IOS - ACTIONS AND OUTLETS

http://www.tutorialspoint.com/ios/ios_actions_and_outlets.htm

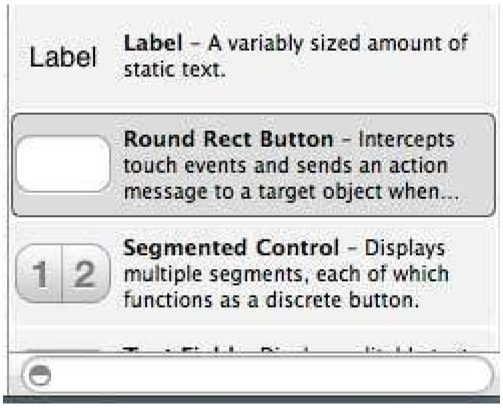
Actions and outlets in iOS are referred to as **ibActions** and **ibOutlets** respectively, where **ib** stands for interface builder. These are related to the UI elements and we will explore them after knowing visually how to implement them.

Actions and Outlets – Steps Involved

Step 1 − Let's use our First iPhone Application.

Step 2 − Select the ViewController.xib file from the files in the navigator section.

Step 3 − Now, you can select the UI elements from the library pane in the right hand side of our window, which is shown below.



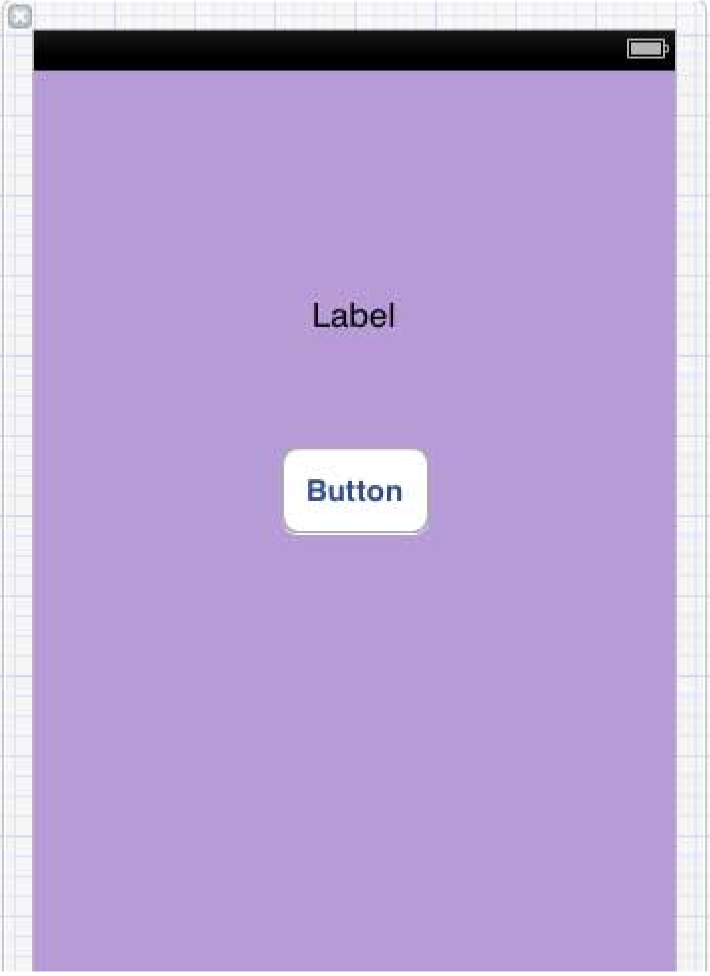
Label — **Label** – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to a target object when...

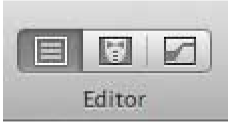
Segmented Control – Displays multiple segments, each of which functions as a discrete button.

Step 4 − You can drag and drop the UI elements to our view in our interface builder.

