实验 5

创建简单注册表单

实验结构

- 使用静态表创建用户输入表单

实验目标

本实验结束后，你将能够：

- 在 Storyboard 中使用静态表创建简单注册表单

导论

你学习了如何在 Xcode 的 terminal 上使用 Xcode 来执行 Objective-C 程序。本节中，你将在 Xcode 中创建一个注册表单，来获得申请人详细信息。

实验：创建简单注册表单

注册表单将有一些静态表，用于填入个人细节。

背景

应用是在多种数据的基础上建立起来的。除了处理器上运行的原始二进制指令之外，应用还包括声音文件、图形文件、屏幕、背景、按钮、游戏令牌、字体文件和其它支持的信息。创建应用时，Xcode 会将这些元素合并到一个产品文件中。大部分构建过程是自动的。

本实验中，你将在使用 Xcode 创建一个注册表单。为此，你需要在 Xcode 上使用静态表创建用户输入表单应用。

实验准备

开始实验之前，你需要有：

- 安装有 Xcode 的 Mac OS
- 苹果开发者帐户

实验：推荐解决方案

1. 启动 Xcode。在 Welcome to Xcode 界面上（参见图 1），选择 Create a new Xcode project 选项，开启新项目创建。

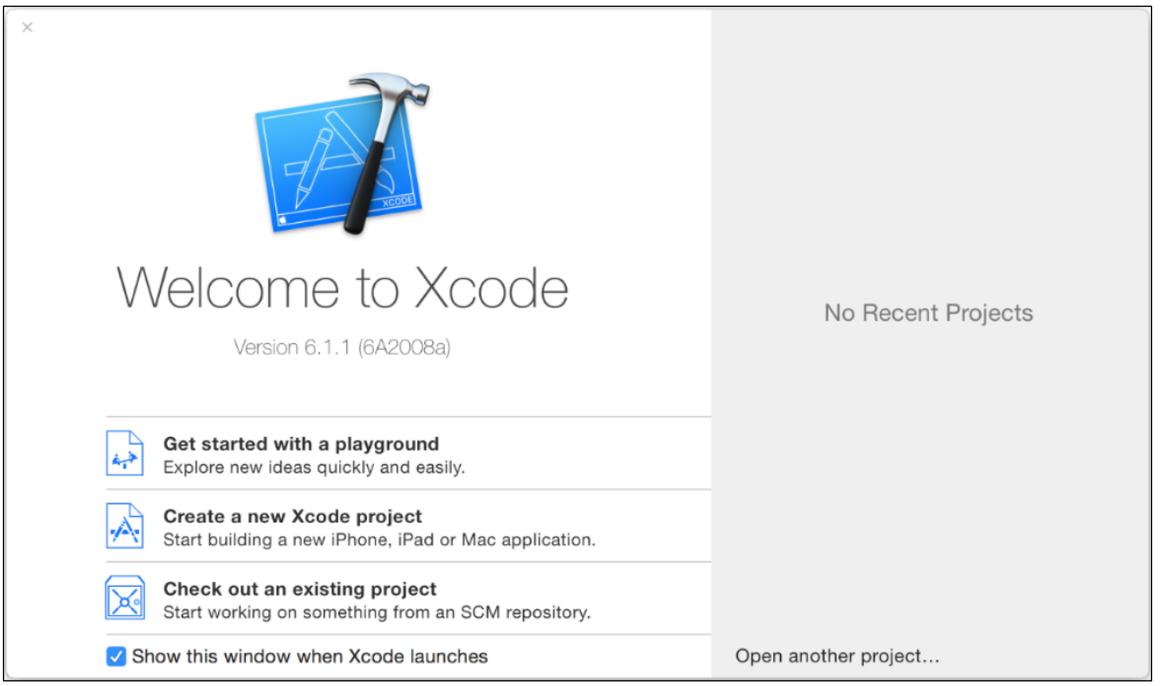

图 1: 选择 Create a New Xcode Project

2. 为应用选择 Single View Application 模板，并点 Next，如图 2 所示：

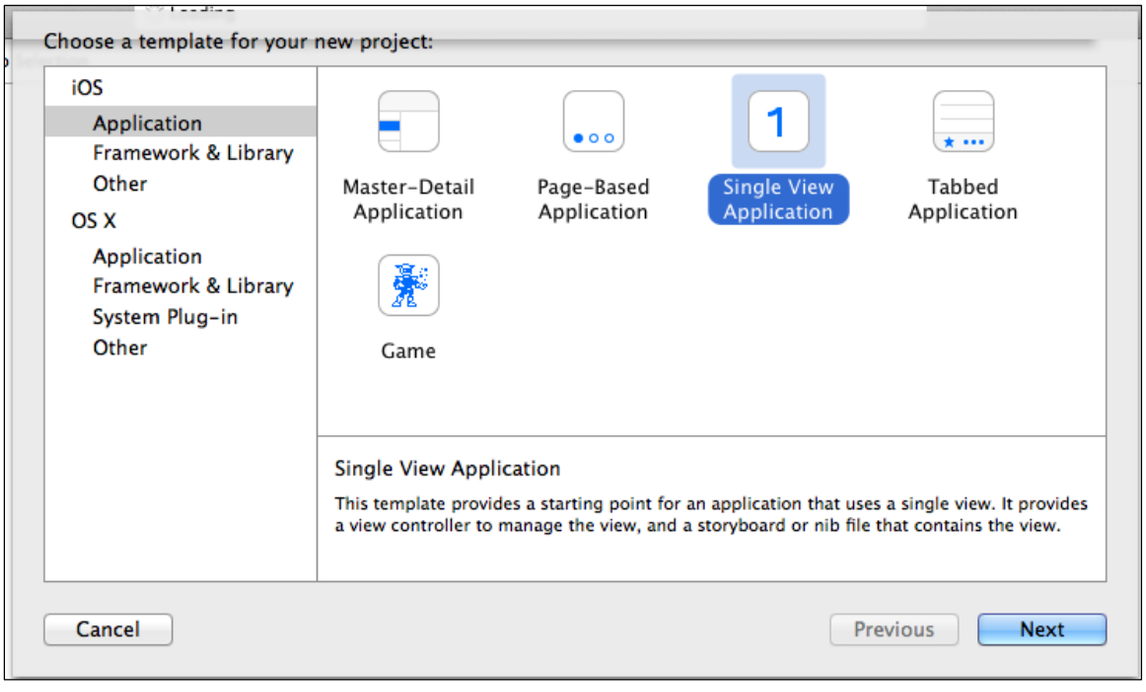

图 2: 选择 Single View Application 模板

