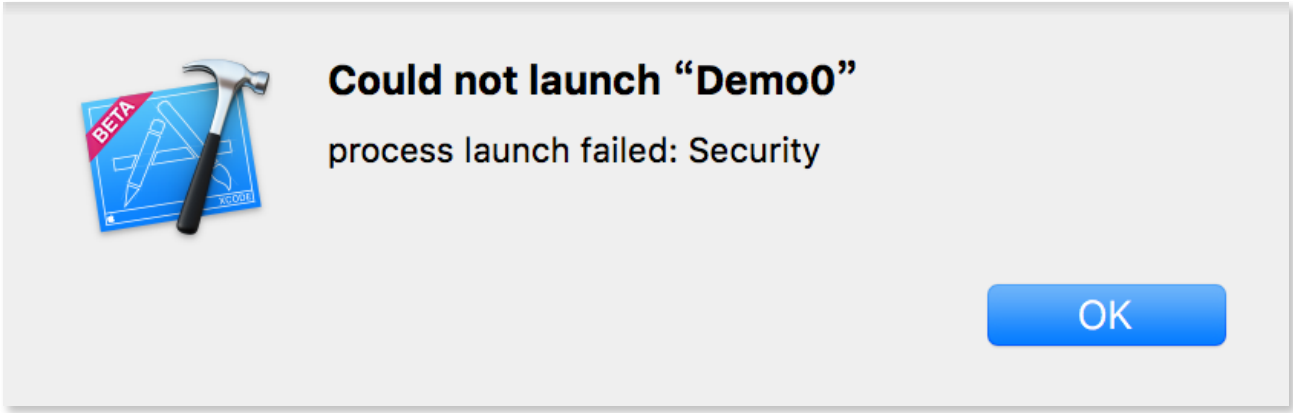
Hello, iOS

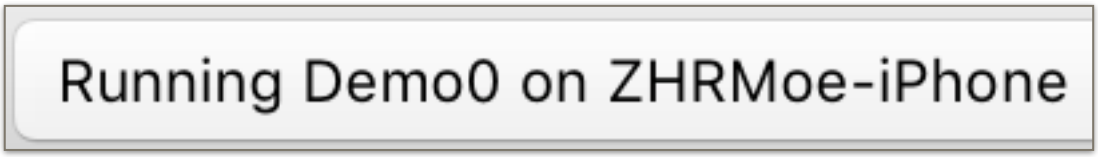
1 动手写代码[1]

上一次我们简单的向空白界面拖拽了一些插件，这虽然是界面设计的基础，但是显然不应该是程序的全部[2]。这一次我们简单的涉及一些关于代码的知识，可能有些同学了解面向对象编程[3]，但是我尽量绕开这个概念说明，在以后应该会介绍这个内容。

这里，首先对上次的内容做两个补充。首先，你需要在Xcode中登陆自己的Apple ID，这样才能够达成编译到手机的第一步。其次，如果你使用了iOS 9，你可能会在第一次编译或者是手机里还没有你的软件时看到这样的提示：



这是因为iOS 9的安全机制发生了一些小变化。这时，你可以打开手机上的设置->通用->描述文件->开发商应用，找到自己的Apple ID并设置为信任。这时再运行就可以顺利的执行测试程序了。



是否还记得上次的程序里，我们使用了UILabel这个插件。一个插件本身应该有若干个用来描述它的内容，比如文本框里面的文字是什么，文字的字体是什么，颜色是什么，更深层的还包括显示多少行等等，我们把这些内容称作属性(property)，对于UILabel到底有什么属性，我们可以打开UILabel.h这个文件看一看。

---

[1] 2015年8月30日，基于OS X 10.11 beta 7， iOS 9 beta 5，Xcode 7 beta 5.

