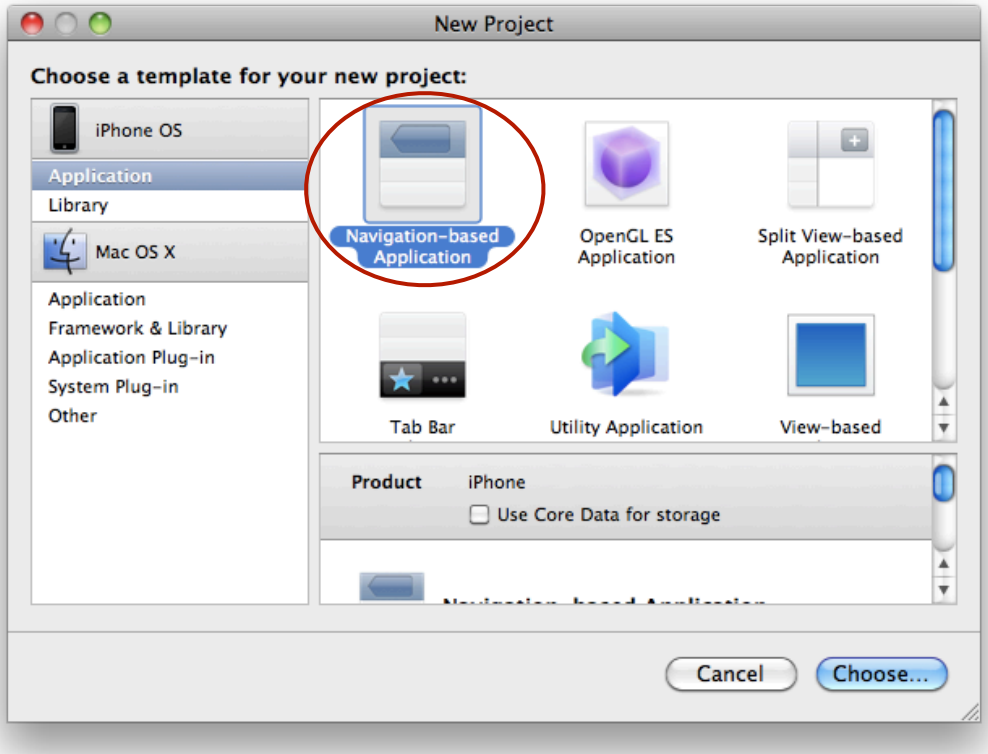
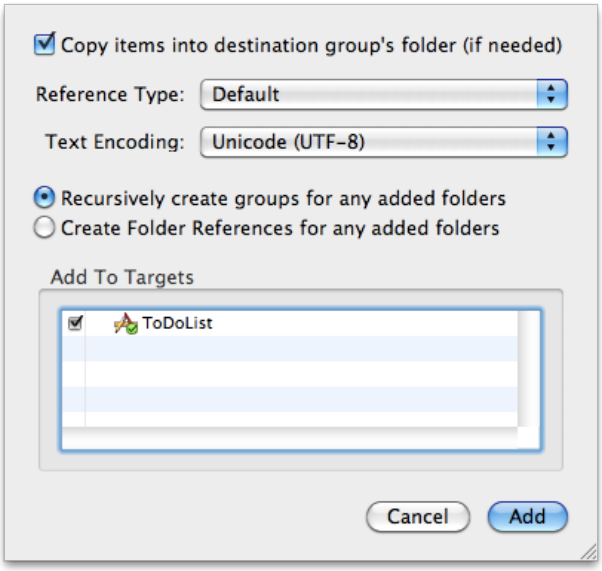
Lab ToDoList