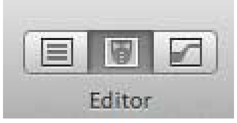
Step 5 − Let us add a Label and Round Rect Button to our view.



Step 6 − From the Editor Selector button in the workspace toolbar found on the top right corner as shown below.

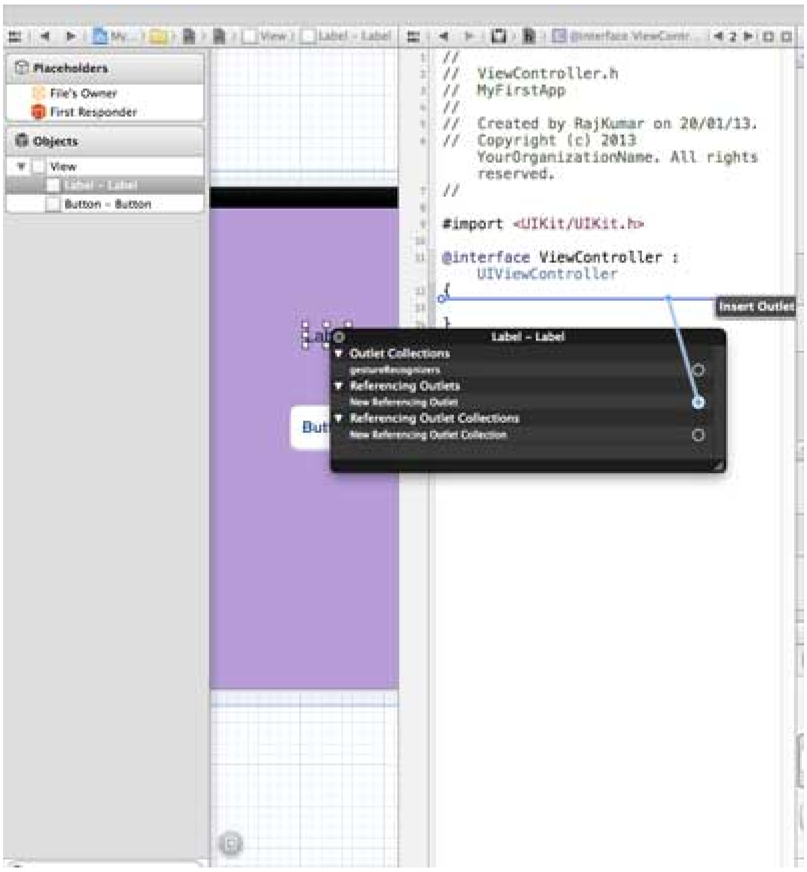


Select Assistant editor button.



Step 7 − We will see two windows in our editor area in the center, one is ViewController.xib file and the other is ViewController.h.

Step 8 − Now, right click on the label and select, hold and drag the new referencing outlet as shown below.



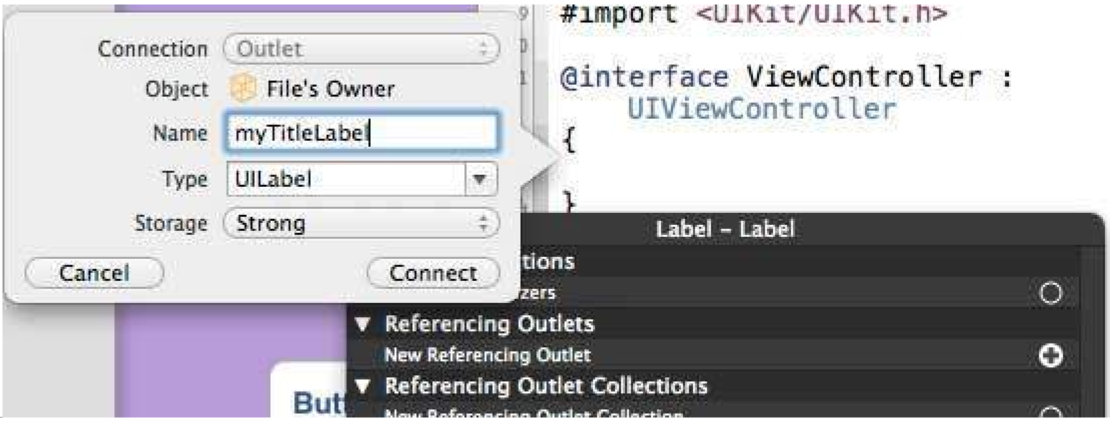
```
//
//  ViewController.h
//  MyFirstApp
//
//  Created by RajKumar on 20/01/13.
//  Copyright (c) 2013
    YourOrganizationName. All rights
    reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController :
    UIViewController
{

}
```