[2] 虽然是向界面拖拽插件，但是这些元素的位置或者是参数都被保留在与storyboard同名的xml文件中，所以它依然是由代码支撑的。

[3] 面向对象编程（Object Oriented Programming，OOP，面向对象程序设计）是一种计算机编程架构。OOP 的一条基本原则是计算机程序是由单个能够起到子程序作用的单元或对象组合而成。OOP达到了软件工程的三个主要目标：重用性、灵活性和扩展性。为了实现整体运算，每个对象都能够接收信息、处理数据和向其它对象发送信息。

```objc
//
//  UILabel.h
//  UIKit
//
//  Copyright (c) 2006-2014 Apple Inc. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <CoreGraphics/CoreGraphics.h>
#import <UIKit/UIView.h>
#import <UIKit/UIStringDrawing.h>
#import <UIKit/UIKitDefines.h>

NS_ASSUME_NONNULL_BEGIN

@class UIColor, UIFont;

NS_CLASS_AVAILABLE_IOS(2_0) @interface UILabel : UIView <NSCoding>

@property(nullable, nonatomic,copy)   NSString           *text;            // default is nil
@property(null_resettable, nonatomic,strong) UIFont      *font;            // default is nil (system font 17 plain)
@property(null_resettable, nonatomic,strong) UIColor     *textColor;       // default is nil (text draws black)
@property(nullable, nonatomic,strong) UIColor            *shadowColor;     // default is nil (no shadow)
@property(nonatomic)        CGSize             shadowOffset;    // default is CGSizeMake(0, -1) -- a top shadow
@property(nonatomic)        NSTextAlignment    textAlignment;   // default is NSTextAlignmentLeft
@property(nonatomic)        NSLineBreakMode    lineBreakMode;   // default is NSLineBreakByTruncatingTail. used for single and multiple lines of text

// the underlying attributed string drawn by the label, if set, the label ignores the properties above.
@property(nullable, nonatomic,copy)   NSAttributedString *attributedText NS_AVAILABLE_IOS(6_0);  // default is nil

// the 'highlight' property is used by subclasses for such things as pressed states. it's useful to make it part of the base class as a user property

@property(nullable, nonatomic,strong)              UIColor *highlightedTextColor; // default is nil
@property(nonatomic,getter=isHighlighted) BOOL     highlighted;          // default is NO

@property(nonatomic,getter=isUserInteractionEnabled) BOOL userInteractionEnabled;  // default is NO
@property(nonatomic,getter=isEnabled)                BOOL enabled;                 // default is YES. changes how the label is drawn

// this determines the number of lines to draw and what to do when sizeToFit is called. default value is 1 (single line). A value of 0 means no limit
// if the height of the text reaches the # of lines or the height of the view is less than the # of lines allowed, the text will be
// truncated using the line break mode.

@property(nonatomic) NSInteger numberOfLines;

// these next 3 property allow the label to be autosized to fit a certain width by scaling the font size(s) by a scaling factor >= the minimum scaling factor
// and to specify how the text baseline moves when it needs to shrink the font.
```

如果你看不懂，也没什么关系，上图中黑色的部分就是属性的名字，一般来说，属性的名字是直接描述了属性的含义的[4]。

这一节，我们要做一个把用户输入的文字用UILabel展示出来的程序。和上次的步骤相似，我们建立Demo1这个项目。

在语言上我们选择Objective-C，这样可以更加直观的看到属性这个概念。

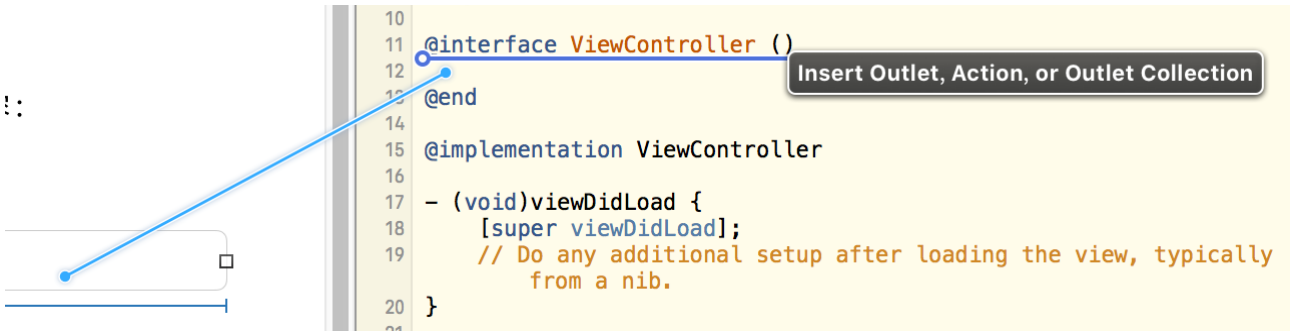
然后，简单的拖拽一些文本框进来，填写内容，并设置约束。这里的约束你可以自由添加，多尝试一些约束也是可以的。

