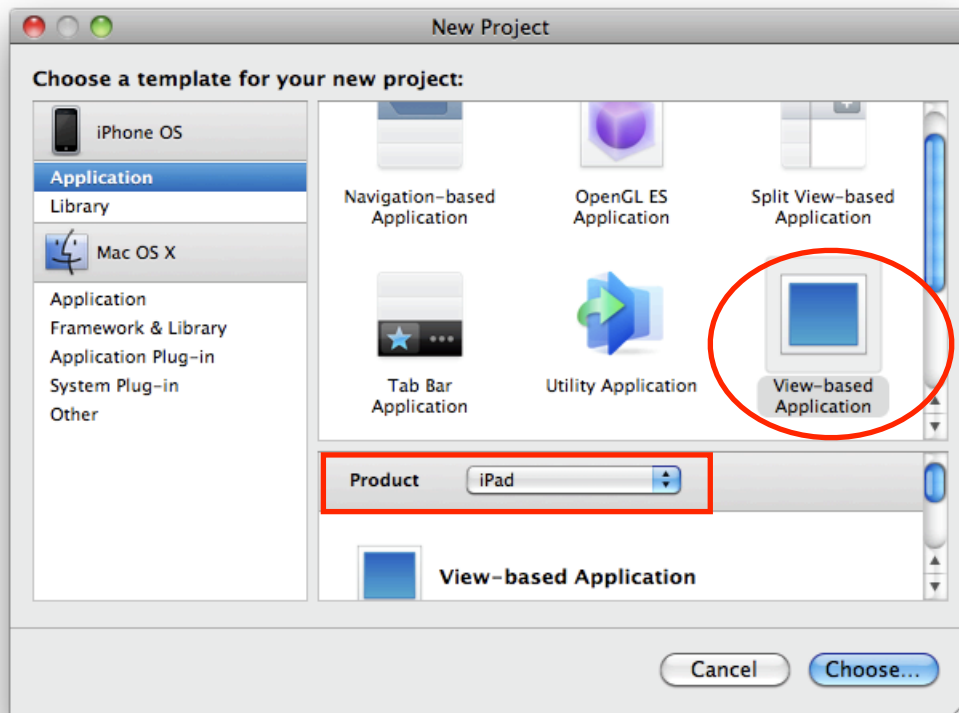
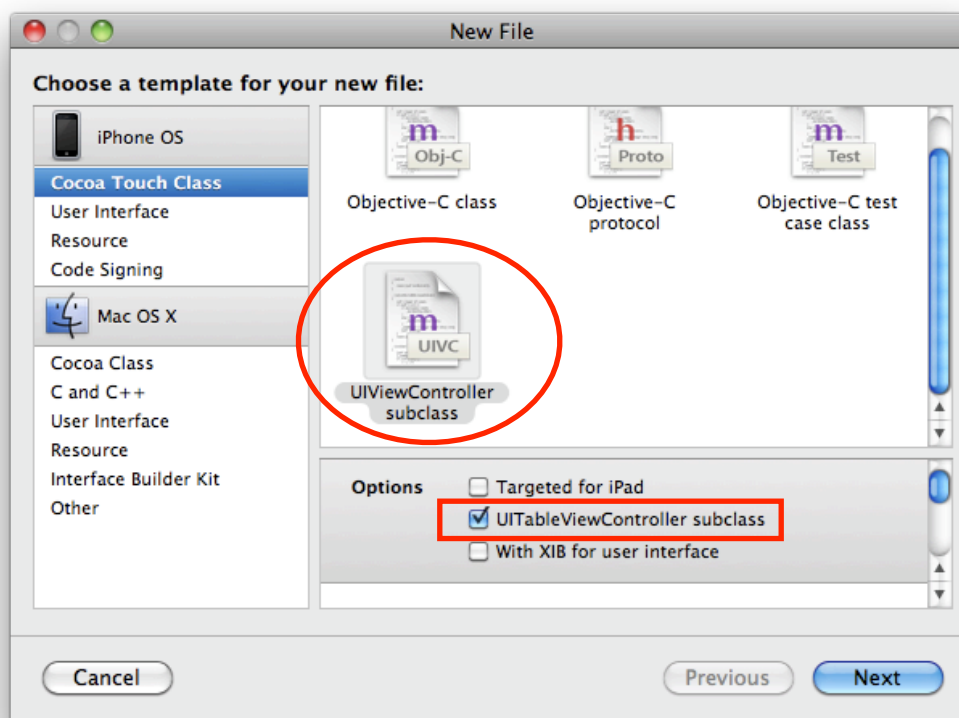