Step 9 − Drop in the ViewController.h in between the curly braces. In case there are no curly braces in the file, add the ViewController before doing this. You will find a pop-up as shown below.



Connection: Outlet
Object: File's Owner
Name: myTitleLabel
Type: UILabel
Storage: Strong

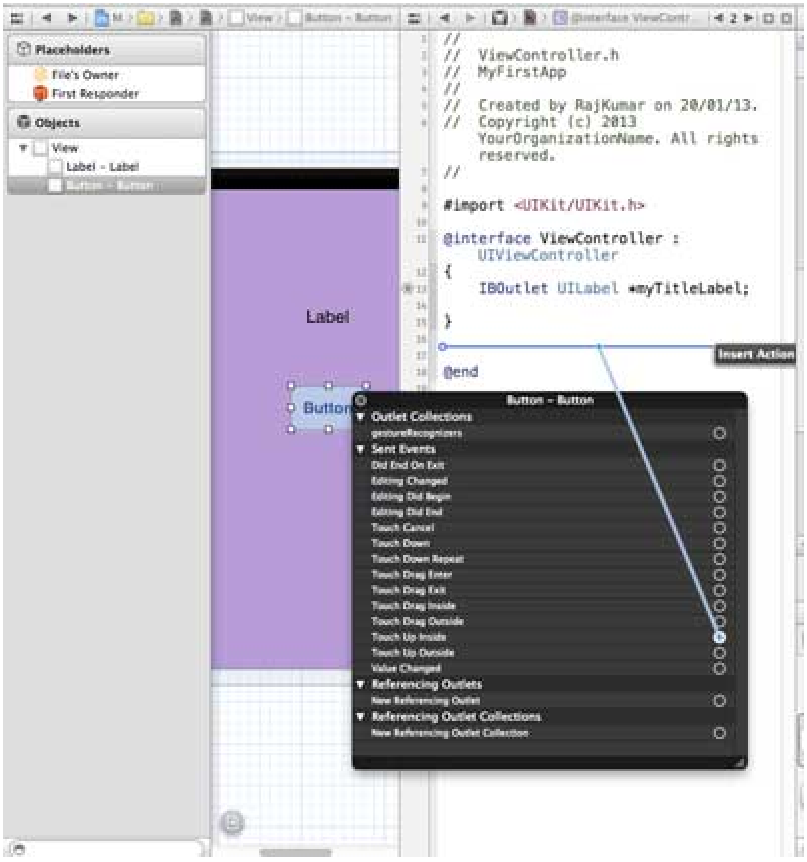
```
#import <UIKit/UIKit.h>

@interface ViewController :
    UIViewController
{

}
```

Step 10 − Type the label name for the outlet, here we have used the label myTitleLabel. Click connect and the ibOutlet will be complete.

Step 11 − Similarly, to add an action, right click the Round rect button, select touch up inside and drag it below the curly braces.