3. 将应用命名为 **RegistrationForm**，按照图 3 填入相关选项，然后点 **Next**。

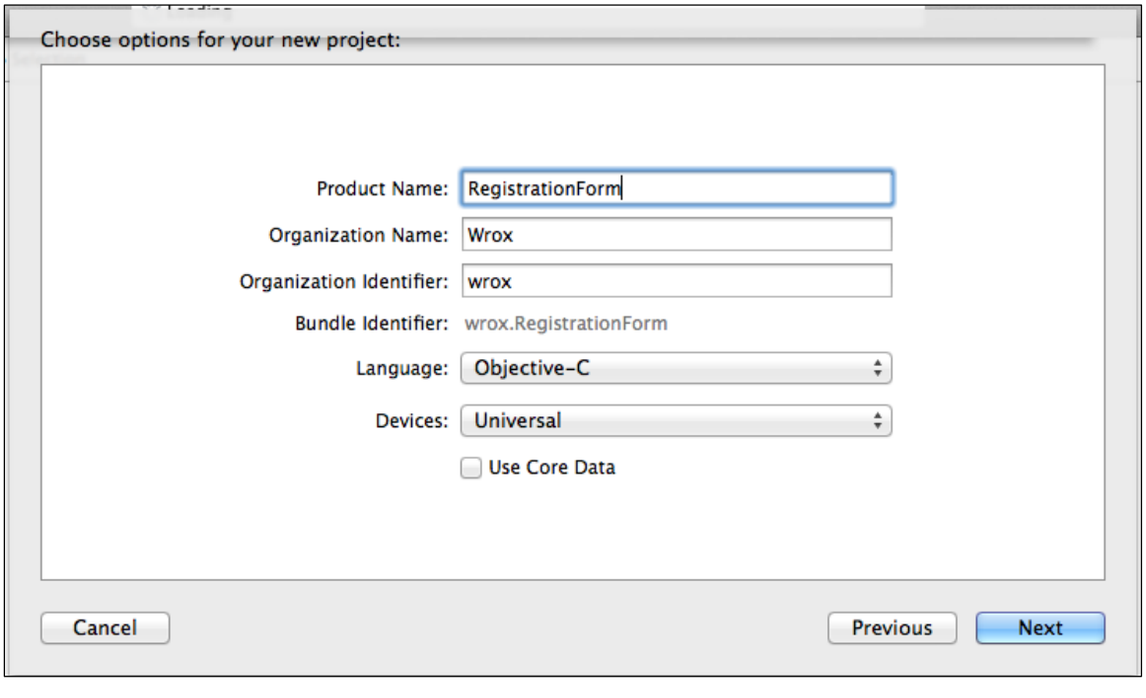

图 3: 将应用命名为 RegistrationForm

4. 下一个界面上（参见图 4），选择一个位置来保存项目，并点 **Create**。

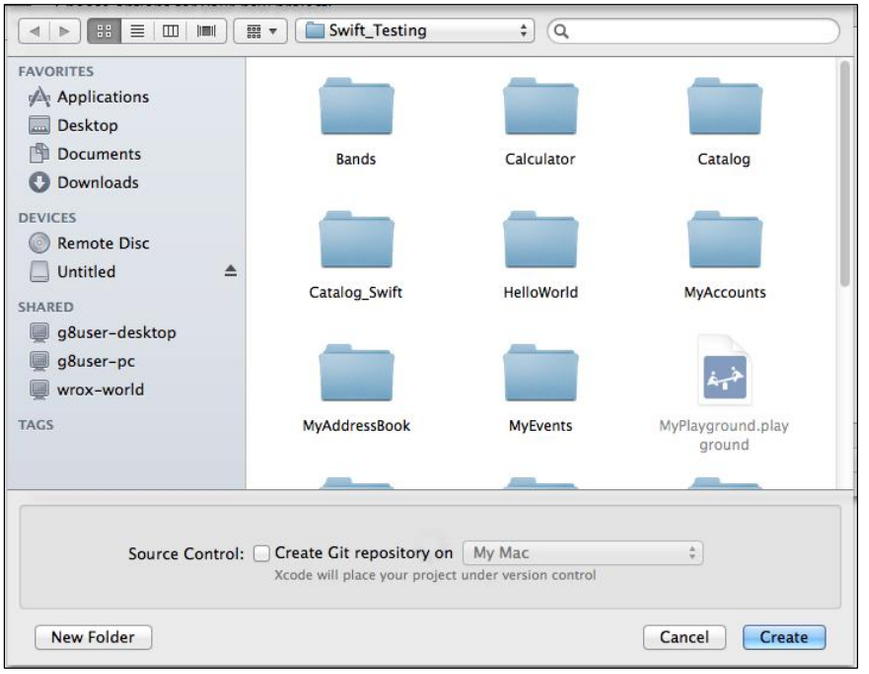

图 4: 选择项目的保存位置

5. **RegistrationForm** 项目文件会在 Xcode 中显示出来，如图 5 所示。这是 Xcode 中 Objective-C 的一般格式。

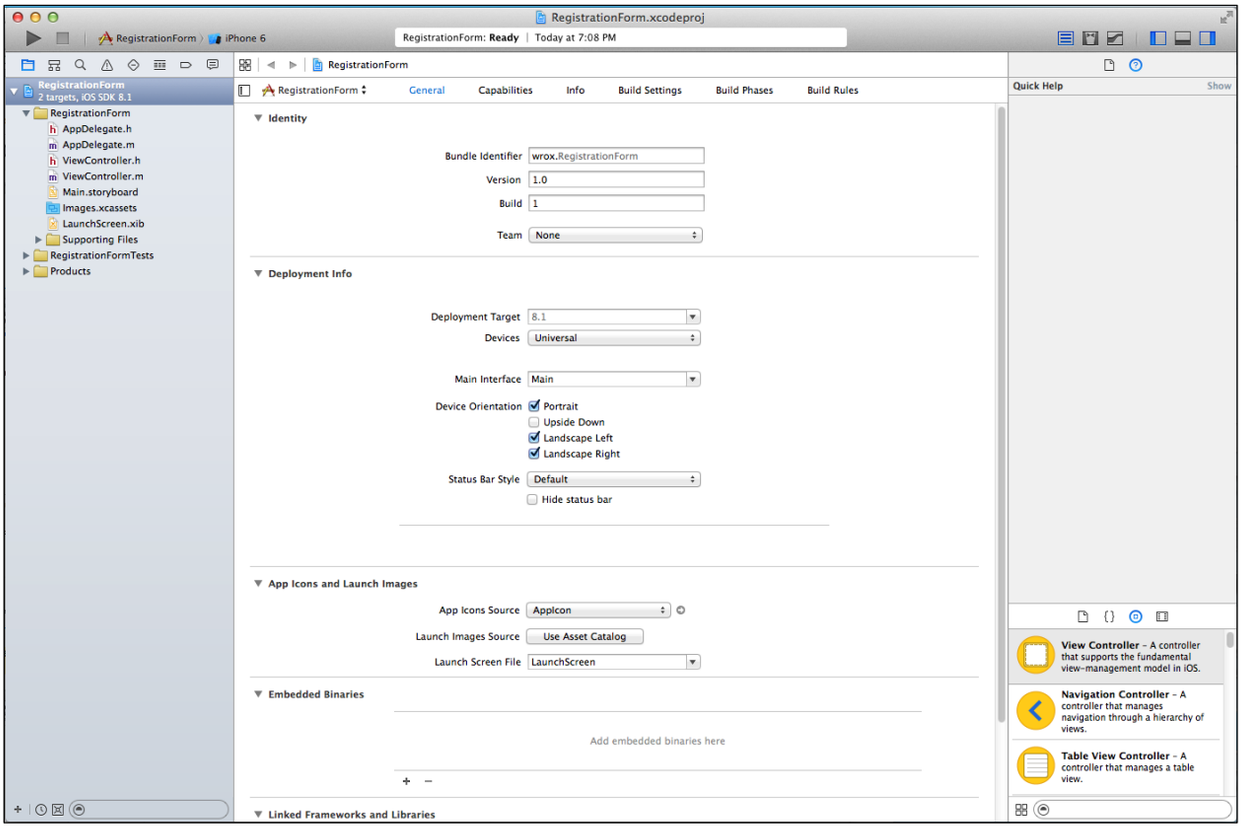

图 5: Xcode 中的 RegistrationForm 项目文件

6. 在 Xcode 窗口中，导航面板位于左侧（参见图 6），项目编辑器在中间（参见图 7），实用工具面板在右侧（参见图 8）。

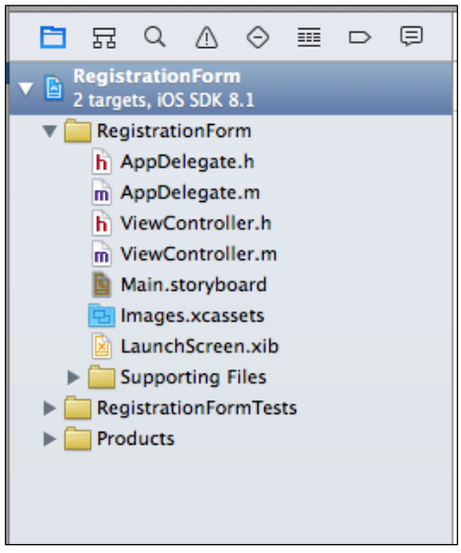

图 6: 左侧导航面板

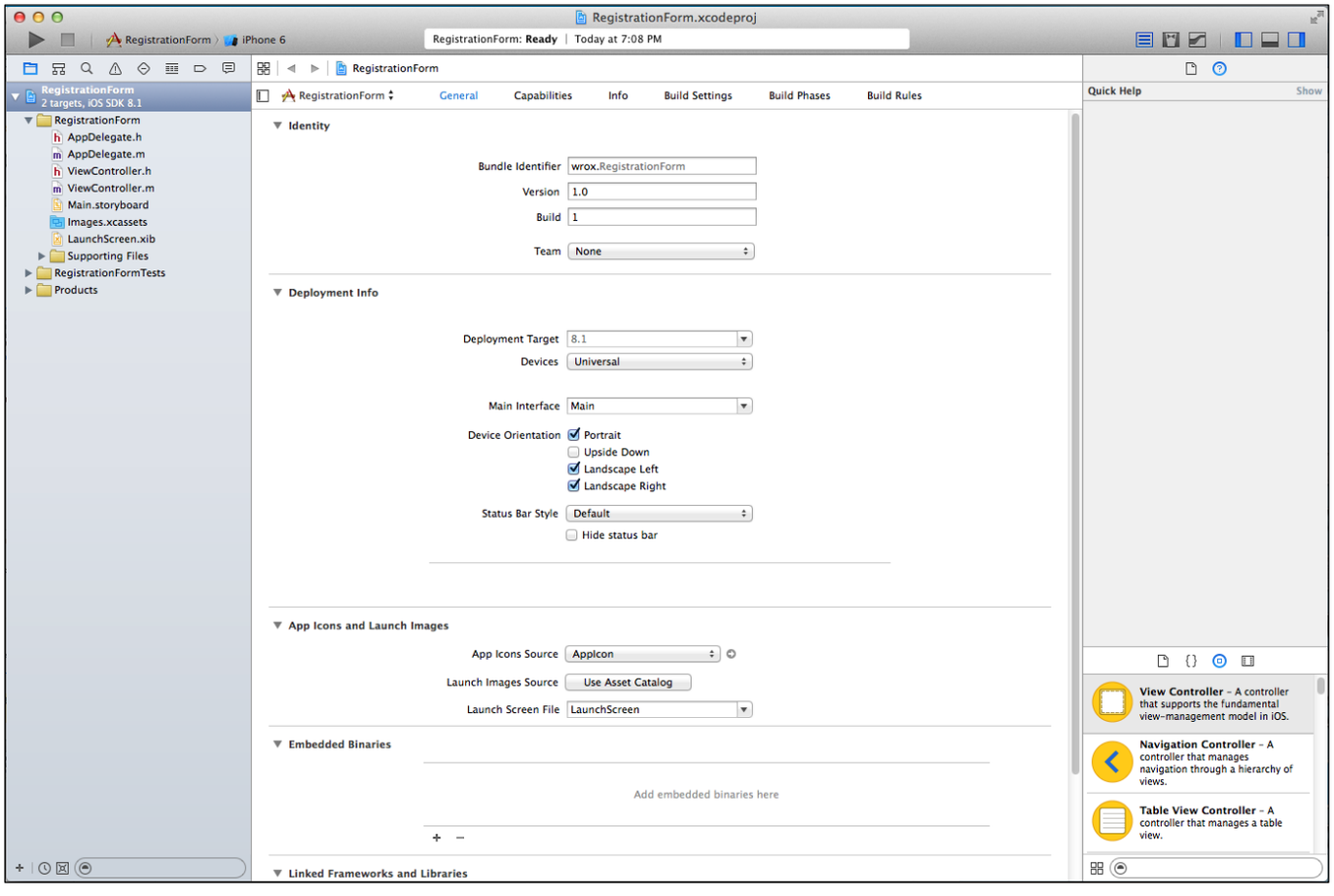

图 7: 中间的项目编辑器

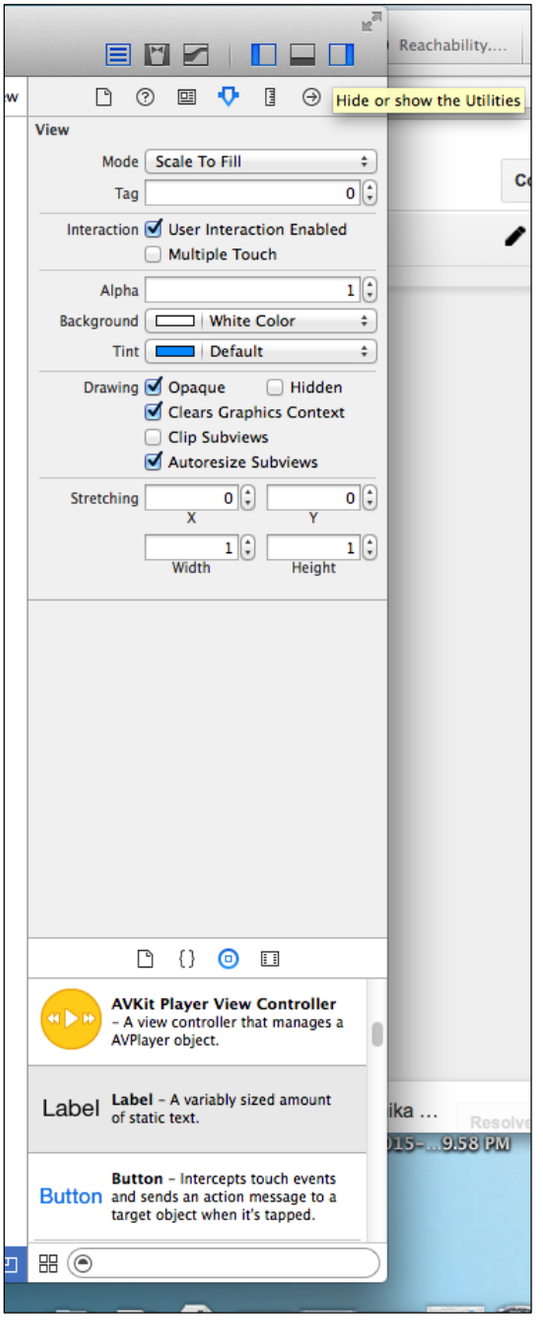

图 8: 右侧实用工具面板

7. 下面，添加一个 `UILabel` 到场景中。从导航面板选择 `Main.storyboard` 文件（参见图 9）。在实用工具面板底部，选择立方体图标表示的 Objects 选项卡。图 10 显示了实用工具面板的特写，其中 Objects 选项卡由蓝色高亮表示。

