Table Edit

In this tutorial we will modify the table view project from the previous tutorial to allow editing of the table items; specifically deleting rows, rearranging rows and adding rows.

Open the Project

Launch Xcode and open the TableView project that you have created.

Delete and Rearrange Rows

Edit the ViewController.h file

Modify the **ViewController.h** file as shown below. The changes are bolded in the listing.

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
    <UITableViewDelegate, UITableViewDataSource>
{
    NSMutableArray *listData;
    UITableView *listTable;
}

@property (nonatomic, retain) NSMutableArray *listData;
@property (nonatomic, retain) IBOutlet UITableView *listTable;

- (IBAction)editPressed:(id)sender;

@end
```

Edit the ViewController.m file

Modify the **ViewController.m** file as shown below.

1) Add this line after the other @synthesize line that you already have.

```
    @synthesize listTable;
```

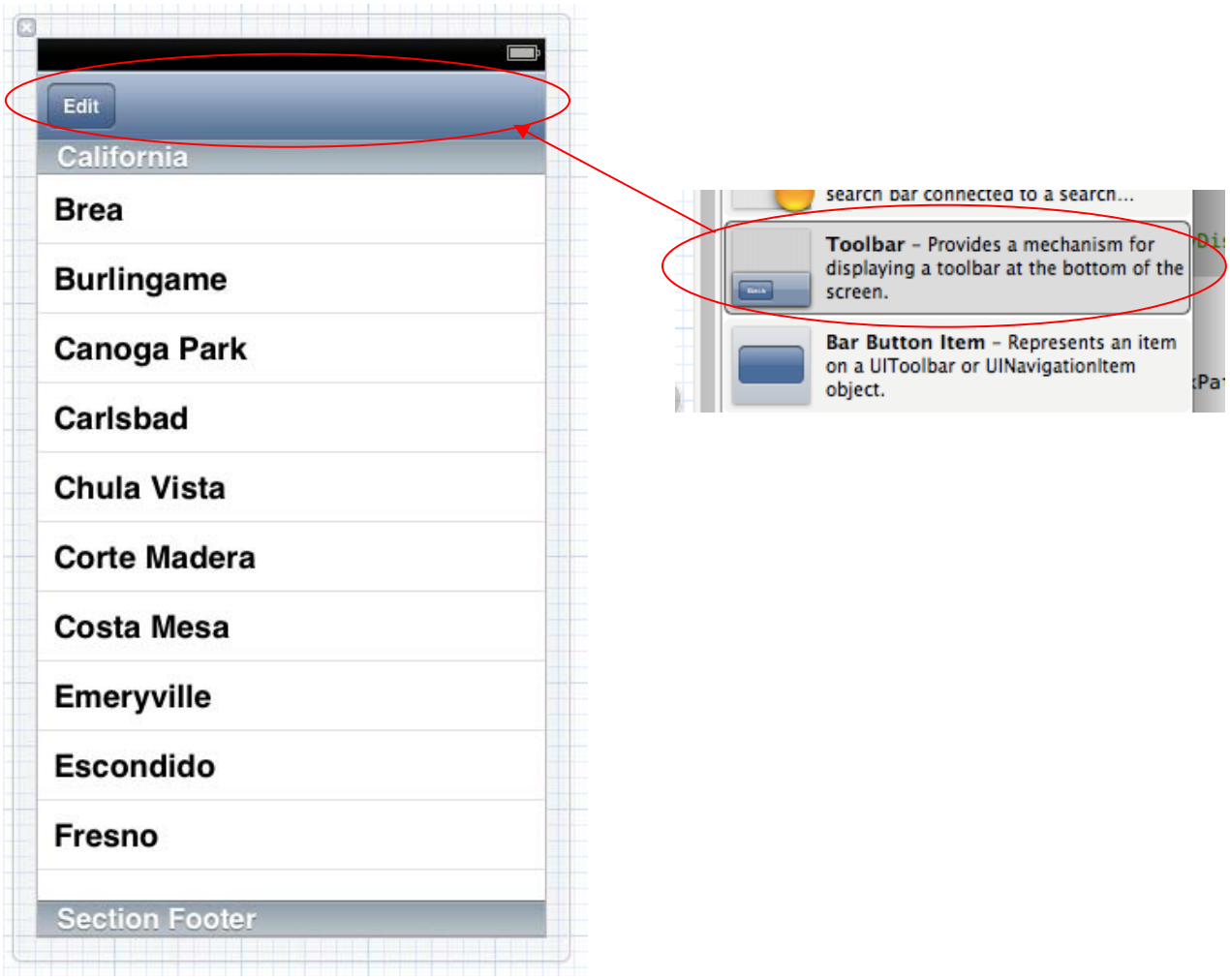
2) Add the following routines to delete and rearrange the rows at the end right before the @end line.