Lab SimplePopover

Step1. 在File開啓一個新的project, 選擇 view based application, Product選擇iPad, 將project取名為 SimplePopover



Step2. 我們新增一個 UIViewController SubClass, 並勾選UITableViewController subclass, 命名為RootController, 來建立我們Popover出來的contentView



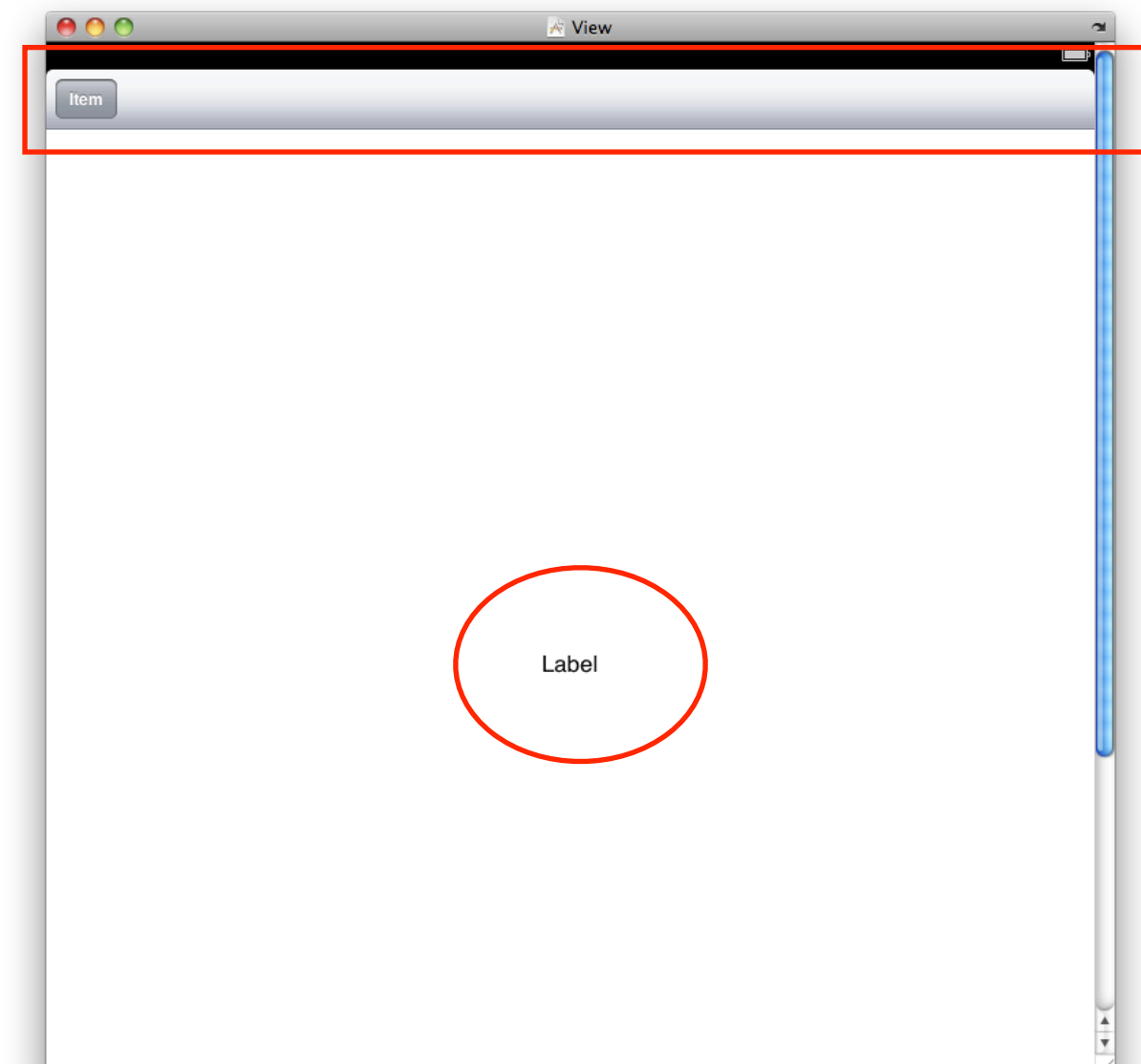
Step3. 在SimplePopoverViewController.h裡面,加入一個UIPopoverController, 以及我們在View中需要的UIToolbar, UILabel, 以及UIBarButtonItem, 以及一個作為傳值使用的id detailItem.