Step1. 在File開啓一個新的project, 選擇 Navigation based application, 將project取名為 ToDoList

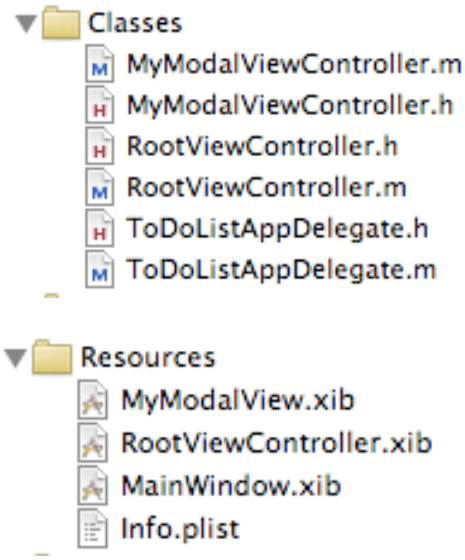


Step2. 我們要使用 lab_SimpleModalView 實作過的 class, 直接將

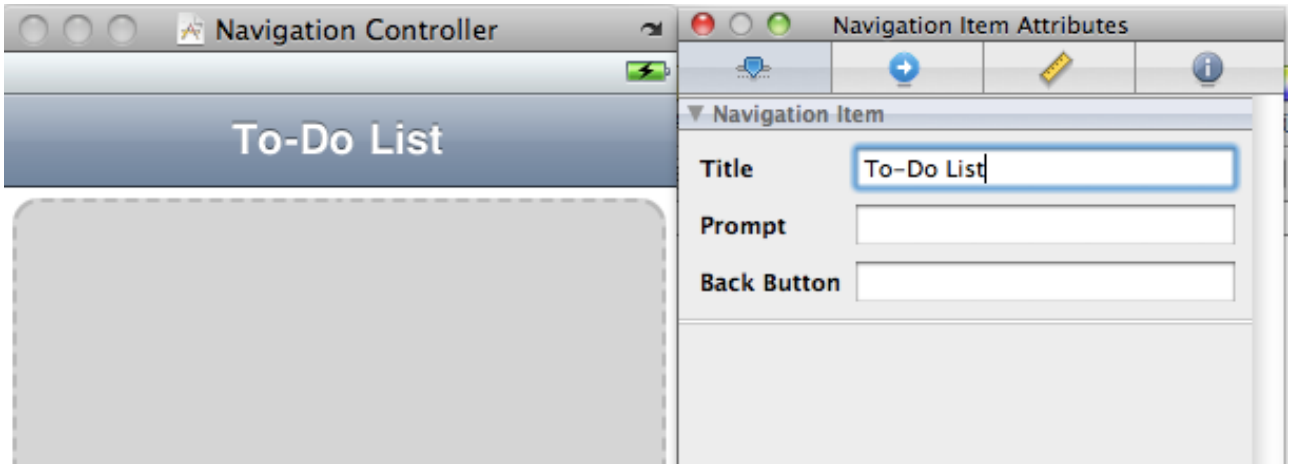
MyModalViewController.h, MyModalViewController.m, MyModalView.xib 由 lab_SimpleModalView 的資料夾拖曳到左方的 Groups & Files,當出現個提示的時候記得將上方的 Copy items into destination group's filder 打勾.



這時候應該有這些檔案



Step 3. Resources >MainWindow.xib, 點選上方的 navigation bar, 在 inspector的attribute設定 title 為, To-Do List