```
// Toggles between editing and not editing modes
- (IBAction)editPressed:(id)sender
{
    if(self.editing) {
        [super setEditing:NO animated:NO];  // turn off editing
        [listTable setEditing:NO animated:NO];
        [listTable reloadData];
        self.navigationItem.rightBarButtonItem.title = @"Edit";
    } else {
        [super setEditing:YES animated:YES];    // turn on editing
        [listTable setEditing:YES animated:YES];
        [listTable reloadData];
        self.navigationItem.rightBarButtonItem.title = @"Done";
    }
}

// The editing style for a row is the kind of button displayed to the left of
the cell when in editing mode.
// either the red minus sign for delete or the green plus sign for add
- (UITableViewCellEditingStyle)tableView:(UITableView *)aTableView
editingStyleForRowAtIndexPath:(NSIndexPath *)indexPath
{
    // No editing style if not editing or the index path is nil.
    if (self.editing == NO || !indexPath)
      return UITableViewCellEditingStyleNone;
    // Determine the editing style based on whether the cell is a placeholder
for adding content or already
    // existing content. Existing content can be deleted.
    if (self.editing && indexPath.row == ([listData count])) {
      return UITableViewCellEditingStyleInsert;   // display the + icon
    } else {
      return UITableViewCellEditingStyleDelete;   // display the - icon
    }
```

```
}

// Override to support editing the table view.
// Update the data model according to edit actions delete or insert.
- (void)tableView:(UITableView *)tv
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle
forRowAtIndexPath:(NSIndexPath *)indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        // delete the row from the data source
        [listData removeObjectAtIndex:indexPath.row];
        // update the table view
        [tv deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
withRowAnimation:UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        /*
        // bring up the add new item controller view
        AddNewItemController *newItemController = [[AddNewItemController
alloc] init];
        newItemController.delegate = self;      // need this line for the
callback to addNewItemSave

        [self presentViewController:newItemController animated:YES
completion:nil];
        [newItemController release];
         */
    }
}

// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:(NSIndexPath
*)indexPath
{
    // Return NO if you do not want the item to be re-orderable.
    if (indexPath.row < [listData count]) {
      return YES;
    } else {
      return NO;
    }
}