在下面再設定我們變數property,以及兩個我們我們會用到的method: configureView和pop.

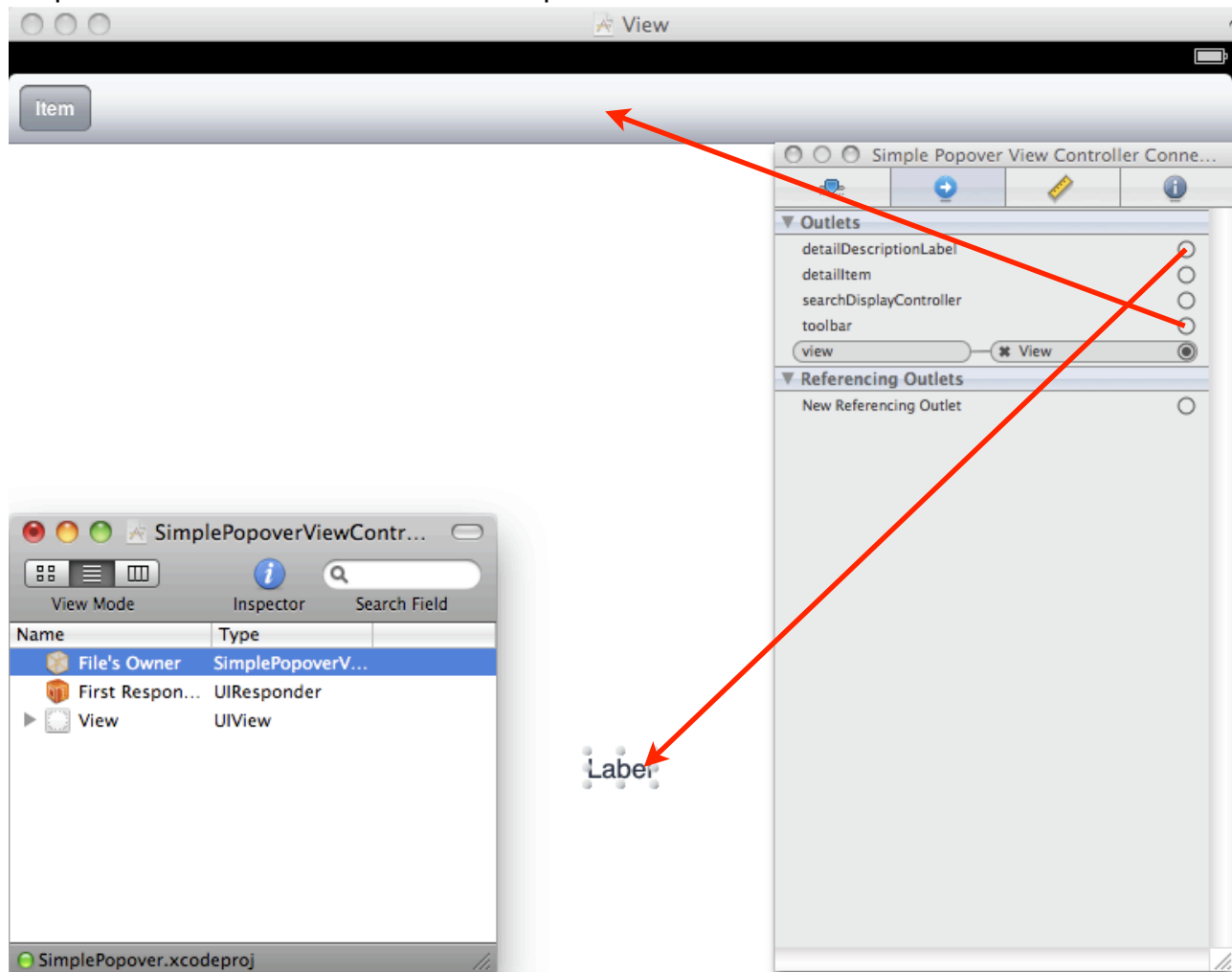
```
#import <UIKit/UIKit.h>
```

```
@interface SimplePopoverViewController : UIViewController {
    UIPopoverController *popoverController;
    IBOutlet UIToolbar *toolbar;
    IBOutlet UILabel *detailDescriptionLabel;
    UIBarButtonItem *barButtonItem;
    id detailItem;
}
@property (nonatomic, retain) UIPopoverController *popoverController;
@property (nonatomic, retain) id detailItem;
- (void)configureView;
- (void)pop;
@end
```