Step 4. Classes > MyModalViewController.h , 稍作改變,

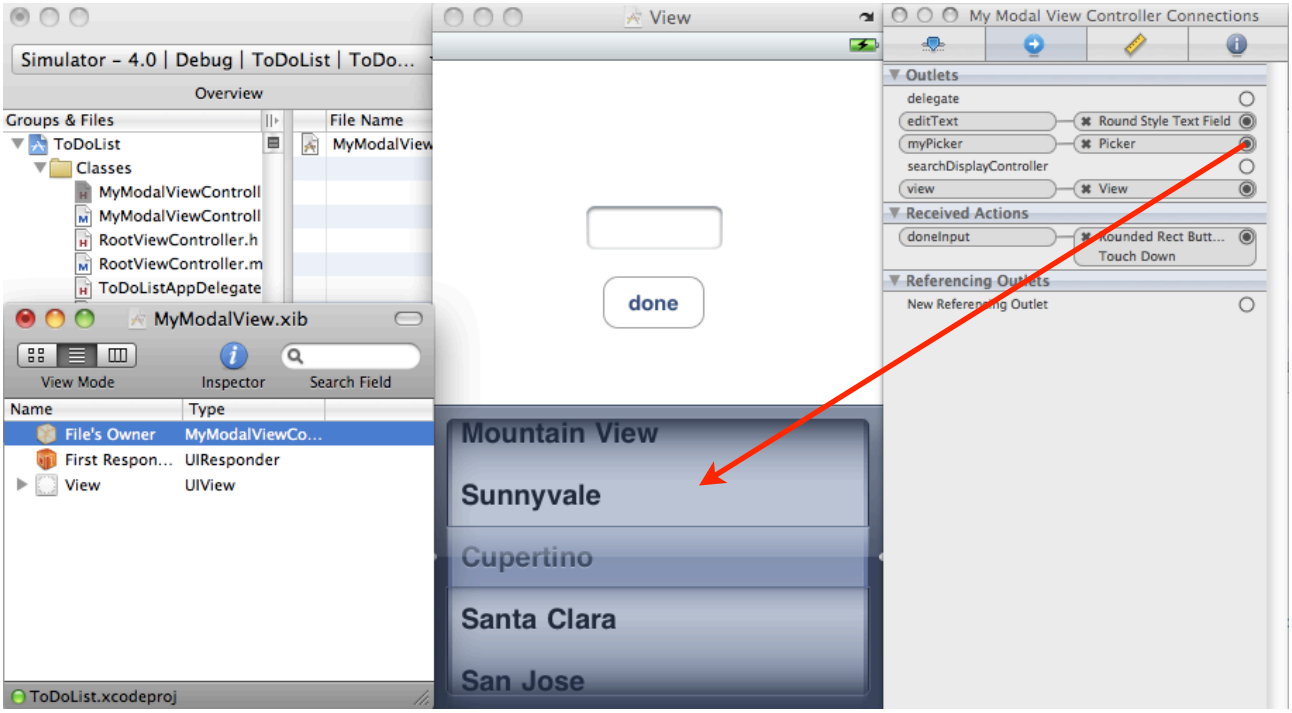
```
@interface MyModalViewController : UIViewController
<UIPickerViewDataSource,UIPickerViewDelegate> {
    IBOutlet UITextField* editText;
    id<SendTextDelegate> delegate;
    IBOutlet UIPickerView* myPicker;
    NSMutableArray* myArray;
    NSInteger selectPicker;
}
```

加入一個 UIPickerView 物件, 一個 NSMutableArray, 一個 NSInteger (使用 int 也可)

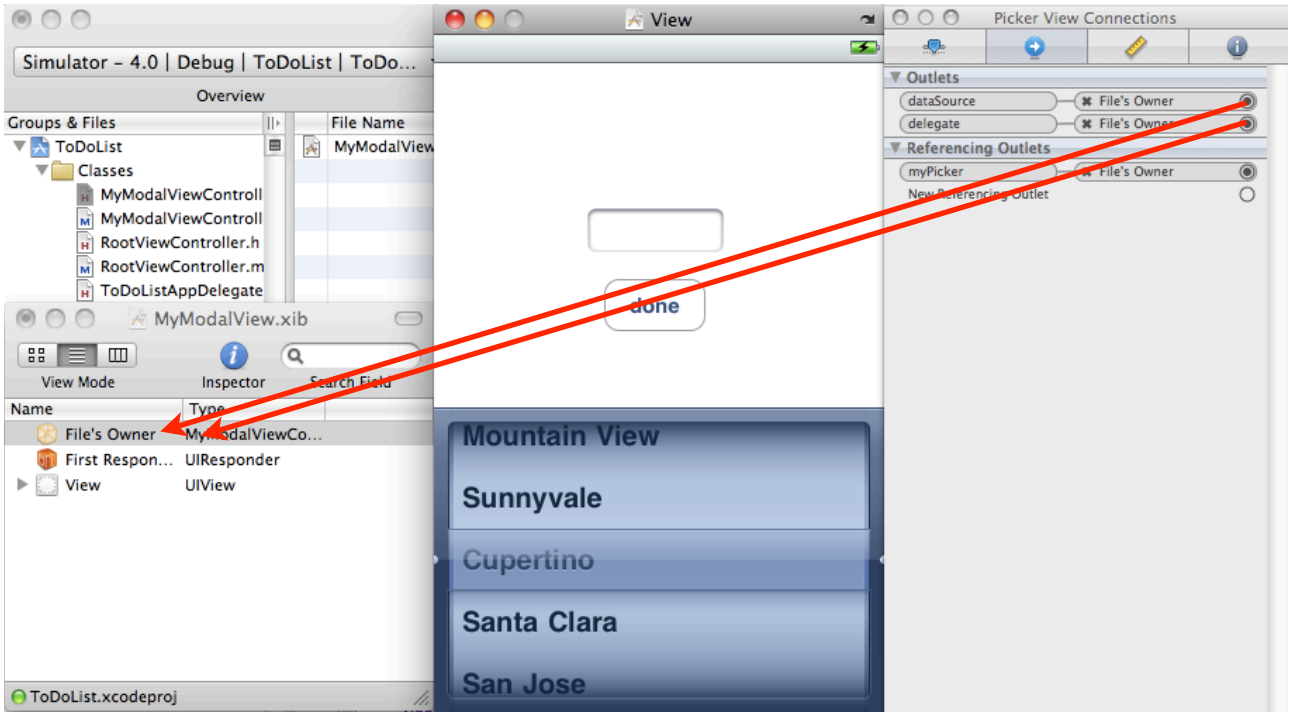
Step 5. protocol的action也稍作改變, 多一個selector,目的是要傳回picker的值

```
@protocol SendTextDelegate <NSObject>
-(void) modalViewController:(MyModalViewController *)
modalController didGenString:(NSString *) targetString withPicker:
(NSInteger) row;
@end
```