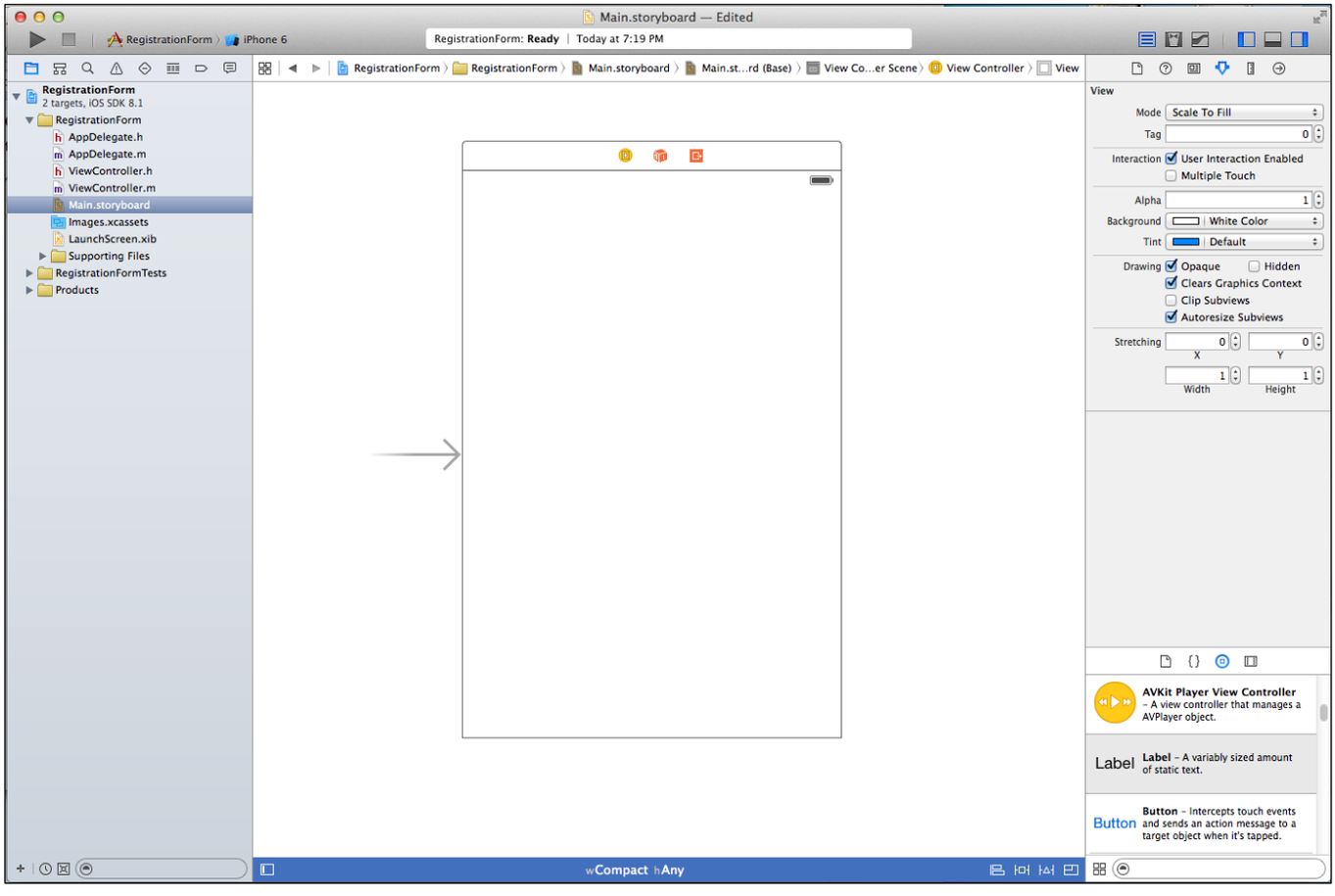

图 9: 在导航面板中选择 `Main.storyboard`

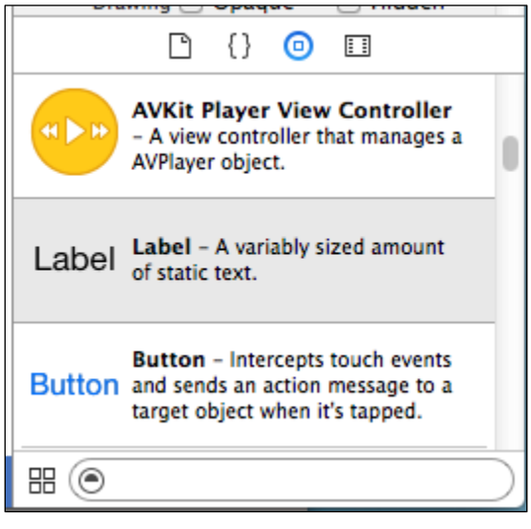

图 10: 选择 Objects 选项卡（蓝色高亮显示的立方体图标）

8. 点实用工具面板的 **Label** 对象。鼠标左键按住并将它拖到项目的用户界面，如图 11 所示：

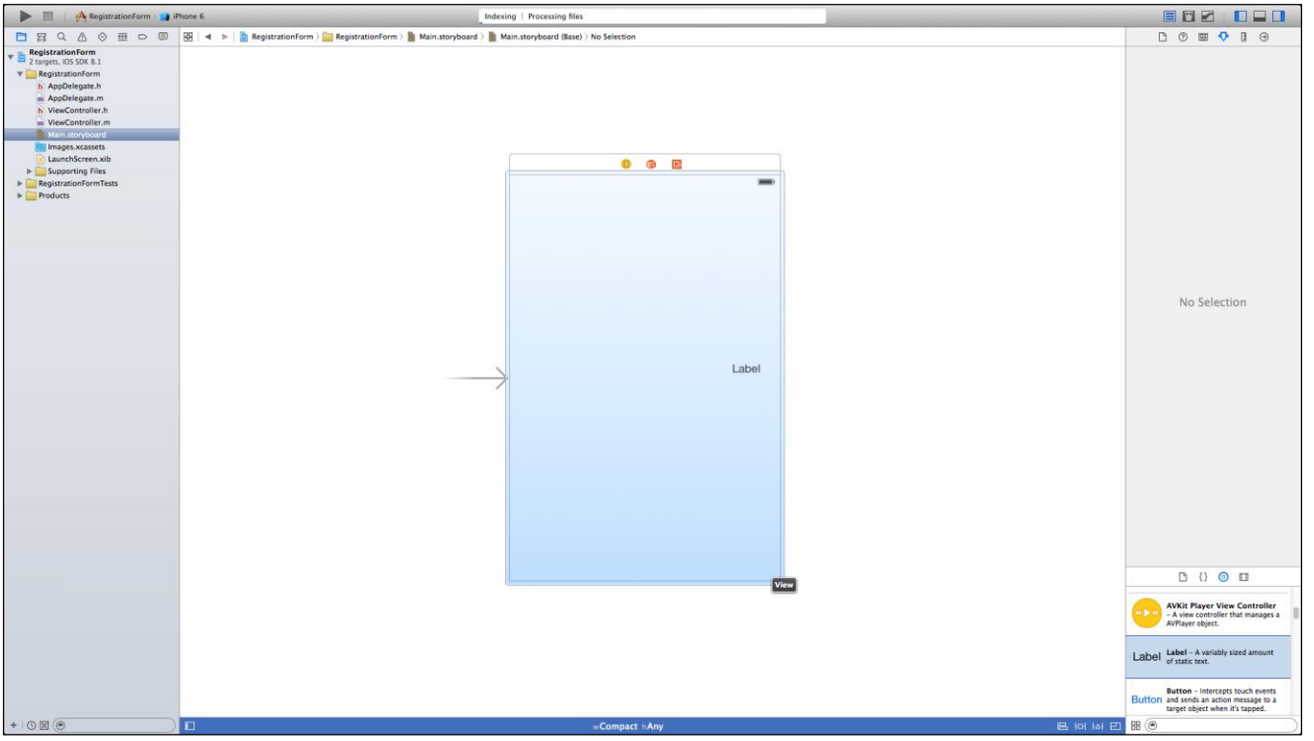

图 11: 将标签对象左键按住，从实用工具面板拖放到项目用户界面

9. 之后，你可以使用实用工具面板中的工具更改标签的文本、尺寸、对齐方式、字体和颜色。例如，你可以将标签文本改成注册表单 **Registration Form**，如图 12 所示：

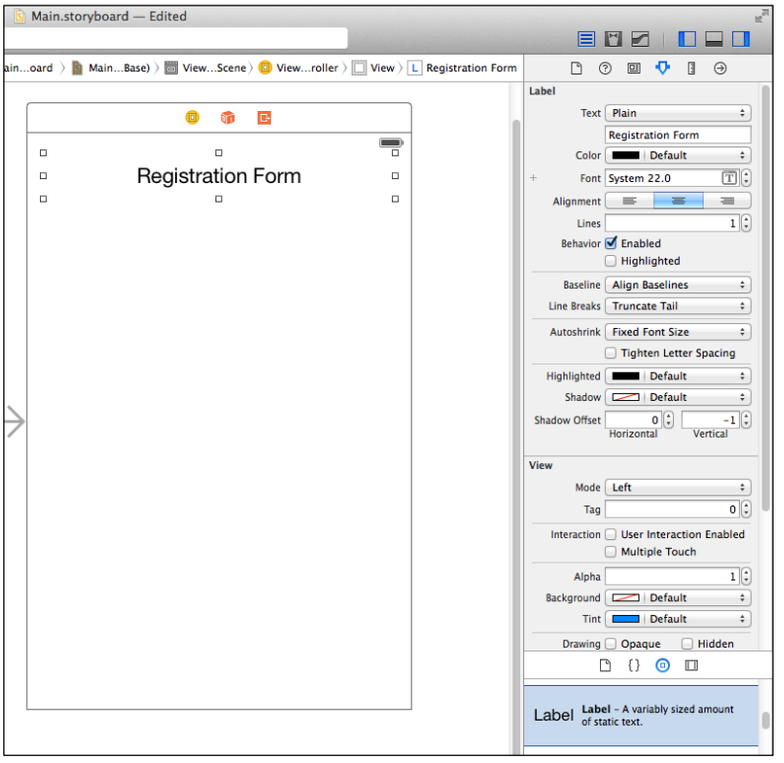

图 12: 使用实用工具面板选项编辑标签

10. 下面，点击按住 **Text Field** 对象并将其从实用工具面板拖到项目用户界面。使用实用工具面板工具调整 **Text Field** 对象大小，如图 13 所示：

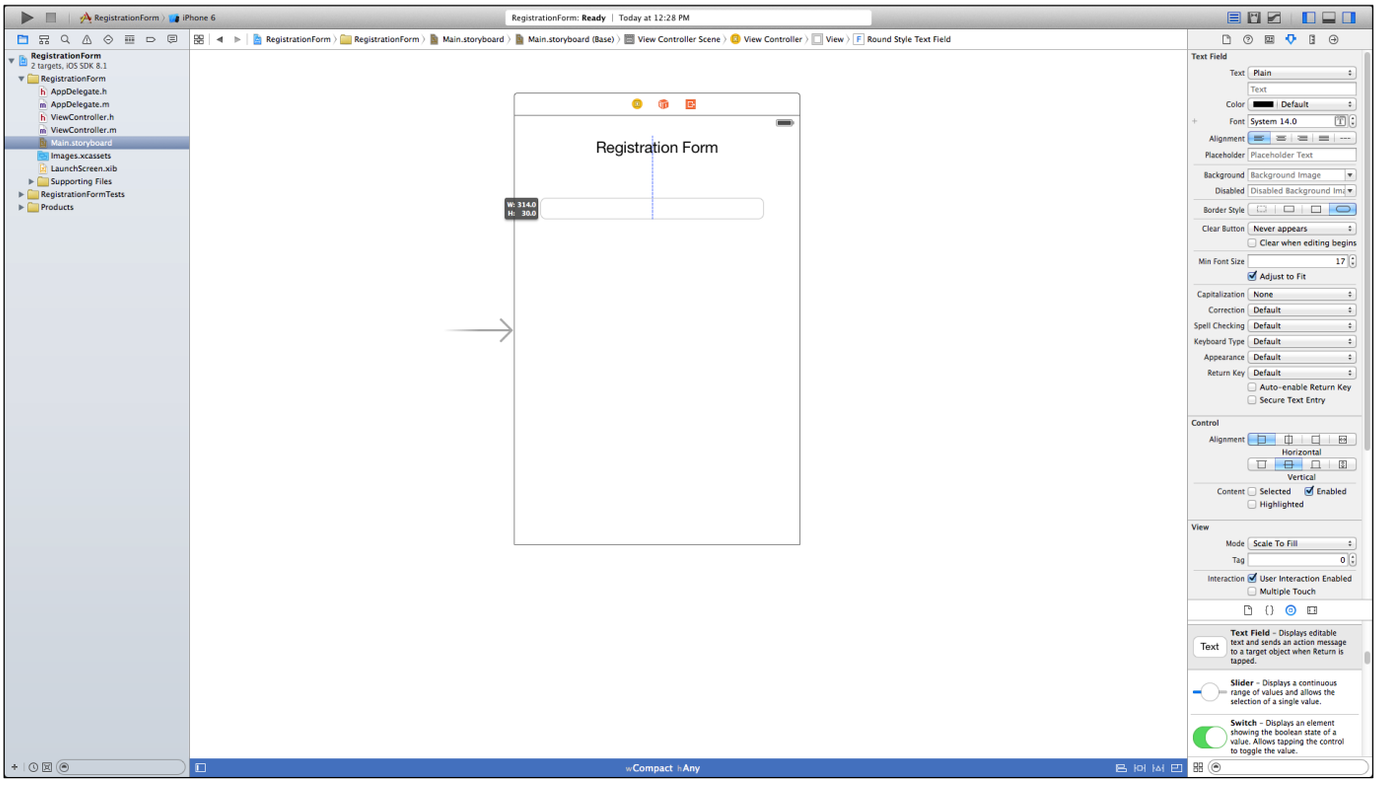

图 13: 将 Text Field 从实用工具面板拖到项目用户界面

11. 下面点选属性检查器图标，如图 14 所示。在 Text Field 中提供一个占位符，**Enter Your Name**，并指定选项，如图 15 所示：

