

# Table Views in iOS

Hands-On Challenges

# Table Views: Beginning to Advanced Hands-On Challenges

Copyright © 2014 Razeware LLC.

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.

This book 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 or other dealings in the software.

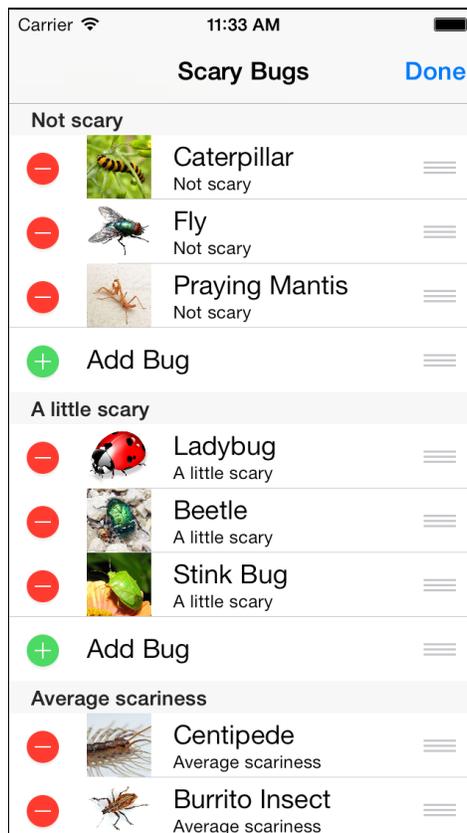
All trademarks and registered trademarks appearing in this book are the property of their respective owners.



# Challenge #5: Moving Rows

You can now add bugs and delete bugs, so there's only one thing left – moving them around!

This time your challenge is to modify your app to show reorder controls when the app is in edit mode. You should be able to move bugs between sections and have their scary bug factor update accordingly:



See if you can do this on your own based on what you learned on the video. If you get stuck, follow along with the full walkthrough below!

## Full Walkthrough

Open the Scary Bugs project where you left it off in the last challenge, or use the starter project provided by the instructor.

Remember from the lecture that there are two methods you need to implement to support moving rows:



1. tableView:canMoveRowAtIndexPath:
2. tableView:moveRowAtIndexPath:toIndexPath:

Let's start with the first. Add this new method to the bottom of the file:

```
- (BOOL)tableView:(UITableView *)tableView
canMoveRowAtIndexPath:(NSIndexPath *)indexPath {
    BugSection *bugSection = self.bugSections[indexPath.section];
    if (indexPath.row >= bugSection.bugs.count && [self isEditing]) {
        return NO;
    } else {
        return YES;
    }
}
```

This method returns YES for any row except for the "Add Bug" row that appears in editing mode. It wouldn't make sense to let the user move the "Add Bug" row!

Next, add the second required method:

```
- (void)tableView:(UITableView *)tableView
moveRowAtIndexPath:(NSIndexPath *)sourceIndexPath
toIndexPath:(NSIndexPath *)destinationIndexPath {

    // 1
    BugSection *sourceSection = [self.bugSections
        objectAtIndex:sourceIndexPath.section];
    ScaryBug *bugToMove = [sourceSection.bugs
        objectAtIndex:sourceIndexPath.row];
    BugSection *destSection = [self.bugSections
        objectAtIndex:destinationIndexPath.section];

    // 2
    if (sourceSection == destSection) {
        [destSection.bugs exchangeObjectAtIndex:destinationIndexPath.row
            withObjectAtIndex:sourceIndexPath.row];
    }

    // 3
    else {
        bugToMove.howScary = destSection.howScary;
        [destSection.bugs insertObject:bugToMove
            atIndex:destinationIndexPath.row];
        [sourceSection.bugs removeObjectAtIndex:sourceIndexPath.row];
    }

    // 4
}
```



```
}
```

Since you have implemented `tableView:canMoveRowAtIndexPath:`, the table view will display the reorder control for all rows, and implements the code to let the user drag and drop rows using the reorder control.

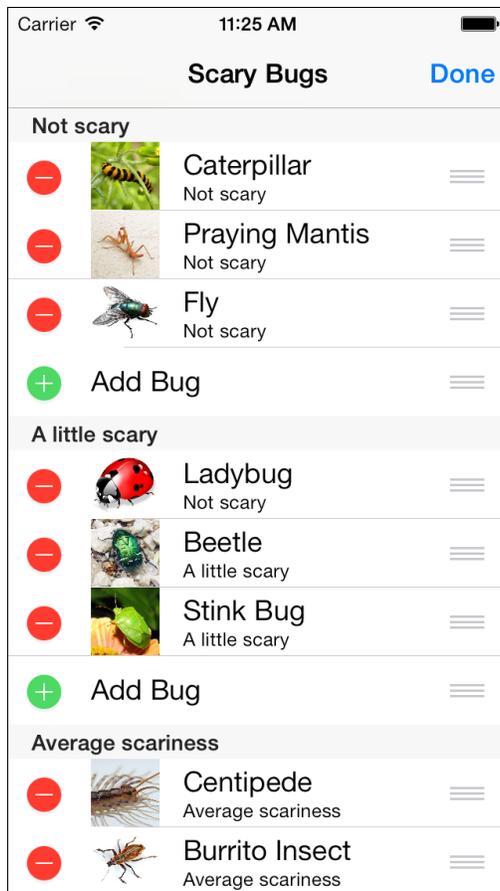
When the user drops a row into a new location, this method is called. The table view already knows about the move, so you only need to update your model appropriately in this method.