Step 6. 在MyModalView.xib放上一個 picker, 點File's Owner,連myPicker到view中的Picker



Step 7. 點選 picker, 將 dataSource 以及delegate 都連結到 File's owner



Step 8. Classes > MyModalViewController.m, 找到 -(IBAction) doneInput,修改成

```
-(IBAction) doneInput
{
     if([self.delegate respondsToSelector:@selector
(modalViewController:didGenString:withPicker:)]){
          [self.delegate modalViewController:self
didGenString:editText.text withPicker:selectPicker];

}
```

Step 9. 找到 -(id) initWithNibName , 把Mark去掉,將 myArray 以及 myPicker 初始化

```
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    if (self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil]) {
        // Custom initialization
            myArray = [[NSMutableArray alloc]
initWithObjects:@"Business",@"Personal",nil];
            myPicker = [[UIPickerView alloc] init];
            [self.view addSubview:myPicker];
            selectPicker = 0;

    }
    return self;
}
```

Step 10. 接著我們實作 myPicker 的設定以及 action

```
- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)
thePickerView {
     return 1;
}

- (NSInteger)pickerView:(UIPickerView *)thePickerView
numberOfRowsInComponent:(NSInteger)component {
     return [myArray count];
}

- (NSString *)pickerView:(UIPickerView *)thePickerView
titleForRow:(NSInteger)row forComponent:(NSInteger)component {
     return [myArray objectAtIndex:row];
}

- (void)pickerView:(UIPickerView *)thePickerView didSelectRow:
(NSInteger)row inComponent:(NSInteger)component {
     selectPicker  = row;
}
```

上面,

```
- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)
thePickerView {
    return 1;
}
```

設定picker的component

```
- (NSInteger)pickerView:(UIPickerView *)thePickerView
numberOfRowsInComponent:(NSInteger)component {
    return [myArray count];
}
```

設定 picker 有幾個 row

```
- (NSString *)pickerView:(UIPickerView *)thePickerView
titleForRow:(NSInteger)row forComponent:(NSInteger)component {
    return [myArray objectAtIndex:row];
}
```

這一段在picker被設定的時候, 會被呼叫好幾次,目的是將每個設定的row給一個物件, 我們是使用 myArray 將物件給 picker.

最後這一段

```
- (void)pickerView:(UIPickerView *)thePickerView didSelectRow:
(NSInteger)row inComponent:(NSInteger)component {

    selectPicker  = row;
}
```

是當picker的某一個row被選取的時候要執行的動作, 我們這裡將被選到的row, 其 integer數字傳給 selectPicker,可以藉由 protocol的action傳到外部的ViewController.

Step 11. Classes > RootViewController.h,記得import MyModalViewController.h

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

@interface RootViewController : UITableViewController
<SendTextDelegate> {
    NSMutableArray* businessData;
    NSMutableArray* personalData;
    UIBarButtonItem* addButton;
}
-(void) addRow;
@end
```

我們放置兩個 mutablearray 以及一個 barbuttonitem, 另外, addRow這個action將會和addButton連結.

Step 12. Classes > RootViewController.m, 首先找到 ViewDidLoad

```
- (void)viewDidLoad {
    businessData = [[NSMutableArray alloc]
initWithObjects:@"Meeting", @"Consulting", @"Make a call", nil];
    personalData = [[NSMutableArray alloc]
initWithObjects:@"Party", @"Shopping",nil];
    addButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemAdd target:self
action:@selector(addRow)];
    // Uncomment the following line to display an Edit button in
the navigation bar for this view controller.
     self.navigationItem.rightBarButtonItem = self.editButtonItem;
[super viewDidLoad];
}
```

將兩個 array 及 barbutton 作初始化,