下面，我们加入输入框和一个按钮，组件的名字分别是UITextField和UIButton，完成后效果如图。

---

[4] 我们强调这一点是因为工程中经常出现自己设计一个模块并描述属性的情况，如果属性的名字起的不够直观，会直接影响自己或者是合作的程序员维护代码的效率和准确度。另，截图中双斜杠后面的内容是注释，一般用来描述上述代码的作用和含义，在实际项目中，也应该多写注释，让自己和其他开发者都能够看懂代码。

问题是，我们怎么让这个界面的两个不同的组件产生交互呢？这里我们可以理解为，**这个界面是一个大组件，而输入框和下方的文本框都是它的属性**，这样我们就可以用属性的角度去思考它们了。同时，点击按钮这个动作触发后，下方的文本框内容才会改变。所以这时候我们开始用代码控制这个过程。



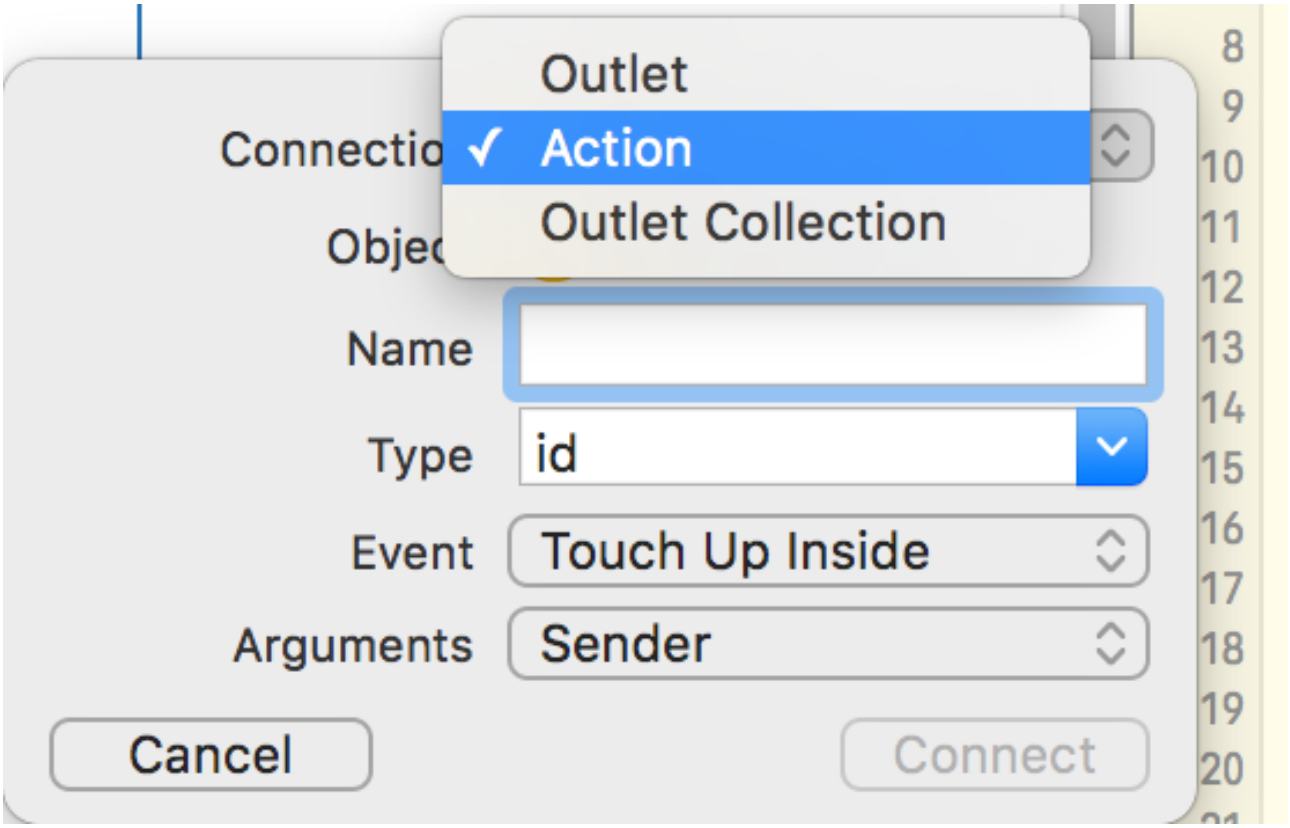
我们在右上角选择双窗口，也就是两个圆的图标。选中输入框，按住ctrl键把它拖拽进代码里。