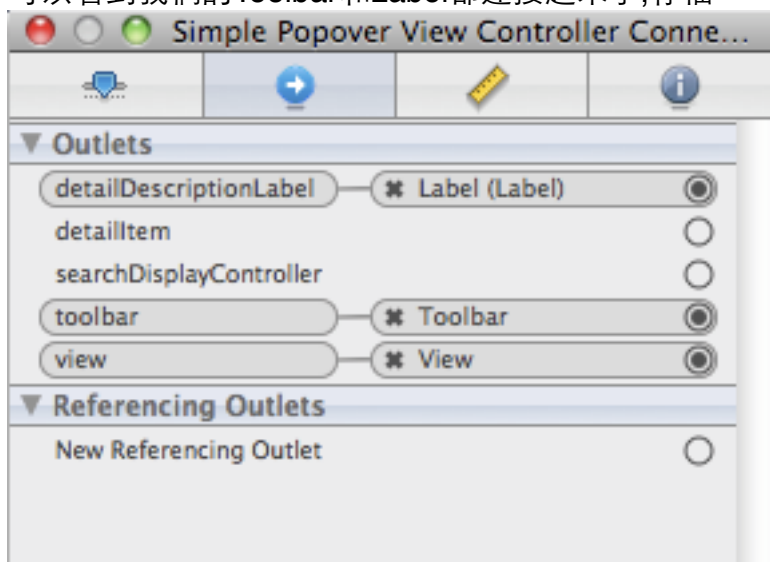
Step4. 存檔後,在Resources資料夾裡開啓SimplePopoverViewController.xib, 在Library裡搜尋然後加入一個Toolbar和一個Label(如下圖示)



Step5. 連接我們的toolbar和detailDescriptionLabel到View上的Toolbar和Label



可以看到我們的Toolbar和Label都連接起來了,存檔



Step6. 在SimplePopoverViewController.m裡面, #import "RootController.h", 並 @synthesize popoverController, detailItem; 找到並把ViewDidLoad{}的Mark去掉, 在ViewDidLoad{}, ViewDidUnload{}, 以及dealloc{}三個method分別加入下列的code, 去設定我們的barButtonItem, 以及最後對我們變數的release.

```
// Implement viewDidLoad to do additional setup after loading the view,  
typically from a nib.
```

```
- (void)viewDidLoad {  
    [super viewDidLoad];  
    barButtonItem = [[toolbar items] objectAtIndex:0];  
    barButtonItem.title = @"Root List";  
    [barButtonItem setAction:@selector(pop)];  
}
```

```
// Override to allow orientations other than the default portrait  
orientation.
```

```
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)  
interfaceOrientation {  
    return YES;  
}
```

```
- (void)didReceiveMemoryWarning {  
    // Releases the view if it doesn't have a superview.  
    [super didReceiveMemoryWarning];  
  
    // Release any cached data, images, etc that aren't in use.  
}
```

```
- (void)viewDidUnload {  
    // Release any retained subviews of the main view.  
    // e.g. self.myOutlet = nil;  
    self.popoverController = nil;  
}
```

```
- (void)dealloc {  
    [super dealloc];  
    [popoverController release];  
    [detailItem release];  
}
```

Step7. 加入下面四個Method, - (void) setDetailItem:(id)newDetailItem {} 是覆蓋原先用property見好的setter, 來加入和完成我們其他想做的功能, 如call configureView{}去更改Label的text, 以及執行 [popoverController dismissPopoverAnimated:YES] 來dismissPopover的ContentView.