There's a lot of code here, so let's go over it section by section:

1. Determines the source `BugSection`, the destination `BugSection`, and the `ScaryBug` to move.
2. If the bug is being moved within the same section (i.e. within the same array), the best way to swap the bug position is with the `exchangeObjectAtIndex:withObjectAtIndex:` method on `NSMutableArray`.
3. If the bug is being moved to a new section, you need to insert the bug in the new section at the right spot, and remove it from the old section. You also update the `howScary` on the bug, because in this app each section represents a certain scariness level of bugs.
4. You'll be adding some extra code here later on.

Build and run, and tap the Edit button to make the reorder controls appear. You'll notice that switching bugs within sections works fine (see the Fly moved down), however if you drag a bug to a different section the cell doesn't refresh (see the Ladybug):





To fix the refreshing issue, you'd think that you could just call `reloadRowsAtIndexPaths:withRowAnimation:` at section #4, however that doesn't work. It causes some strange behavior with the table view, such a blank row and mismatched cells – likely due to the fact that when moving a cell, the row you are moving is in a temporary “move” state.

So to fix this, add the following code to section #4 in `tableView:moveRowAtIndexPath:toIndexPath:`

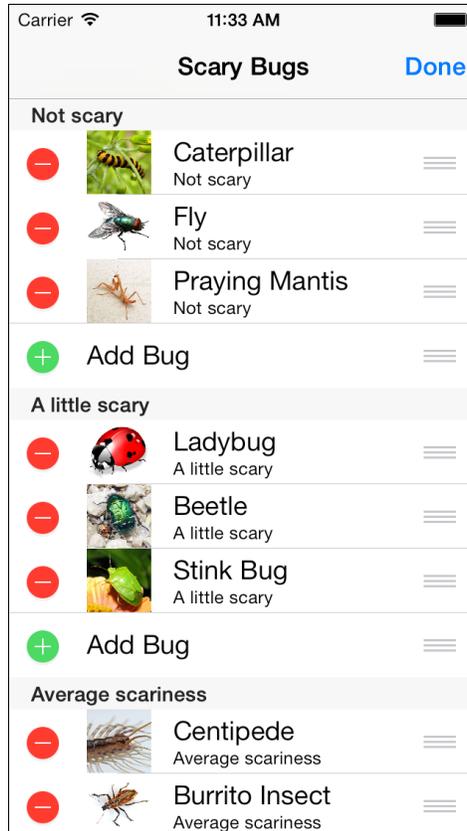
```
double delayInSeconds = 0.2;
dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW,
    (int64_t)(delayInSeconds * NSEC_PER_SEC));
dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
    [self.tableView reloadRowsAtIndexPaths:@[destinationIndexPath]
        withRowAnimation:UITableViewRowAnimationNone];
});
```

This is some code that executes a block of code on the main thread after a delay of 0.2 seconds to refresh the moved cell, effectively waiting until the move animation completes. I'm not sure if there is a better way to do this; if anyone comes up with



a good way, please let me know. (And by the way don't say reloadData on table view; the goal is to reload a single cell, not the entire table view!)

Build and run and you'll see if you move a bug to a new section it reloads correctly now:



However, there's one bug left. Currently if you try to move a row right below the "Add Bug" row, the app will crash.

It doesn't make sense to allow a user to move a row below the Add Bug row, so you need a way to prevent the user from moving a row to certain spots. To do this, add the following new method:

```
- (NSIndexPath *)tableView:(UITableView *)tableView
targetIndexPathForMoveFromRowAtIndex:(NSIndexPath *)sourceIndexPath
toProposedIndexPath:(NSIndexPath *)proposedDestinationIndexPath
{
    BugSection *bugSection =
        self.bugSections[proposedDestinationIndexPath.section];
    if (proposedDestinationIndexPath.row >= bugSection.bugs.count) {
        return [NSIndexPath indexPathForRow:bugSection.bugs.count-1
            inSection:proposedDestinationIndexPath.section];
    } else {
        return proposedDestinationIndexPath;
    }
}
```



```
}  
}
```

This method is called whenever the user tries to drag a row to a new spot, and gives you a chance to say yay or nay (by returning an alternative spot).

Here you check to see if the user is trying to drag the row below the "Add Bug" row, and if so you return the index path for the last valid spot before that point.

Build and run, and you can now reorder in style!