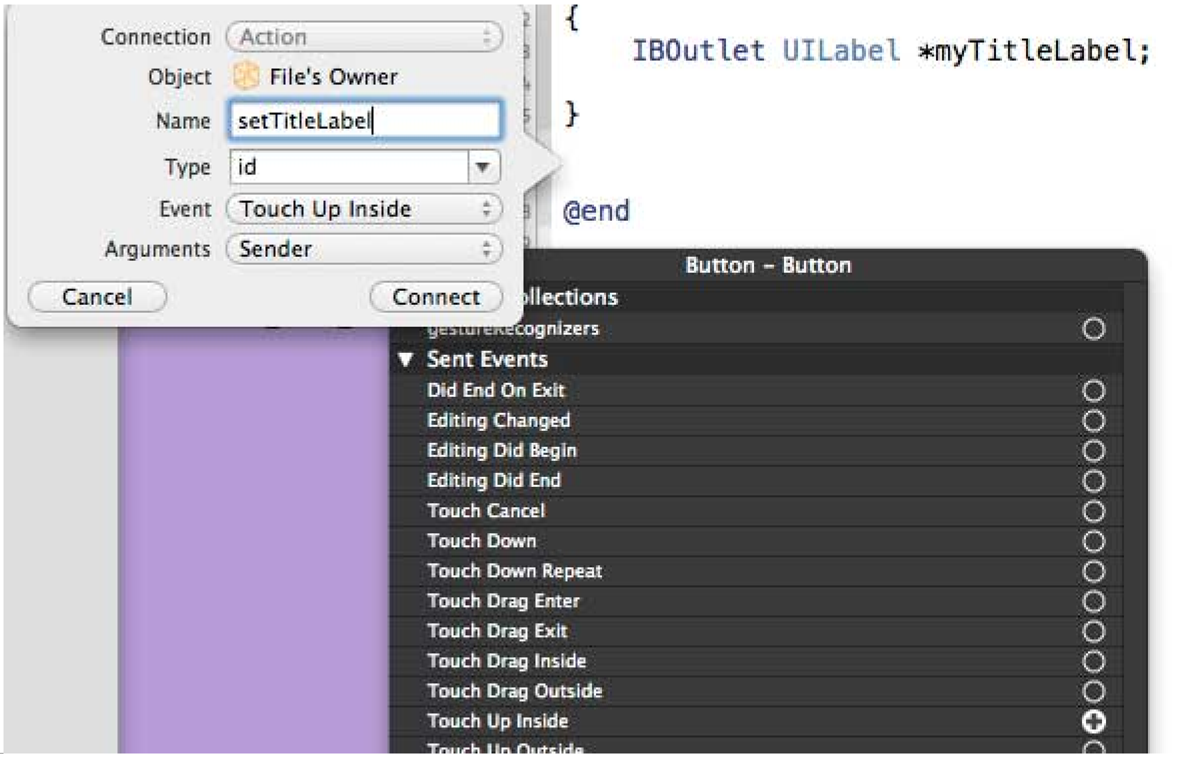
```
//
//  ViewController.h
//  MyFirstApp
//
//  Created by RajKumar on 20/01/13.
//  Copyright (c) 2013
//  YourOrganizationName. All rights
//  reserved.
//

#import <UIKit/UIKit.h>

@interface ViewController :
    UIViewController
{
    IBOutlet UILabel *myTitleLabel;
}

@end
```

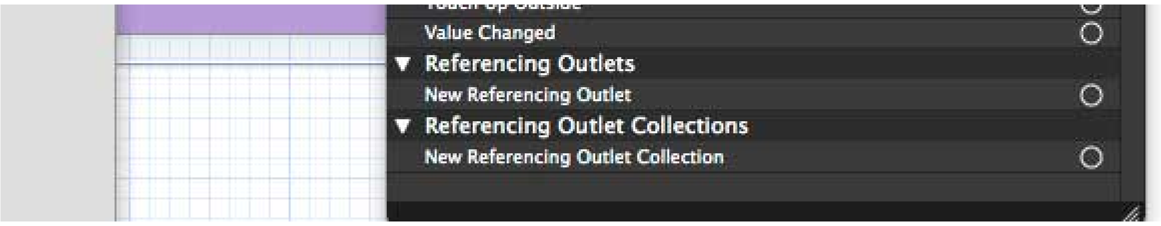
Step 12 − Drop it and name it setTitleLabel.



