Extracted from:

iPhone SDK Development

Building iPhone Applications

3.9 Deleting rows

Deleting rows from your table views is so easy that you basically get it for free. You just add a button to your navigation bar to put your list in edit mode and then make the UITableViewDataSourcetableView:commitEditingStyle:forRowAtIndexPath: handle a delete request. This method handles both insertions and deletions, determined by the edit style passed in. For now we only care about deletion though since we don't yet have an interface for adding to the list.

To activate this method we need to be able to put our table view in edit mode. It turns out this has been made really easy for us (surprise, surprise). Check out the commented out implementation of viewDidLoad that XCode generated for us:

Download TableViews/BasketballTeams/Classes/RootViewController.m

```
- (void)viewDidLoad {
        // Add the following line if you want the list to be editable
    // self.navigationItem.leftBarButtonItem = self.editButtonItem;
}
```

Just uncomment that line :-). It automatically adds a button labeled "Edit" on the left side of our navigation bar. Build & Go and then press the Edit button. Our table view in edit mode will look like Figure 3.6, on the following page. Pretty awesome don't you think? The red system minus buttons appearing to the left of each row and the Edit button's label changing to "Done" is all taken care of for you automatically. If you press one of the red minus signs a Delete button will even appear over on the other side of the row. Once you press that our tableView:commitEditingStyle:forRowAtIndexPath: will actually be called. So all we have to do is implement a couple lines in that.

If you search for tableView:commitEditingStyle:forRowAtIndexPath: in RootViewController.m you'll find that, as usual, XCode has provided us with a commented out sample implementation. It turns out, as usual, that's it's pretty close to what we want. For now ignore the second conditional in the method as that has to do with inserting a new row. Take a look at the UITableViewCellEditingStyleDelete condition.

Download TableViews/BasketballTeams/Classes/RootViewController.m

```
-  (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
 forRowAtIndexPath:(NSIndexPath *)indexPath {
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
```

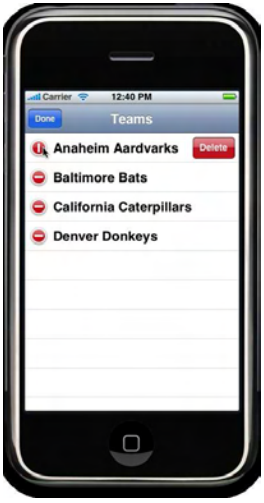

Figure 3.6: Table view in edit mode

```
                          withRowAnimation:YES];
        [teams removeObjectAtIndex:indexPath.row];
    }
```

We need to do three things when a row is deleted, and the sample code already does one of them: We need to delete the row from table view by using UITableView's deleteRowsAtIndexPaths:withRowAnimation: method, then we need to actually remove that item from our data, in this case our teams array. Since we're passed in the index path, we just have to add a call to removeObjectAtIndex: to our teams array. Lastly, we need to wrap the code that updates the table view in calls to its beginUpdates and endUpdates methods so that it knows changes have been made and can adjust its layout accordingly.

Download TableViews/BasketballTeams/Classes/RootViewController.m

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
 forRowAtIndexPath:(NSIndexPath *)indexPath {
    [tableView beginUpdates];
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
                          withRowAnimation:YES];
        [teams removeObjectAtIndex:indexPath.row];
    }
    [tableView endUpdates];
```

```
}
```

3.10 Inserting rows

Our basketball team table view is coming along but it only has a fraction of the teams we want to follow. What about the Edmonton Egrets and the Fresno Ferrets? We'll need to implement a data entry interface to get all these other teams into our table view.

So far we've been living and working inside of the RootViewController that XCode provided us. It's time to branch out. We'll need another view and view controller to display the interface for adding a new team. To trigger this chain of events we'll need another button in our navigation bar to go along with the Edit button. This one will be an Add button that appears on the right hand side of the navigation bar.

Back in Section 3.9, *Deleting rows*, on page 49 when we needed an edit button and the thoughtful engineers at Apple came to the rescue by providing us will an already implemented UIBarButtonItem object via the built in editButtonItem method. There's no built in corresponding addButtonItem but the good news is that it's easy enough to implement.

To mimick the API already provided to us with the editButtonItem we'll go setup a addButtonItem instance variable plus a property over in our RootViewController.h header file.

Download TableViews/BasketballTeams/Classes/RootViewController.h