// Override to support rearranging the table view.
// Process the row move. This means updating the data model to correct the
item indices.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath
*)fromIndexPath toIndexPath:(NSIndexPath *)toIndexPath
{
    NSString *item = [[listData objectAtIndex:fromIndexPath.row] retain];
    [listData removeObjectAtIndex:fromIndexPath.row];
    [listData insertObject:item atIndex:toIndexPath.row];
    [item release];
}
```

Edit the ViewController.xib file

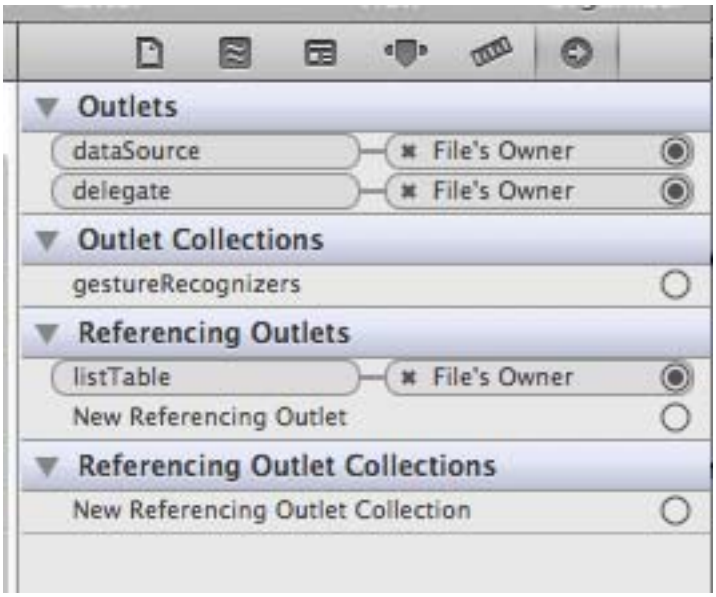
1) Insert a Toolbar above the Table View object in your **ViewController.xib** file as shown below. You'll need to adjust the height of the Table View



2) Change the Toolbar button title to **Edit**. To select the Toolbar button, you need to first single-click on the Toolbar to select the toolbar and then single-click on the button to select the button.

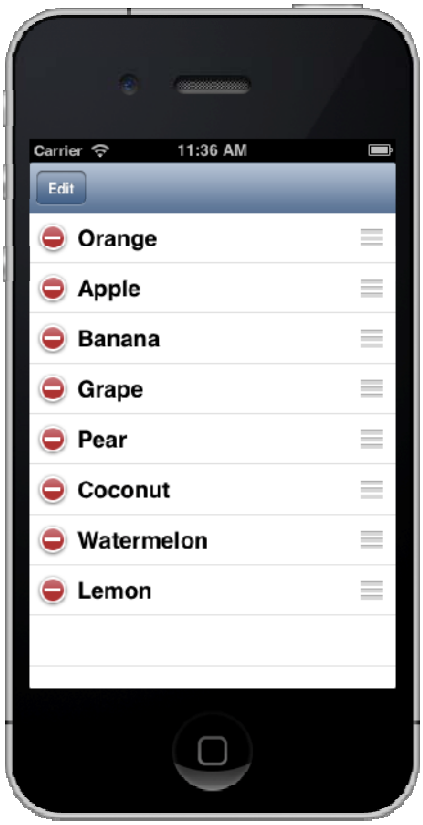
3) Connect the Toolbar button's Sent Actions selector to **editPressed:**

4) Connect the Table View's Referencing Outlets to **listTable**.



Run

Press Run to see the result. Click on the Edit button in the Toolbar to go into edit mode. Once in edit mode, you can delete or rearrange the rows. Click on the Edit button again to get out of the edit mode.



Add New Rows

Edit the ViewController.h file

Modify the **ViewController.h** file as follows.

1) Add this line after the other #import line that you already have.

```
#import "AddNewItemController.h"
```

Edit the ViewController.m file

Modify the **ViewController.m** file as shown below.

1) Modify the **numberOfRowsInSection** routine to match the following.

```
// sets the number of rows in the table
- (NSInteger)tableView:(UITableView *)tableView
numberOfRowsInSection:(NSInteger)section
{
     int count = [listData count];
     if (self.editing) {
       count++;    // one extra row for add when in editing mode
     }
    return count;
}
```

2) In the **cellForRowAtIndexPath** routine, replace the textLabel line

```
// specify the content of each cell
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath: (NSIndexPath *) indexPath
{

...

    // the cell content is obtained from the listData indexed by the row
number
  cell.textLabel.text = [listData objectAtIndex:indexPath.row];
```

with the following:

```
// specify the content of each cell
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath: (NSIndexPath *) indexPath
{

...

    // the cell content is obtained from the listData indexed by the row
number
     if (indexPath.row < [listData count]) {
       cell.textLabel.text = [listData objectAtIndex:indexPath.row];
       cell.textLabel.textColor = [UIColor blackColor];
     } else {
       cell.textLabel.text = @"Add new fruit";
       cell.textLabel.textColor = [UIColor lightGrayColor];
     }

