basically, I did this in two steps. I removed the old WeightEntry class first. Then I inserted the new WeightEntry class. Since they were the same name, we could have just left the #import lines alone.

In particular, on page 431 I have you remove #import "WeightEntry.h" from DetailViewController.m, but we never put it back. Obviously, DetailViewController.m needs to import the WeightEntry class for it to compile correctly. So, you can just skip the instructions on page 431 and leave that #import directive alone.