```
addButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemAdd target:self
action:@selector(addRow)];
```

在這一段中, 也將這個button的action設定為addRow.

Step 13. 在numberOfSectionsInTableView, 修改section數目

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 2;
}
```

Step 14. 在numberOfRowsInSection, 加入設定一個section中有幾個row,

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section {
    if(section == 0)
            return [businessData count];
     if(section == 1)
            return [personalData count];
     else
            return 0;
}
```

我們一共有兩個section, 上方的 section是 section =0, 我們把row的個數設定為 businessData array 的大小, 將下方的section設定為 personalData 的大小.

Step 15. 接著加入設定section的名稱

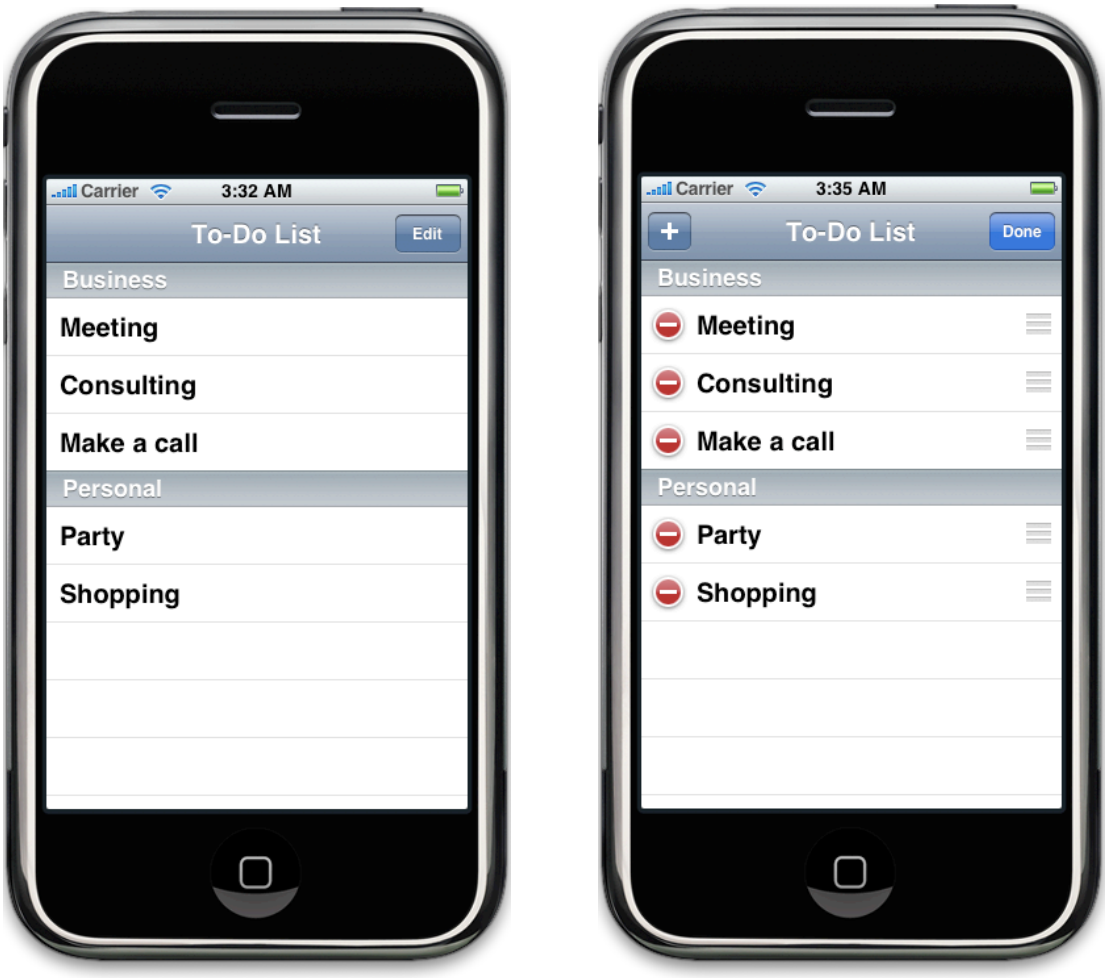
```
- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:
(NSInteger)section
{
      if(section == 0)
            return @"Business";
      if(section == 1)
            return @"Personal";
      else
            return @"none";
}
```

Step 16. 接著修改