-(void) pop{}是alloc一個RootController的實體, 並加入到我們建立的popoverController實體來作為ContentView,並使用我們toolbar上的barButtonItem來pop出來.

-(void) willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation) toInterfaceOrientation duration:(NSTimeInterval)duration{}
是使我們在iPad轉向時(simulator時使用command + 左或右) 去更改toolbar的frame來符合不一樣的空間長度(44是toolbar default設定寬度)

```
- (void)setDetailItem:(id)newDetailItem {
    if (detailItem != newDetailItem) {
        [detailItem release];
        detailItem = [newDetailItem retain];
        [self configureView];
    }
    if (popoverController != nil) {
        [popoverController dismissPopoverAnimated:YES];
    }
}
```

```
- (void)configureView {
    detailDescriptionLabel.text = [detailItem description];
}
```

```
-(void) pop{
    if (popoverController == nil) {
        RootController *root = [[RootController alloc] init];
        popoverController = [[UIPopoverController alloc]
initWithContentViewController:root];
        [root release];
    }

    [popoverController presentPopoverFromBarButtonItem:barButtonItem
permittedArrowDirections:UIPopoverArrowDirectionAny
animated:YES];
}
```

```
- (void) willAnimateRotationToInterfaceOrientation:(UIInterfaceOrientation)
toInterfaceOrientation duration:(NSTimeInterval)duration{
    if (toInterfaceOrientation == UIInterfaceOrientationPortrait){
        toolbar.frame = CGRectMake(0, 0, 768, 44);
    }
    else {
        toolbar.frame = CGRectMake(0, 0, 1024, 44);
    }
}
```

Step8. 在RootController.m裡 import SimplePopoverAppDelegate.h 和 SimplePopoverViewController.h, 並設定Sections數(return 1)和每個Section裡的Row的數量(return 10), 並在 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {}裡加入 cell.textLabel.text = [NSString stringWithFormat:@"Row %d", indexPath.row]; 去設定每個cell裡的Text.

```
#import "RootController.h"
```

```
#import "SimplePopoverAppDelegate.h"
```

```
#import "SimplePopoverViewController.h"
```

```
@implementation RootController
```

```
#pragma mark -
```

```
#pragma mark Table view data source
```

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {  
    // Return the number of sections.  
    return 1;  
}
```

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:  
(NSInteger)section {  
    // Return the number of rows in the section.  
    return 10;  
}
```

```
// Customize the appearance of table view cells.
```

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

```
    static NSString *CellIdentifier = @"Cell";
```

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

```
    // Configure the cell...
```

```
    cell.textLabel.text = [NSString stringWithFormat:@"Row %d", indexPath.row];  
    return cell;  
}
```

Step9. 在 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
(NSIndexPath *)indexPath {}裡加入

```
SimplePopoverAppDelegate *appDelegate = [[UIApplication sharedApplication]
delegate];
appDelegate.viewController.detailItem = [NSString stringWithFormat:@"Row
%d", indexPath.row];
```

來在選擇ContentView裡面的Row之後,在原來View裡的label顯示出來.

```
#pragma mark -
#pragma mark Table view delegate
```

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath {
```

```
// Navigation logic may go here. Create and push another view controller.
```

```
SimplePopoverAppDelegate *appDelegate = [[UIApplication sharedApplication]
delegate];
appDelegate.viewController.detailItem = [NSString stringWithFormat:@"Row
%d", indexPath.row];
```

```
/*
<#DetailViewController#> *detailViewController =
[[<#DetailViewController#> alloc] initWithNibName:@"<#Nib name#>" bundle:nil];
// ...
// Pass the selected object to the new view controller.
[self.navigationController pushViewController:detailViewController
animated:YES];
[detailViewController release];
*/
}
```