在弹出的窗口里的Name栏填写这个输入框的名字，比如textField。注意，这里的名字采取驼峰式的命名方法，即**除了第一个字母小写外，之后每遇到一个单词就大写其首字母，但是不加空格**。

然后用同样的方法添加下方的文本框，代码如图：

```
9  #import "ViewController.h"
10
11 @interface ViewController ()
12
13 @property (weak, nonatomic) IBOutlet UITextField *textField;
14 @property (weak, nonatomic) IBOutlet UILabel *showLabel;
15
16 @end
17
```

左侧的圆点表示这行代码和storyboard中的组件产生了链接。下面我们设计点按按钮的动作，方法相同，但是连接方式要选择Action：



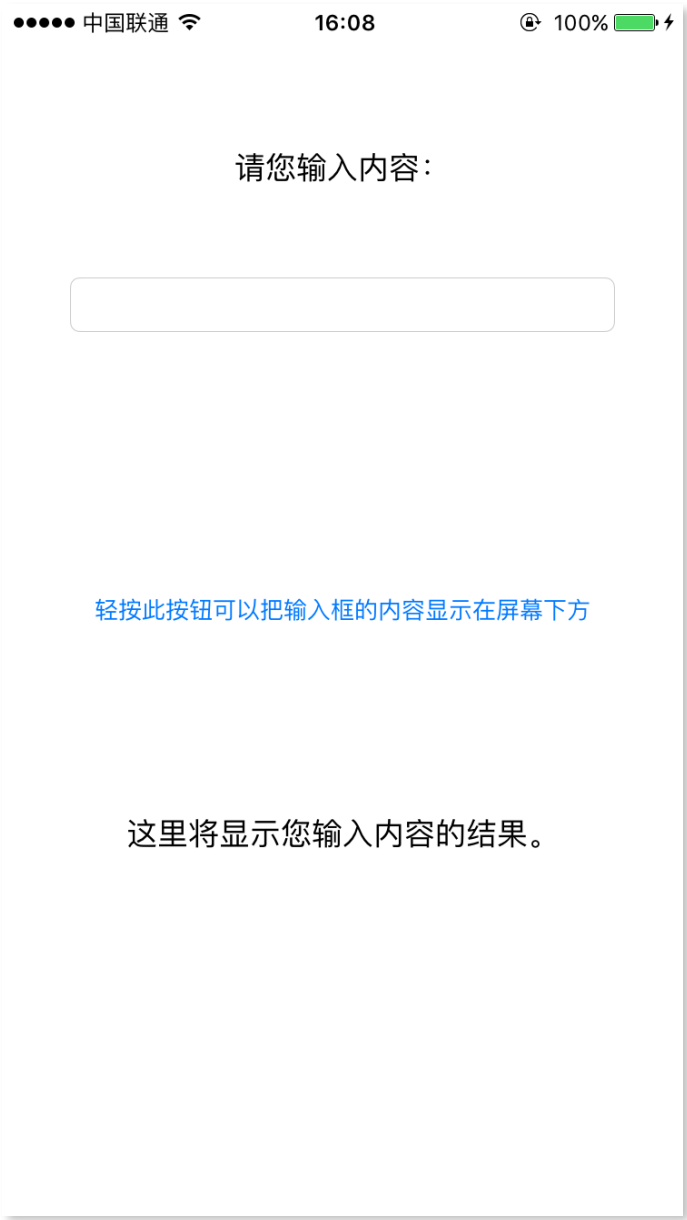

```
13 @property (weak, nonatomic) IBOutlet UITextField *textField;
14 @property (weak, nonatomic) IBOutlet UILabel *showLabel;
15 - (IBAction)transformButtonDidPressed:(id)sender;
```