```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithFrame:CGRectZero
reuseIdentifier:CellIdentifier] autorelease];
    }

    // Configure the cell.
     // Add the code below
     if(indexPath.section == 0)
            cell.text = [businessData objectAtIndex:indexPath.row];
     else if(indexPath.section == 1)
            cell.text = [personalData objectAtIndex:indexPath.row];
     else
            cell.text = nil;
    return cell;
}
```

將tableView 的row填入array中的物件.

Step 17. 這時候的外觀應該會像這樣, 按下 edit 顯示右圖



Step 18. 我們接著要去實作右圖的功能, 首先我們實作刪除 row的功能

```
- (void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath{

     if(editingStyle== UITableViewCellEditingStyleDelete){
           if(indexPath.section==0)
           {
                [businessData removeObjectAtIndex:indexPath.row];
           }
           else
           {
                [personalData removeObjectAtIndex:indexPath.row];
           }
           [tableView deleteRowsAtIndexPaths:[NSArray
arrayWithObject:indexPath] withRowAnimation:YES];
     }
}
```

table view上的row和 array 是緊密結合的, 所以我們必須在這個action裡刪除array的值以及 table view 的row.

Step 19. 在上一頁的右圖中, 每個row右邊有三條線的地方是可以拖曳的, 我們實作它的功能

```
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)
fromIndexPath toIndexPath:(NSIndexPath *)toIndexPath {
     if(fromIndexPath.section != toIndexPath.section)
     {
           if(fromIndexPath.section == 0)
           {
                 [personalData insertObject:[businessData
objectAtIndex:fromIndexPath.row] atIndex:toIndexPath.row];
                 [businessData
removeObjectAtIndex:fromIndexPath.row];
           }
           else
           {
                 [businessData insertObject:[personalData
objectAtIndex:fromIndexPath.row] atIndex:toIndexPath.row];
                 [personalData
removeObjectAtIndex:fromIndexPath.row];
           }
     }
}
```

要注意的是, 因為row和array緊密結合, 當我們改變row的位置,也需要改變array的內容, 如果某個row移動到所屬的section的其它位置, 因為物件屬於同一個array, 所以系統自動幫我們改變array的值, 當物件被移動到不同section的時候,因為不同section的row對應到不同的array, 所以我們需要將移入的section,其array增加一個物件, 而移出的section其array減少一個物件.

Step 20. 加入(拿掉Mark)下列程式, 使得row可以移動.

```
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:
(NSIndexPath *)indexPath {
     return YES;
}
```

Step 21. 接著我們實作 增加row的功能,

```
-(void)setEditing:(BOOL)editing animated:(BOOL)animated
{
     [super setEditing:editing animated:YES];
     if (editing) {
         self.navigationItem.leftBarButtonItem = addButton;
    } else {
         self.navigationItem.leftBarButtonItem = nil;
    }
}
```

加入這一段程式, 讓左上角的加號按鈕只在 edit 鈕被按下得時候出現.

Step 22. 實作 addRow 這個action, 這個action是對應到左上角的加號按鈕的

```
-(void) addRow
{
     MyModalViewController* myModalViewCon =
[[MyModalViewController alloc] initWithNibName:@"MyModalView"
bundle:nil];
     myModalViewCon.delegate = self;
     [self presentModalViewController:myModalViewCon animated:YES];
     [myModalViewCon release];
}
```

這一段程式和我們在 lab_SimpleModalView 學過的 Get Text 按鈕類似,會呼叫出ModalView.

Step 23. 實作 protocol的action, 我們將使用者輸入的文字回傳回來,當作row的物件放入 array, 並且由 withPicker這個selector取回的row,知道目前picker是選到business還是 personal, 我們將資料放入正確的array. 最後呼叫 tableView的 reloadData , 使我們加入的物件可以被顯示出來.

```
-(void) modalViewController:(MyModalViewController *) modalController
didGenString:(NSString *) targetString withPicker:(NSInteger ) row
{
     if(row==0)
     [businessData insertObject:targetString atIndex:([businessData count])];
     else
     [personalData insertObject:targetString atIndex:([personalData
count])];
     [self setEditing:NO animated:YES];
     [self.tableView reloadData];
     [self dismissModalViewControllerAnimated:YES];
}
```

Step 24. build and GO, 實作的功能都有了