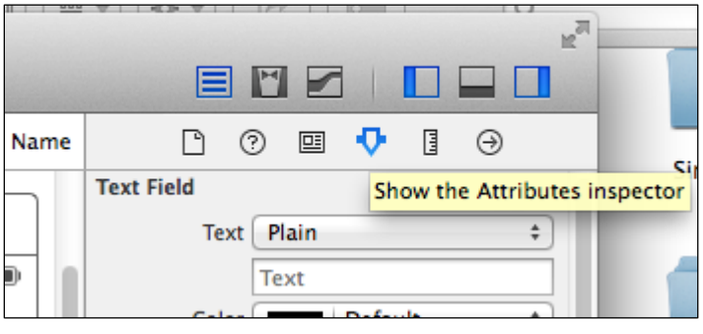

图 14: 点属性检查器，在 Text Field 中添加占位符

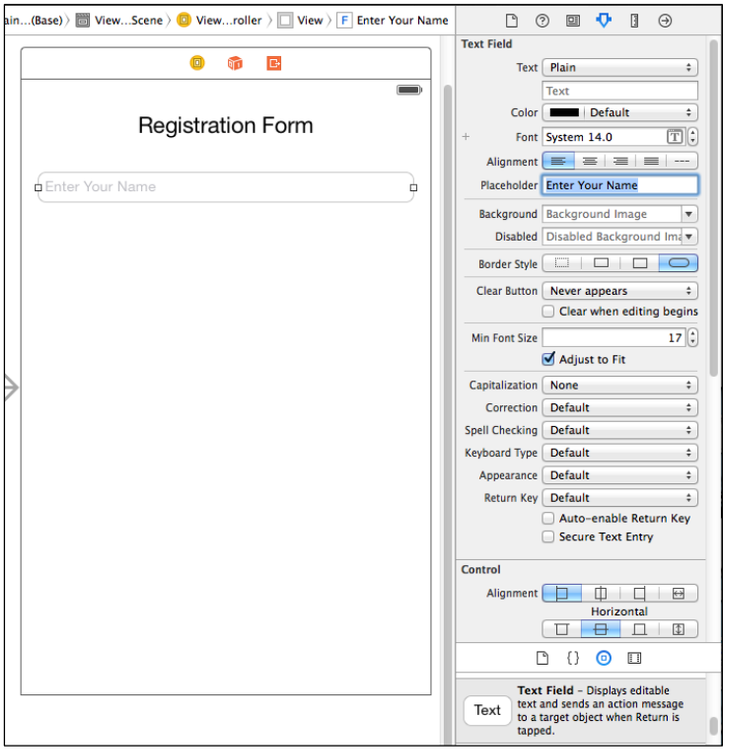

图 15: 指定占位符名称和其它选项

12. 你还可以类似添加五个 Text Field 到项目用户界面，并提供占位符用于电子邮箱、电话号码、公司名、员工 ID、地址，如图 16 所示：

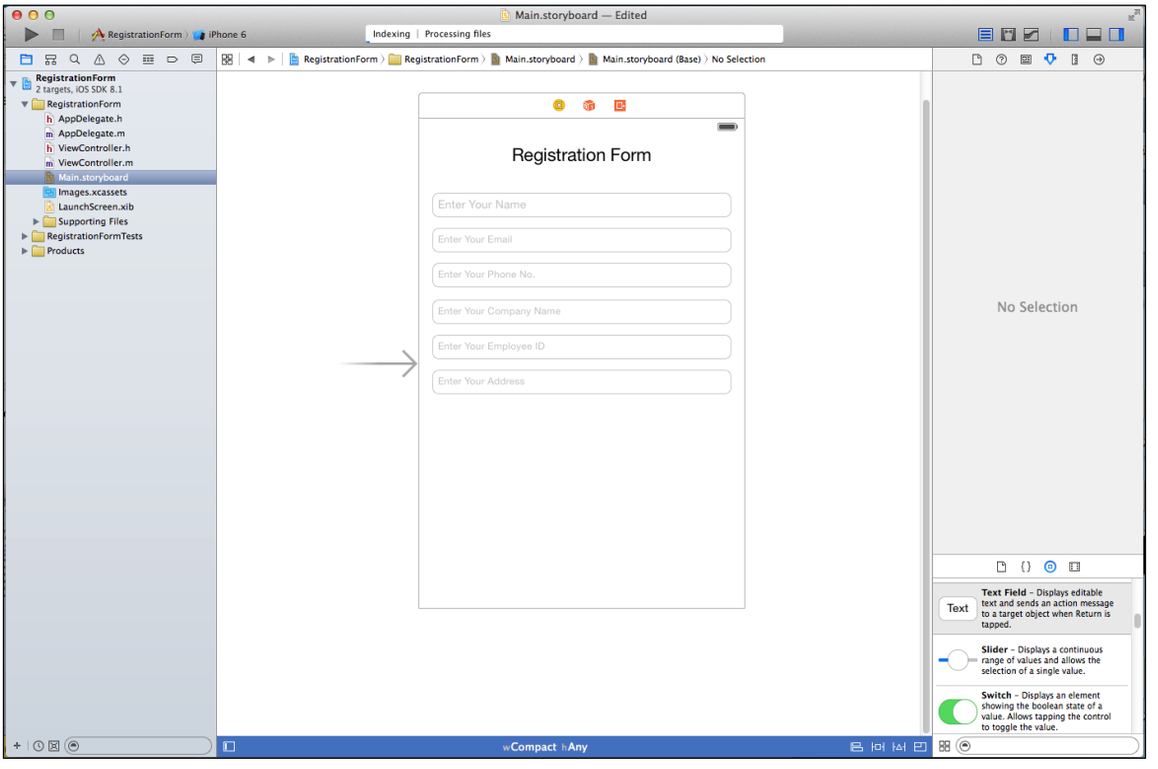

图 16: 添加带有占位符的 Text Field，用于电子邮箱、电话号码、公司名、员工 ID、地址

13. 下面添加一个按钮用于提交注册表单。为此，点击按住 **Button** 对象并将其从实用工具面板拖到项目用户界面，如图 17 所示：

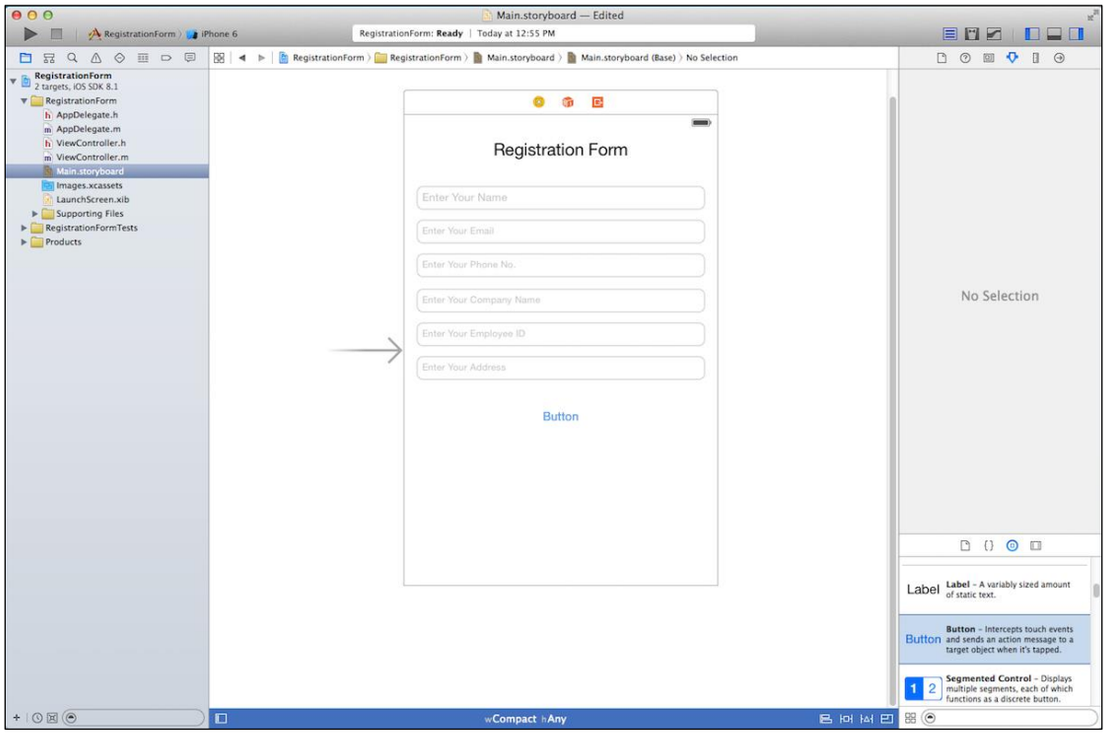

图 17: 从实用工具面板添加按钮到项目用户界面

14. 点项目用户界面中添加的 **Button**，使用属性检查器中的选项将其改名为 **Submit**，如图 18 所示：

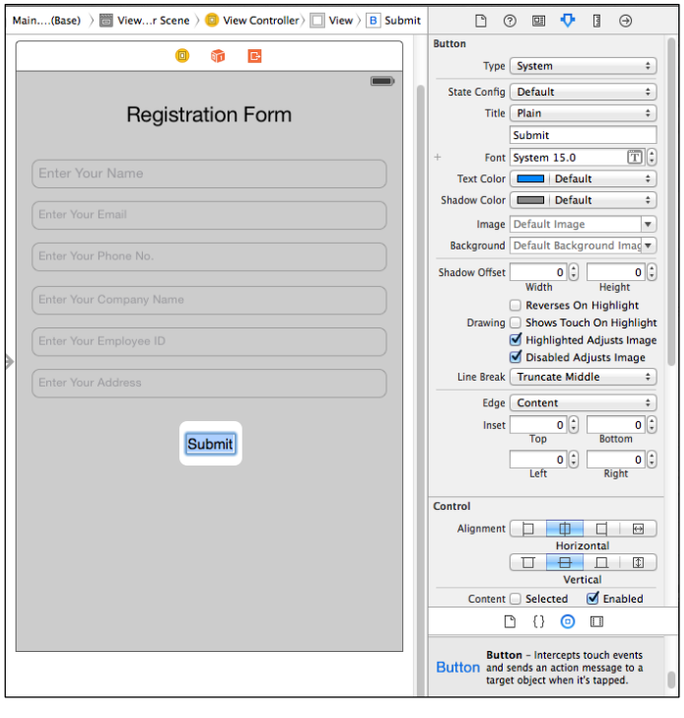

图 18: 将添加的按钮命名为 Submit

15. 往注册表单中添加了一个标签、六个文本框和一个提交按钮后，你需要将文本框和按钮同 `ViewController.h` 文件中的 outlet 和 action 关联起来。为此，你需要通过点击辅助编辑器来打开 `ViewController.h` 文件，如图 19 所示：