     return cell;
```

3) Modify the **commitEditingStyle** routine by adding the bolded lines below to match the following.

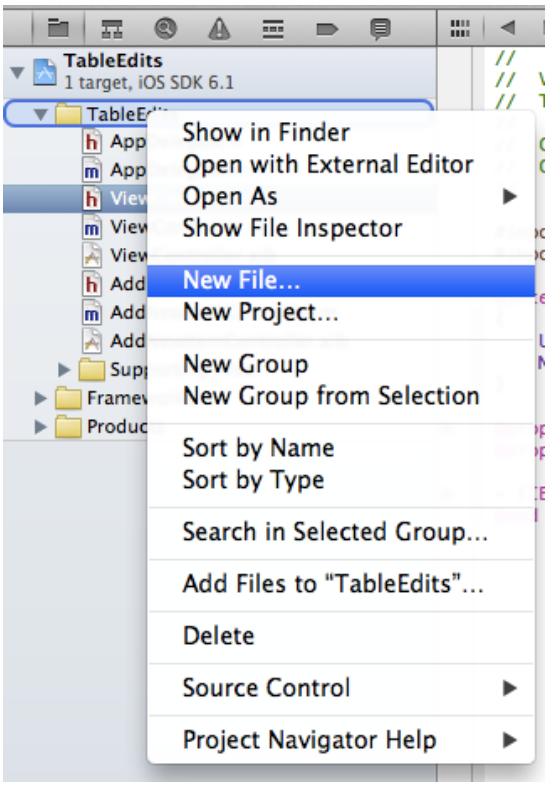
```
// Override to support editing the table view.
// Update the data model according to edit actions delete or insert.
- (void)tableView:(UITableView *)tv commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)
indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        // delete the row from the data source
        [listData removeObjectAtIndex:indexPath.row];
        // update the table view
        [tv deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
withRowAnimation:UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {

        // bring up the add new item controller view
         AddNewItemController *newItemController =
[[AddNewItemController alloc] init];
         newItemController.delegate = self;  // need this line for the
callback to addNewItemSave

         [self presentViewController:newItemController animated:YES
completion:nil];
         [newItemController release];