Connection: Action
Object: File's Owner
Name: setTitleLabel
Type: id
Event: Touch Up Inside
Arguments: Sender

```
{
    IBOutlet UILabel *myTitleLabel;
}

@end
```


Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection

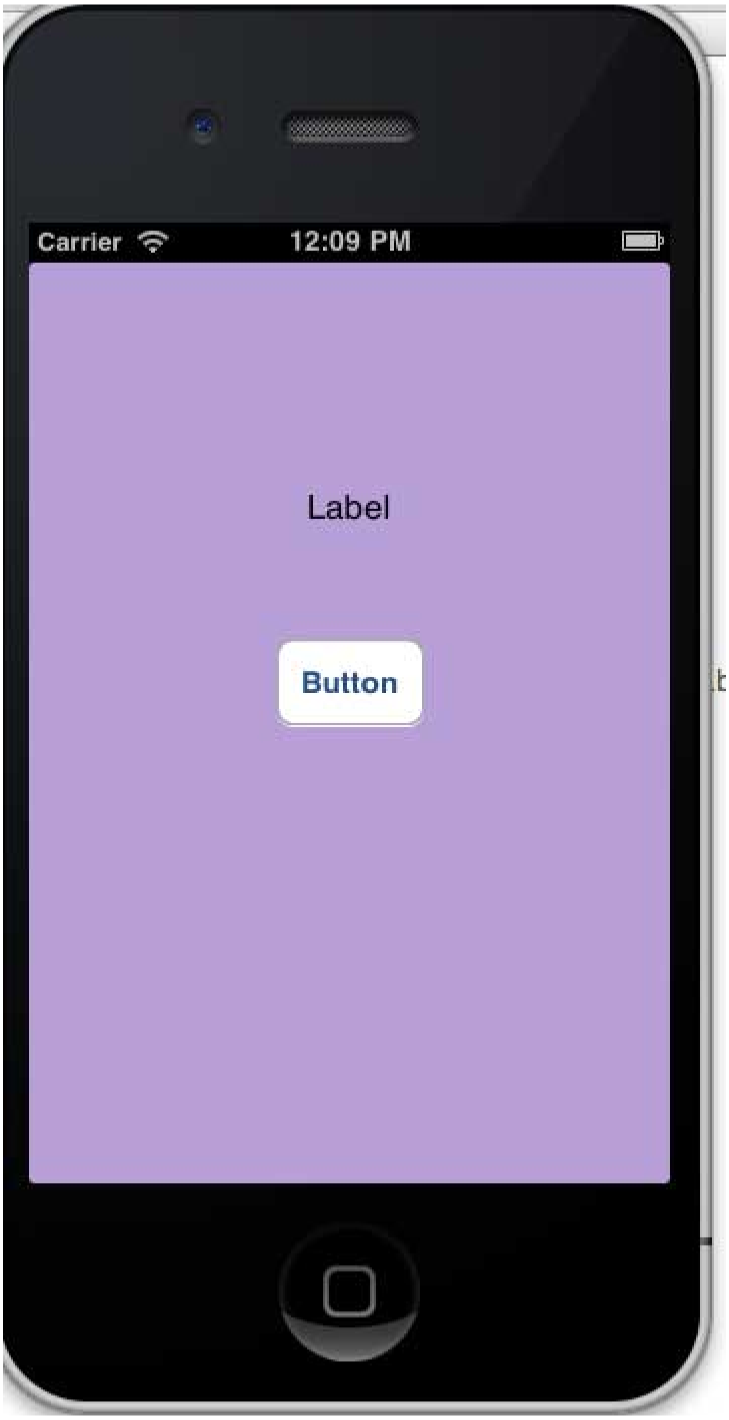
Step 13 − Select ViewController.m file, you'll find a method as shown below.

```
-(IBAction) setTitleLabel:(id)sender{
}
```