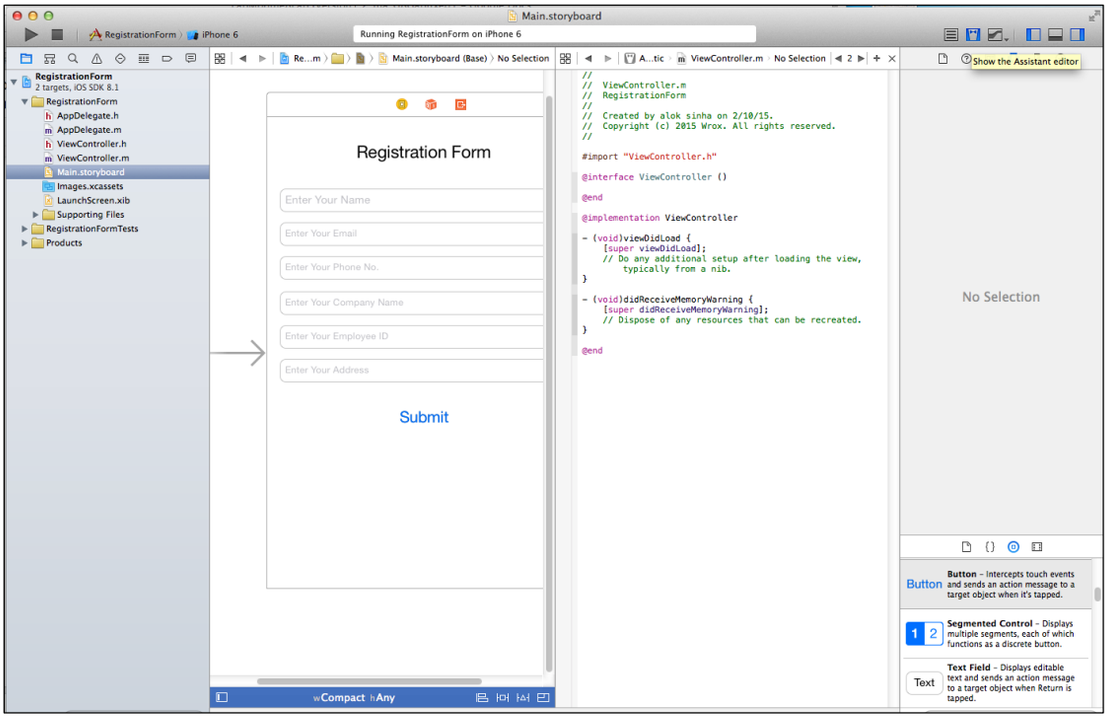

图 19: 点辅助编辑器来打开 `ViewController.h` 文件

16. 下面右键按住，并将每一个 Text Field 拖放到 `ViewController.h` 文件中。这会为 Text Field 生成一个指针变量，如图 20 所示：

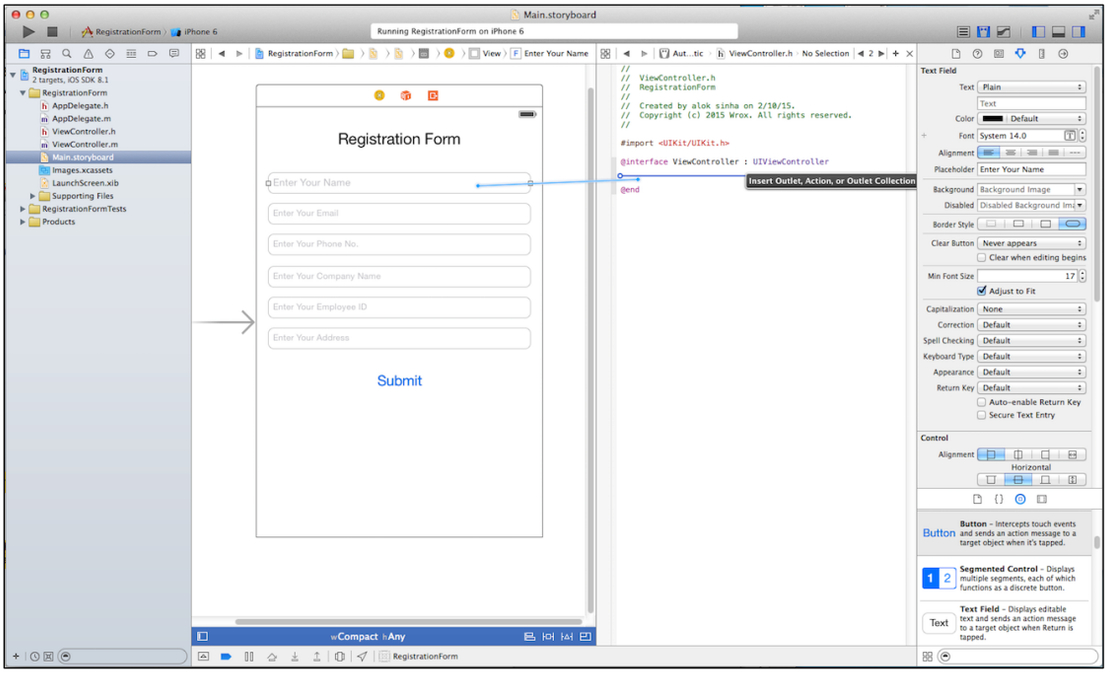

图 20: 右键按住，并将每一个文本框拖放到 `ViewController.h` 文件中

17. 将 Text Field 拖放到 `ViewController.h` 文件中以后，你会看到如图 21 所示的弹窗。弹窗中，在 **Name** 文本框中键入 **EnterName**，然后点 **Connect** 按钮。

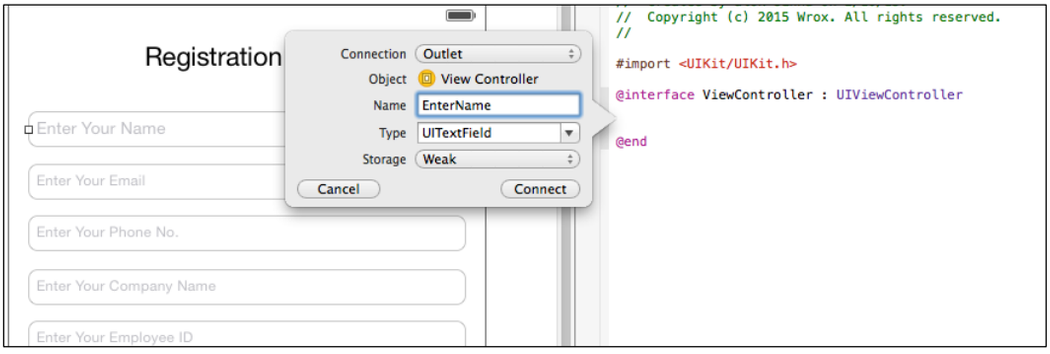

图 21: 通过弹窗在 `ViewController.h` 中关联文本框

18. 将剩余所有文本框关联到 `ViewController.h` 文件中也是类似操作，如图 22 所示：

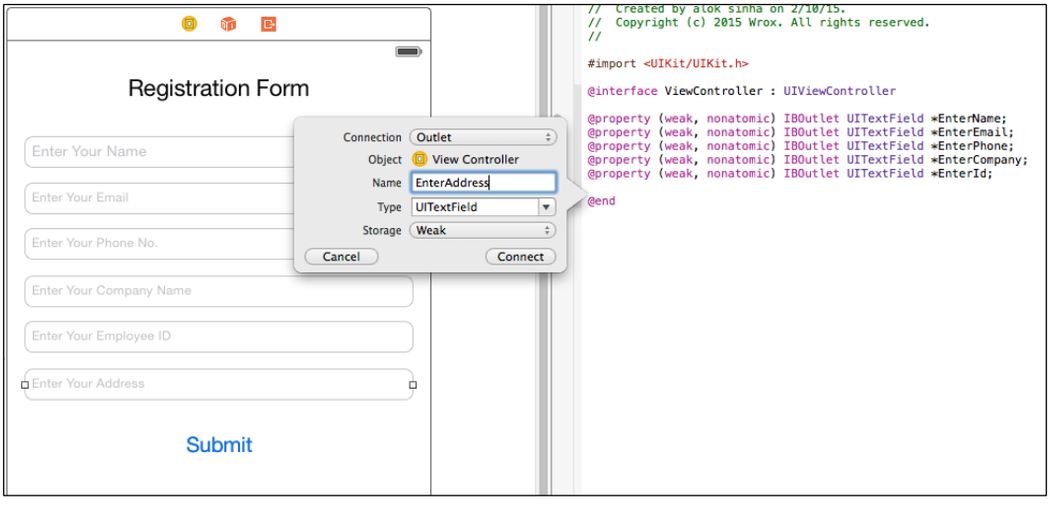

图 22: 在 `ViewController.h` 中关联所有文本框

19. 下一步，以相同方式关联 **Submit** 按钮。右键按住，并将 **Submit** 按钮拖放到 `ViewController.h` 文件中。在出现的弹窗中，**Name** 文本框中键入 **Submit**，然后点 **Connect** 按钮，如图 23 所示：