    }
```

4) Add the following **addNewItem** routine to the end right before the @end line.

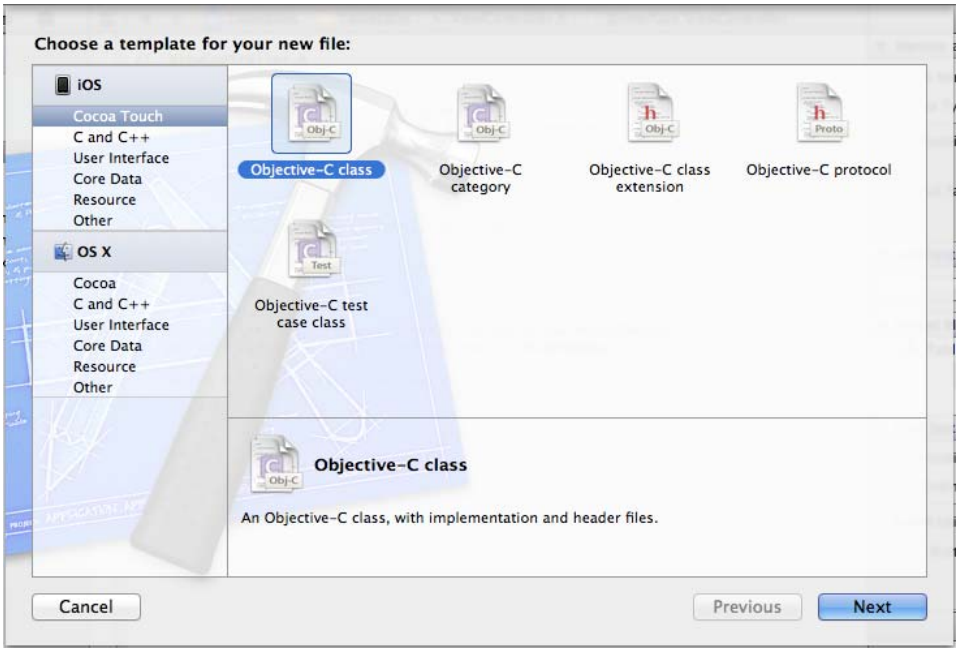
```
// this method is called when the Done button in the AddNewItem view is
pressed
- (void)addNewItem:(NSString *)string
{
    if ([string length] > 0) {
        if (listData == nil) {
            listData = [[NSMutableArray alloc] init];
        }
      [listData insertObject:string atIndex:[listData count]];
      [listTable reloadData];
    }
}
```

Add the AddNewItemController

1) Right-click on your project folder and select **New File**.



2) In the **Choose a template for your new file** window, select **Objective-C class**. Then click Next.



3) Do the following in the **Choose options for your new file** window

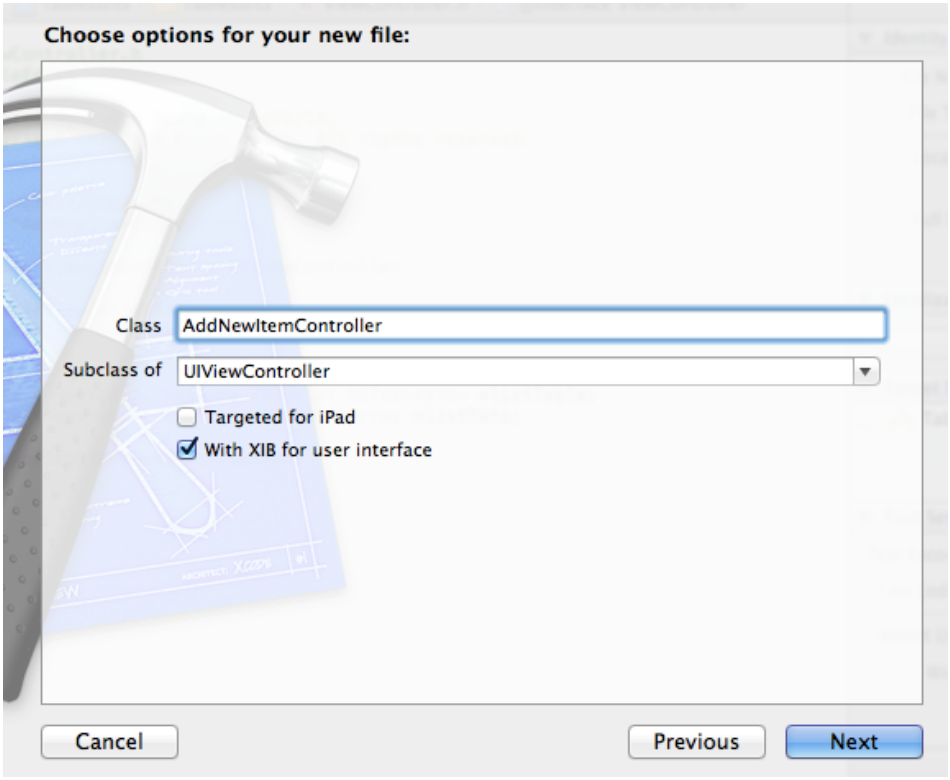
   a) Type in **AddNewItemController** for the Class name.

   b) Select **UIViewController** for the Subclass.

   c) Check the **With XIB for user interface** box.

   d) Click **Next**.

   e) Click **Create**.



Edit the AddNewItemController.h file

Modify the **AddNewItemController.h** file as shown below.

```
#import <UIKit/UIKit.h>

@protocol AddNewItemDelegate <NSObject>
    -(void)addNewItem:(NSString *)string;   // this method is in
ViewController.m
@end

@interface AddNewItemController : UIViewController {
    id delegate;
    UITextField *txtNewItem;
}

@property (nonatomic, retain) id delegate;
@property (nonatomic, retain) IBOutlet UITextField *txtNewItem;

-(IBAction)saveNewItem;

@end
```

Edit the AddNewItemController.m file

Modify the **AddNewItemController.m** file as shown below.

```
#import "AddNewItemController.h"

@implementation AddNewItemController

@synthesize delegate;
@synthesize txtNewItem;

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)viewWillAppear:(BOOL)animated
{
    //self.navigationItem.title = @"Add Fruit";
}

- (IBAction)saveNewItem
{
    [self.delegate addNewItem:txtNewItem.text];
    [self dismissViewControllerAnimated:YES completion:nil];
}
```

Edit the AddNewItemController.xib file

1) Insert a Text Field object to your **AddNewItemController.xib** file.
2) Connect the Text Field's **Did End On Exit** event to **saveNewItem**.
3) Connect the Text Field's **Referencing Outlets** to **txtNewItem**.

Run

Press Run to see the result. Click on the Edit button in the Toolbar to go into edit mode. You should see the Add new fruit line with a green plus sign at the end of the list. Click on the plus sign to get the next screen to type a new fruit. Press return and the new fruit will be added to the list.