```
#pragma mark -
#pragma mark Memory management
```

```
- (void)didReceiveMemoryWarning {
// Releases the view if it doesn't have a superview.
[super didReceiveMemoryWarning];
```

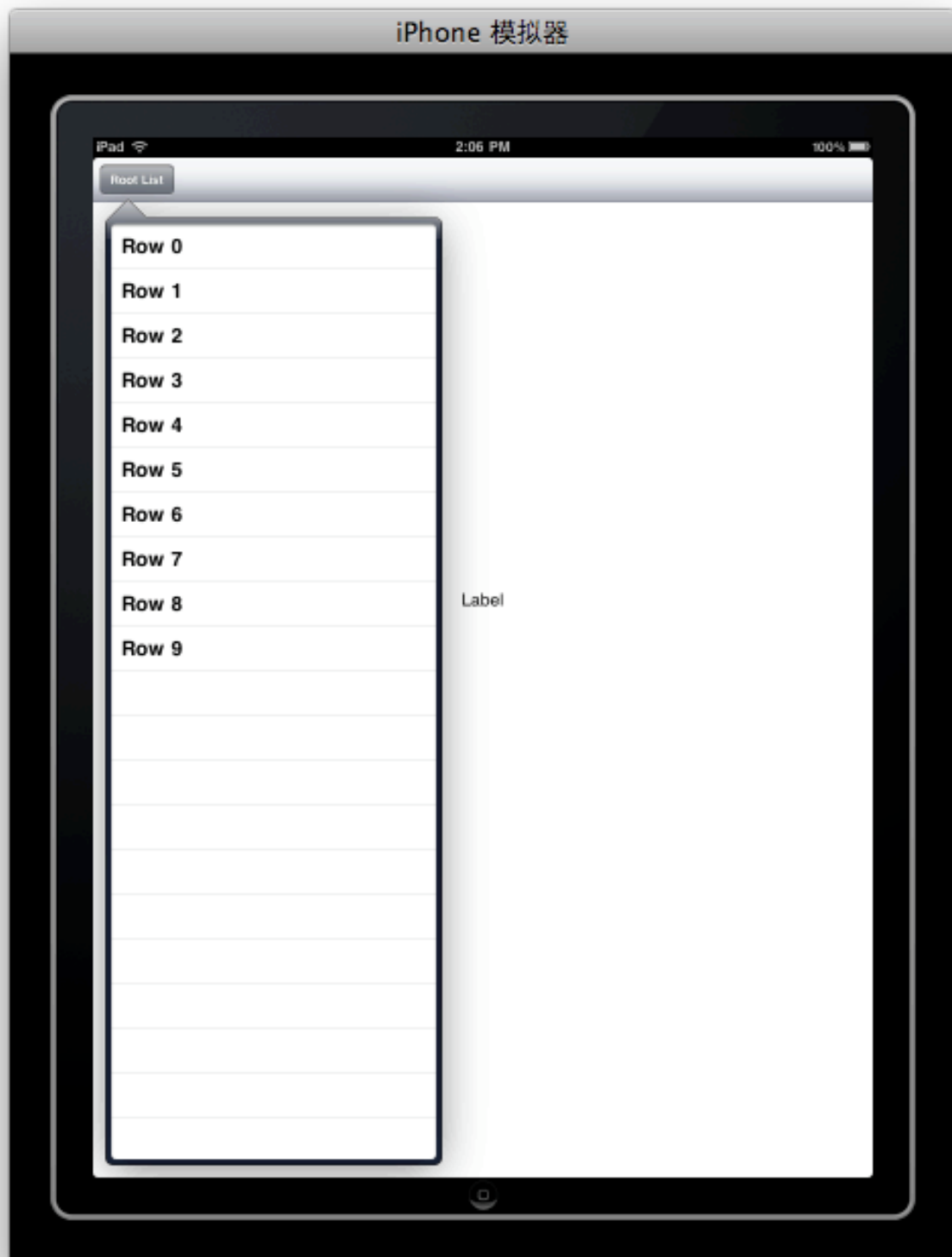
```
// Relinquish ownership any cached data, images, etc that aren't in use.
}
```

```
- (void)viewDidUnload {
// Relinquish ownership of anything that can be recreated in viewDidLoad or
on demand.
// For example: self.myOutlet = nil;
}
```

```
- (void)dealloc {
[super dealloc];
}
```

```
@end
```

Step10. Build and GO
點Root List來Pop我們的ContentView



選擇後在Label顯示我們選擇的Row