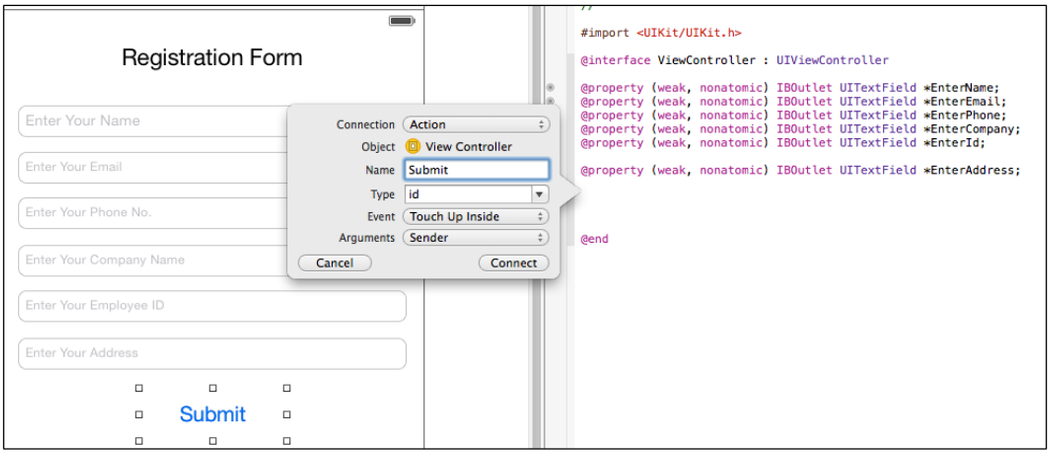

图 23: 在 `ViewController.h` 中关联提交按钮

20. 下面，将这些 Text Field 作为 `TouchButton` 添加。这将让用户在触摸文本框后能够输入文本。为此，右键按住，并将 Text Field 拖放到 `ViewController.h` 文件中。在出现的弹窗中，**Connection** 列表中选择 **Action**，**Name** 文本框中键入 `TouchButton`，然后点 **Connect** 按钮，如图 24 和 25 所示：