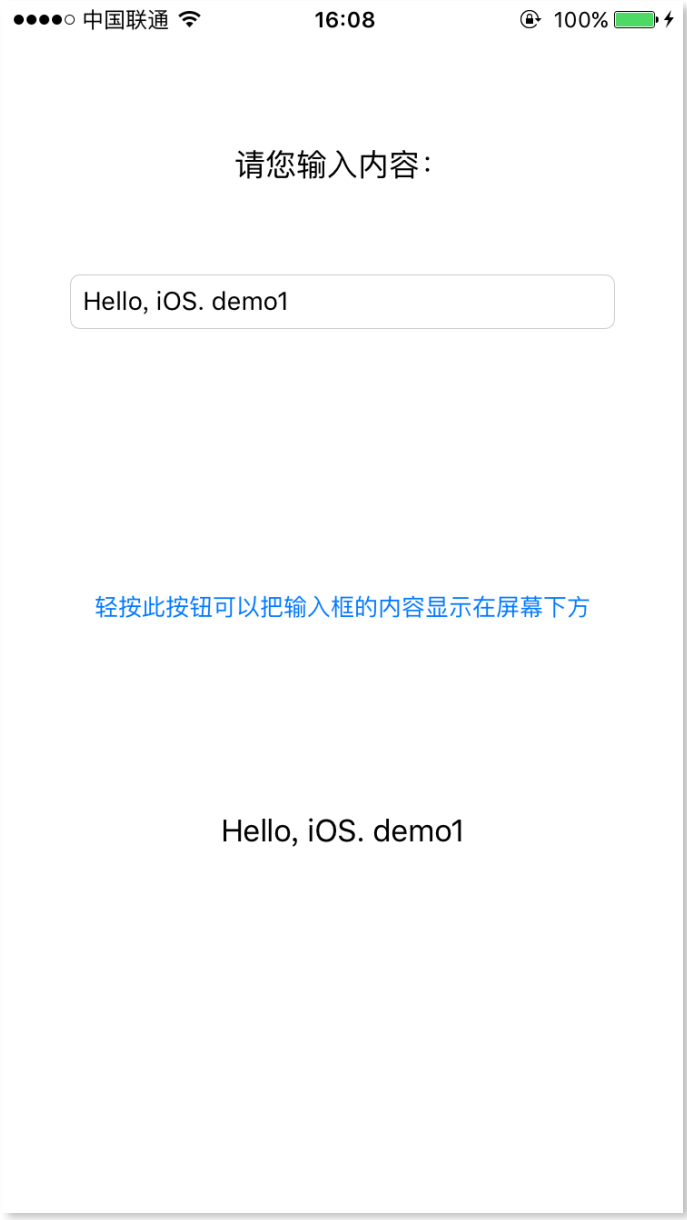
这样，需要连接的部分就到这里了。我们切换回单界面，开始考虑怎么描述这个动作。根据我们之间的想法，点击按钮的时候，我们需要把输入框的文字内容写进下方的文本框里。所以，我们看到文件中下方有一个部分，和我们刚才给按钮起的名字非常相似，这行代码将写在这里。

```
31 - (IBAction)transformButtonDidPressed:(id)sender {
32     self.showLabel.text = self.textField.text;
33 }
```

我们来解释一下这行代码的含义。self指的是这个界面本身，由于我们考虑输入框和文本框都是它的属性，所以可以用点这个符号连接起来表示从属关系。由于输入框和文本框都有文本这个属性，所以可以用点符号连接text。需要注意这个等号两侧的内容不可以交换位置，因为在程序中等号表达赋值操作，是将后面的值传给前面。

写好之后我们测试一下功能。运行如图：





这一节，我们终于把工程和写代码联系在了一起，虽然我们绕过了非常多复杂繁琐的概念。我的目的其实是要简单地介绍iOS开发的步骤，并且让大家爱上iOS的开发。

第1节 **EOF**[5]

[5] End Of File的简略写法，在程序中常用于检查文件是否已经读完。