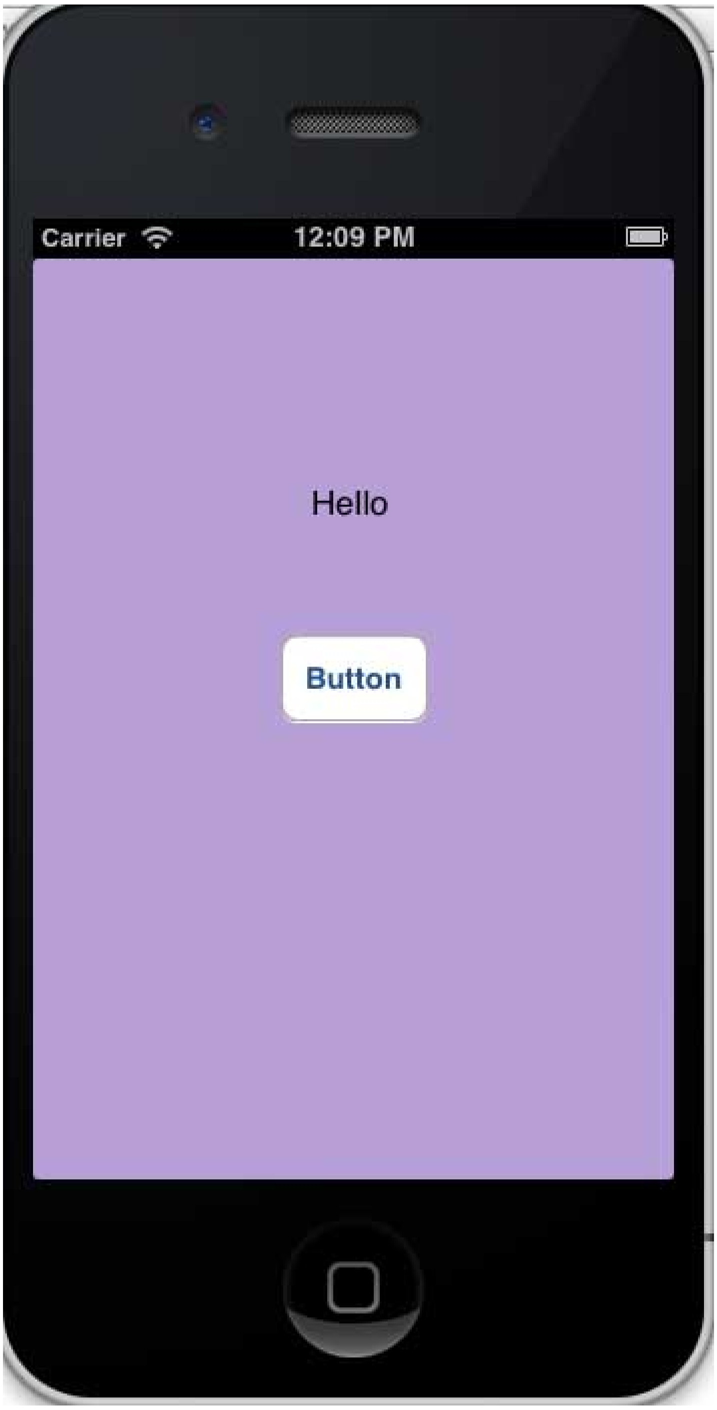
Step 14 − Add a statement as shown below inside the above method.

```
[myTitleLabel setText:@"Hello"];
```

Step 15 − Let us now run the program by selecting the run button. You will see the following output.



Step 16 − Now click the button.



Step 17 − The label that we created have been changed by the action on the button.

Step 18 From the above example, we can conclude that IBOutlet creates a reference to the UIElement *herefortheUILabel*. Similarly, the IBAction links the UIButton with a method, which is called on the event touch up inside.

Step 19 − You can play around with actions by selecting different events while creating the action.