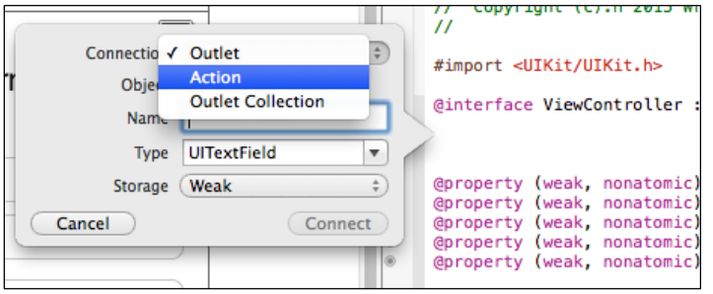

图 24: 在 `ViewController.h` 中添加 Action 按钮

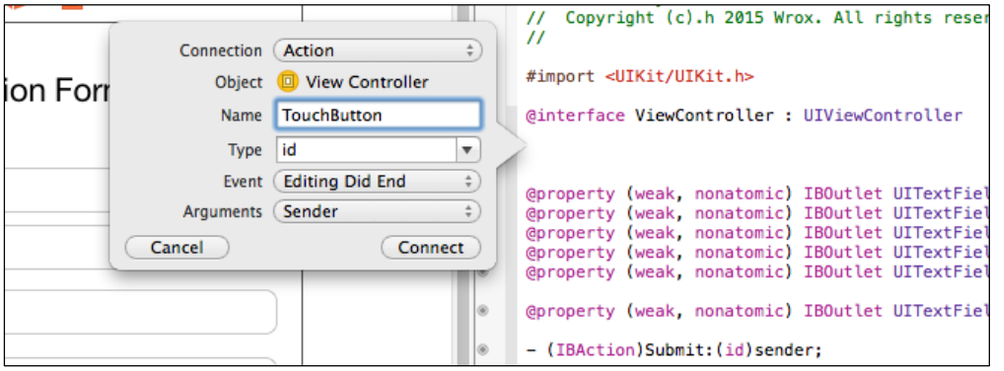

图 25: 将 Action 的 Text Field 命名为 `TouchButton`

21. 右键按住，并将每一个 TextField 拖放到之前创建的 `(IBAction)TouchButton`，如图 26 所示：

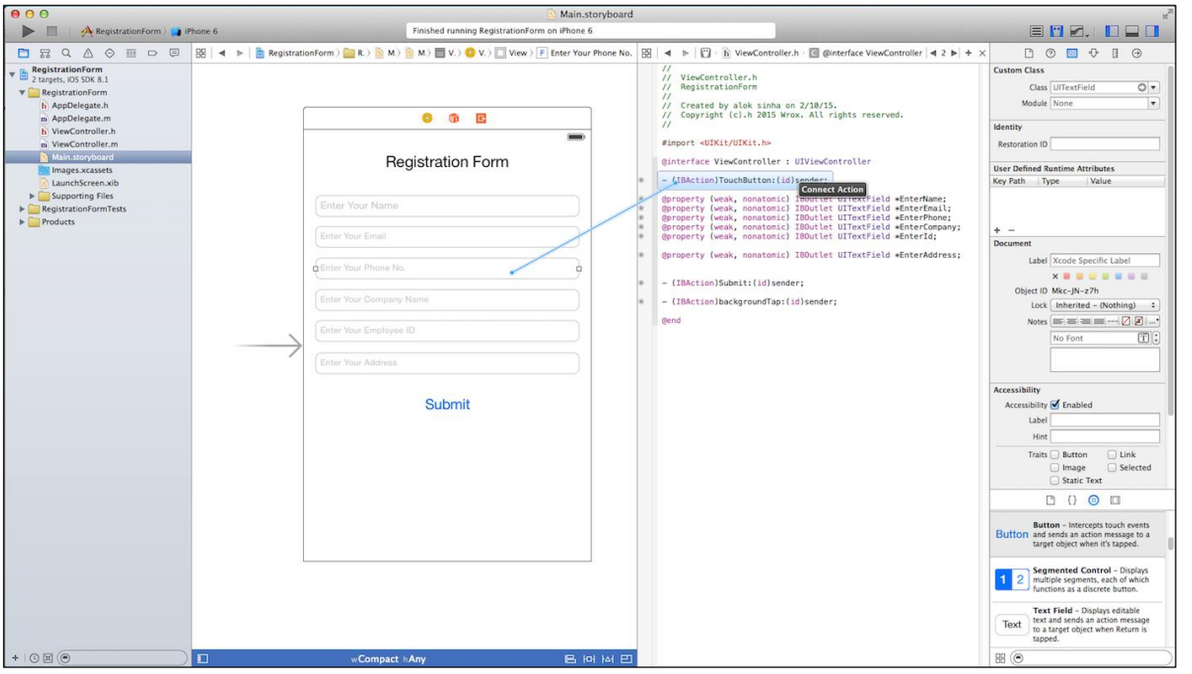

图 26: 拖放 Text Field 到 `(IBAction)TouchButton`

22. 下面，在导航面板中选择 `ViewController.m` 文件并添加相应代码，如图 27 所示：

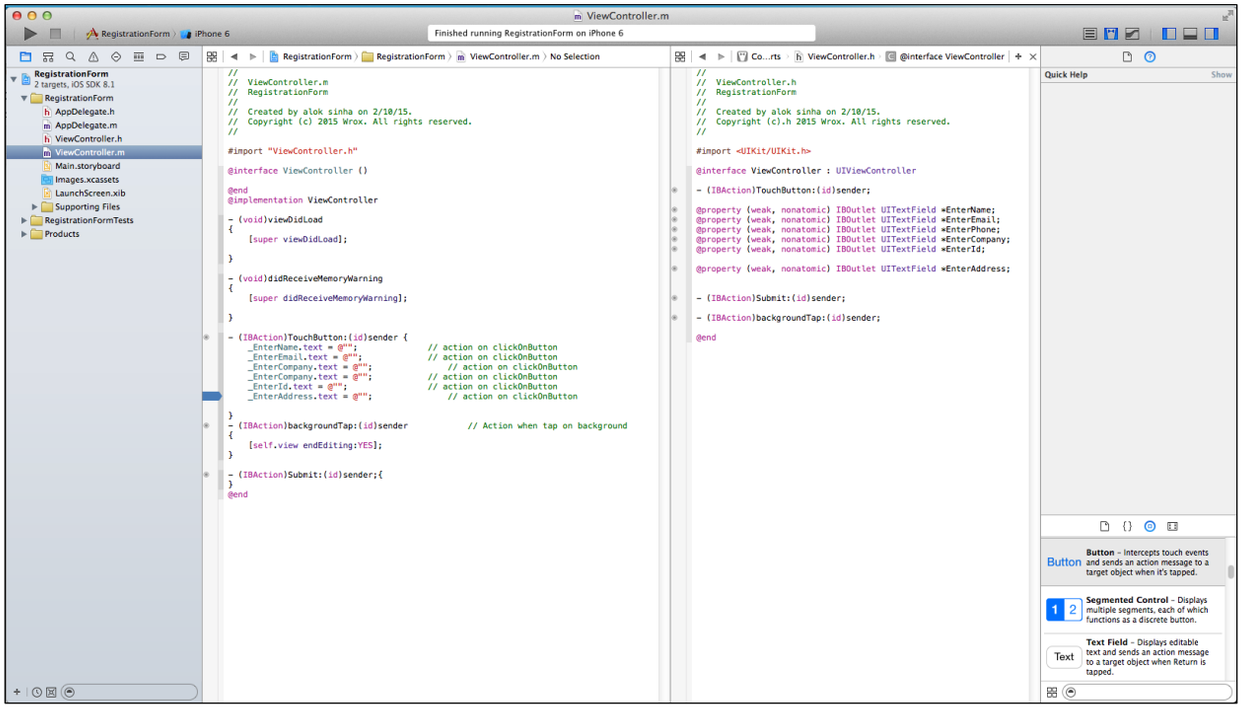

图 27: 在 `ViewController.m` 文件中添加代码

23. 最后，从 **Product** 菜单选择 **Run** 选项来运行应用，如图 28 所示。你会看到注册表单应用安装到了 iOS 模拟器上，如图 29 所示。开启项目后，应用输出结果如图 30 所示。

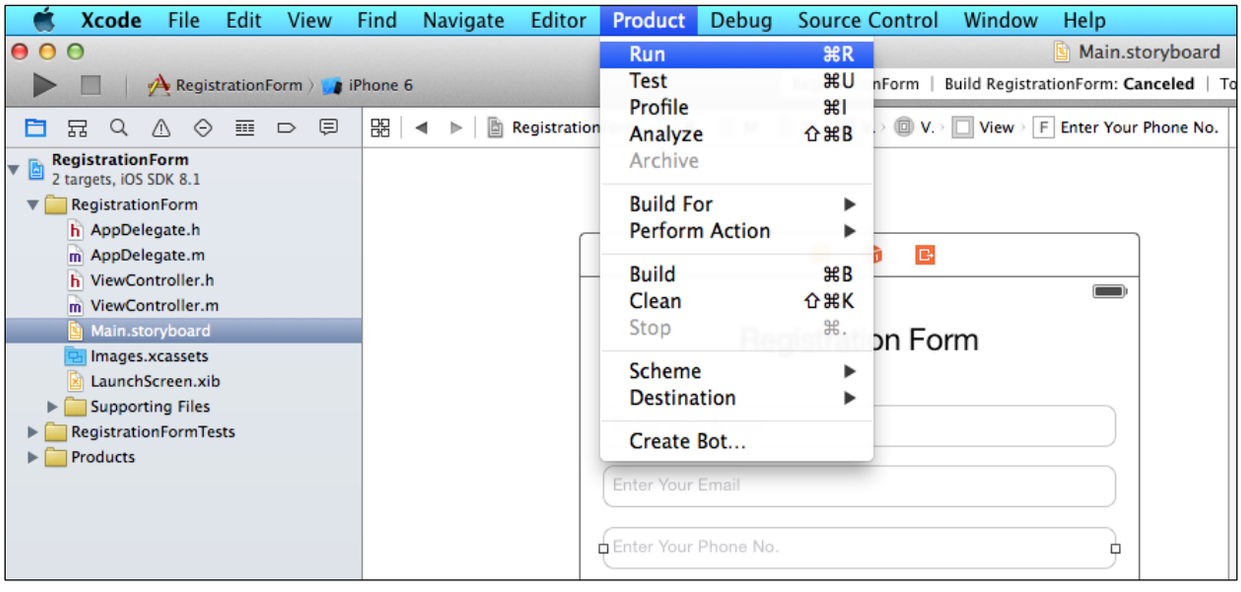

图 28: 运行注册表单应用

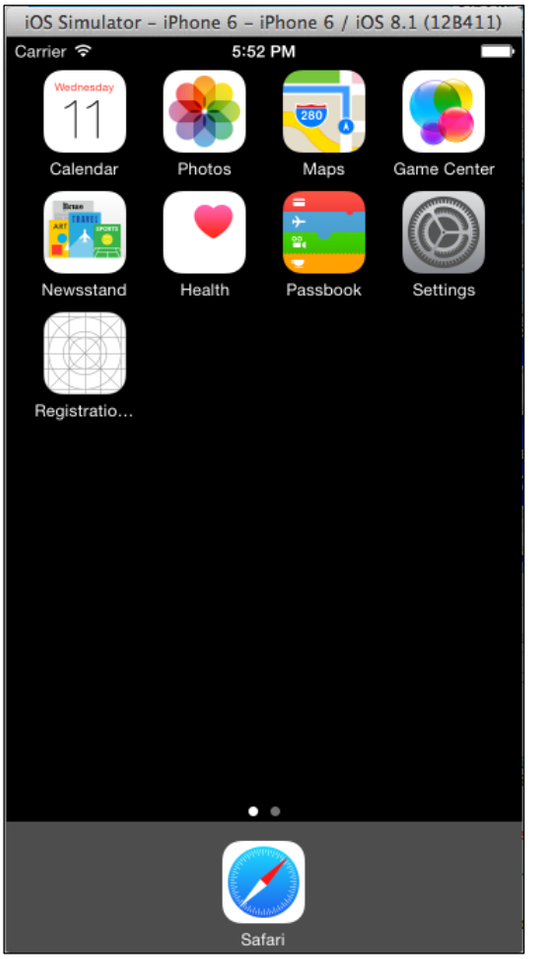

图 29: 模拟器屏幕上的注册表单应用

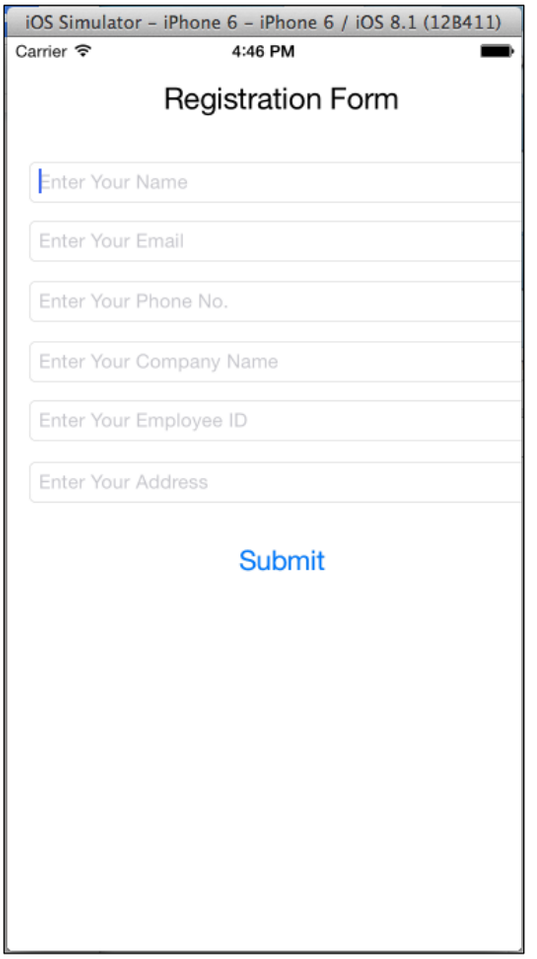

图 30: 模拟器上注册表单应用的输出画面

源码

1. ViewController.h 文件

```
#import <UIKit/UIKit.h>
@interface ViewController :UIViewController

- (IBAction)TouchButton:(id)sender;                          // Button Action variable
@property (weak, nonatomic) IBOutletUITextField *EnterName;        // Name TextView
@property (weak, nonatomic) IBOutletUITextField *EnterEmail;       // Email TextView
@property (weak, nonatomic) IBOutletUITextField *EnterContact;     // Contact TextView
@property (weak, nonatomic) IBOutletUITextField *EnterCompany;     //Company TextView
@property (weak, nonatomic) IBOutletUITextField *EnterId;    // Id TextView
@property (weak, nonatomic) IBOutletUITextField *EnterAddress;     // Address TextView
@property (weak, nonatomic) IBOutletUILabel *Label;          // Label UILabel
- (IBAction)backgroundTap:(id)sender;           // Onclick on background Keyboard should be
invisible
@end
```

2. ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@implementation ViewController

- (void)viewDidLoad
{
    [superviewDidLoad];

}

- (void)didReceiveMemoryWarning
{
    [superdidReceiveMemoryWarning];

}

- (IBAction)TouchButton:(id)sender {
    _EnterName.text = @"";                      // action on clickOnButton
    _EnterEmail.text = @"";                     // action on clickOnButton
    _EnterContact.text = @"";                         // action on clickOnButton
    _EnterCompany.text = @"";                   // action on clickOnButton
    _EnterId.text = @"";                        // action on clickOnButton
    _EnterAddress.text = @"";                         // action on clickOnButton

}- (IBAction)backgroundTap:(id)sender    // Action when tap on background
{
    [self.viewendEditing:YES];
}
@end

@end
```