```
@interface RootViewController : UITableViewController {
        NSMutableArray *teams;
        UIBarButtonItem *addButtonItem;
}
@property (nonatomic, retain) NSMutableArray *teams;
@property (nonatomic, retain) UIBarButtonItem *addButtonItem;
@end
```

Once we've synthesized the declared property and made sure to clean up after ourselves in the dealloc method by releasing the addButtonItem we can initialize our button in the initWithCoder: method. For this we'll use UIBarButtonItem's initWithBarButtonSystemItem:target:action: initializer method. You can stick the following code after the line where the teams property is set:

Download TableViews/BasketballTeams/Classes/RootViewController.m

```
self.addButtonItem = [[UIBarButtonItem alloc]
                      initWithBarButtonSystemItem:UIBarButtonSystemItemAdd
```

```
                                        target:self
                                        action:@selector(addButtonWasPressed)];
```

It turns out that UIBarButtonItem ships with many partially configured system buttons [1]. The add button is here and has a plus sign as its label. It goes by the name UIBarButtonSystemItemAdd so that's what we passed as the argument. Next we're just saying that the current class will implement the method that the button press invokes and that the method, or "action" in target/action parlance, will be the yet-to-be-implemented addButtonWasPressed.

Now that we have the button stored in our addButtonItem property, let's set the rightBarButtonItem in viewDidLoad right after we assign the leftBarButtonItem.

Download TableViews/BasketballTeams/Classes/RootViewController.m

```
- (void)viewDidLoad {
        self.navigationItem.leftBarButtonItem = self.editButtonItem;
        self.navigationItem.rightBarButtonItem = self.addButtonItem;
}
```

Our button has been added but we haven't yet implemented the addButtonWasPressed method yet. Let's quickly stub out a simple implementation so we can test our work. It'll just log to the console when the button is pressed so we know we've hooked everything up correctly. Declare addButtonWasPressed in RootViewController.h.

Download TableViews/BasketballTeams/Classes/RootViewController.h

```
- (void)addButtonWasPressed;
```

And over in the implementation file's addButtonWasPressed we add the call to NSLog().

```
- (void)addButtonWasPressed {
        NSLog(@"Add button pressed");
}
```

With that done, we can now Build & Go. If we look at XCode's console (Run->Console) while we press the add button we'll see the message logged as in Figure 3.7, on the following page.

1. To see the full list of buttons look up UIBarButtonSystemItem in XCode's documentation browser. These buttons are automatically localized also!

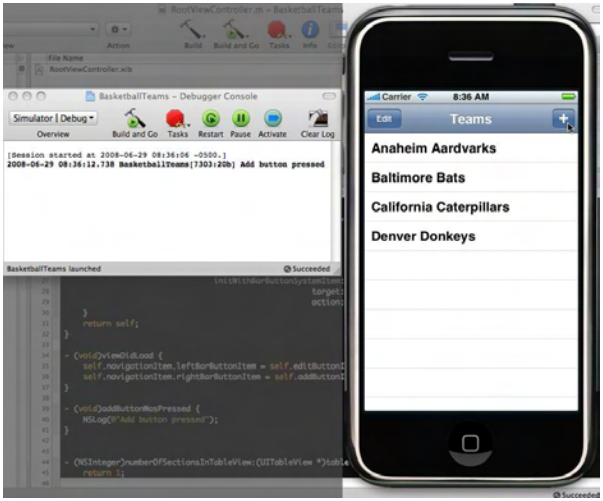

Figure 3.7: Logging add button presses

3.11 Sliding in a view for adding data

Alright, now we're ready to design the interface that's going to let us enter team names to add. In this scenario, a common pattern for data entry is to have a view slide in overtop of our list when the add button is pressed. That view will contain a text field for adding the team name. Once we've entered the name and press save, the view is dismissed and we are returned to an updated list of teams. The view's interface will be really simple: just a single text field.

In Interface Builder let's create a new interface by selecting File->New from the menu (or Command-N). Choose the View template. When our empty view pops up let's start off by saving it immediately into our project. Press Command-S and when prompted we'll name our interface file AddTeamViewController and save it to the top level of our XCode project directory. XCode will ask you if we want to add it to the project. We do. Make sure you check the box next to the project name. The .nib will now appear in the Groups & Files section of you XCode sidebar. Drag and drop it into the Resources group (where our other .nibs are.)

We need a text input field in our view to type in the title of the team we want to add. For that we need a UITextField. In Interface Builder's Library palatte, type "text" in the search field. When the text field appears

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style, and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

iPhone SDK Development's Home Page
http://pragprog.com/titles/amiphd
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragprog.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragprog.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragprog.com/news
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragprog.com/titles/amiphd.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	www.pragprog.com/catalog
Customer Service:	orders@pragprog.com
Non-English Versions:	translations@pragprog.com
Pragmatic Teaching:	academic@pragprog.com
Author Proposals:	proposals@pragprog.com