Wiley：移动开发工程师实验报告

云课堂昵称：MartinZhou　　实验日期：2015.12.21

一、实验题目

创建发送电子邮件、短信和拨电话的应用

二、实验要求

创建一个叫作 emailer 的应用,让用户能在该应用内发送电子邮件、短信和拨打电话号码。要创建这一应用,你需要执行下面这些任务:

1.在 Xcode 中使用 Single View Application 创建应用
2.从应用内调用邮件功能
3.从应用内调用短信功能
4.从应用内调用电话功能

三、操作步骤

1. 打开 Xcode 并选择 Create a new Xcode project 选项
2. 选择 Single View Application 模板,并点 Next
3. 下一个界面上,将项目命名为 MZEmailer,选择 Objective-C 作为编程语言,然后点 Next
4. 选择项目保存位置,然后点 Create
5. 在 Xcode 中打开 MZEmailer 项目。现在你需要手动添加一些库，你还需要在 ViewController.h 文件中添加 MFMailComposeViewController 到 MZEmailer 项目,用于发送电子邮件和短信。需要添加的库见图1：

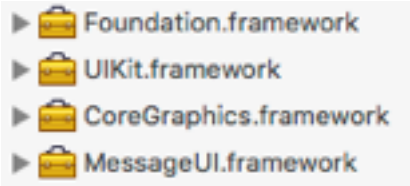

图 1

6. 要添加新库,选择 General 选项卡,并点 Linked Frameworks and Libraries 部分的+图标。然后选择并添加需要的库。
7. 选择 Main.storyboard 文件并设计应用的用户界面,过程中需要用到实用工具面板中的 UILabel、按钮等对象，如图 2 所示:

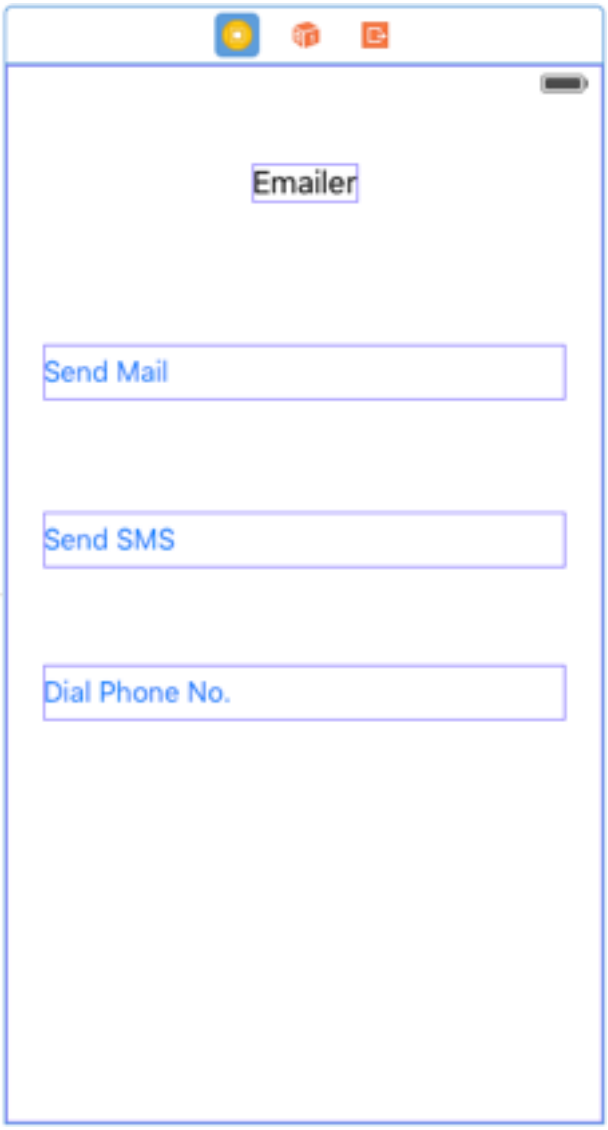

图 2

8. 添加 Send Mail 按钮,需要右键按住这个按钮,并拖放到 ViewController.m。将按钮命名为SendMail,然后点 Connect。
9. 类似为应用添加 Send SMS 和 Dial Phone No.按钮。
10. 要发送电子邮件,你需要在 ViewController.h 文件中添加#import <MessageUI/MessageUI.h>,具体代码如下所示:

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

@interface ViewController : UIViewController <MFMailComposeViewControllerDelegate, MFMessageComposeViewControllerDelegate>
```

11. 然后,编辑 ViewController.m 文件,如下面代码所示。这里使用了 MFMailComposeViewController 方法,它不能在 Xcode 模拟器上工作,因此你需要在实际设备上运行这段代码。

```
- (IBAction)SendEmail:(id)sender {

    // Email Subject
    NSString *eSub = @"Test Email";
```

```
    // Email Body
    NSString *mBody = @"M9d3 EMailer app";
    // Recipient address
    NSArray *recipients = [NSArray arrayWithObject:@"john@gmail.com"];

    MFMailComposeViewController *Ecomp = [[MFMailComposeViewController alloc] init];
    Ecomp.mailComposeDelegate = self;
    [Ecomp setSubject:eSub];
    [Ecomp setMessageBody:mBody isHTML:YES];
    [Ecomp setToRecipients:recipients];
    // Present mail view controller on screen
    [self presentViewController:Ecomp animated:YES completion:NULL];
}

- (void) mailComposeController:(MFMailComposeViewController *)controller
          didFinishWithResult:(MFMailComposeResult)result
                        error:(NSError *)error
{
    switch (result)
    {
        case MFMailComposeResultCancelled:
            NSLog(@"Mail cancelled");
            break;
        case MFMailComposeResultSaved:
            NSLog(@"Mail saved");
            break;
        case MFMailComposeResultSent:
            NSLog(@"Mail sent");
            break;
        case MFMailComposeResultFailed:
            NSLog(@"Mail sent failure: %@", [error localizedDescription]);
            break;
        default:
            break;
    }

    // Close the Mail Interface
    [self dismissViewControllerAnimated:YES completion:NULL];

}
```

12.再在 ViewController.m 文件中添加 SendSMS 按钮的代码。这里添加了 MFMessageComposeViewController类,为发送短信提供标准系统用户界面。该类会调用 canSendText 类方法来确保用户设备配置好了短信发送功能。代码如下所示:

```
- (IBAction)SendSMS:(id)sender {

        MFMessageComposeViewController *controller =
[[[MFMessageComposeViewController alloc] init] self];
    if([MFMessageComposeViewController canSendText])
        {
        controller.body = @"This is Test SMS from Emailer app.";
        controller.recipients = [NSArray arrayWithObjects:@"12345678", nil];
        controller.messageComposeDelegate = self;

        [self presentViewController:controller animated:YES completion:nil];
    }
```

```
    }
- (void)messageComposeViewController:(MFMessageComposeViewController *)controller
                 didFinishWithResult:(MessageComposeResult)result
    {
        switch (result) {
            case MessageComposeResultCancelled:
                NSLog(@"Cancelled");
                break;
            case MessageComposeResultSent:
                NSLog(@"Message sent");
                break;
            default:
                break;
    }
        [self dismissViewControllerAnimated:NO completion:nil];
}
```

13.SendSMS 按钮会打开 iOS 实际设备的短信功能。由于这段代码是在 Xcode 模拟器上运行的,短信服务是不能用的,你会得到如图 3 所示的警告消息:

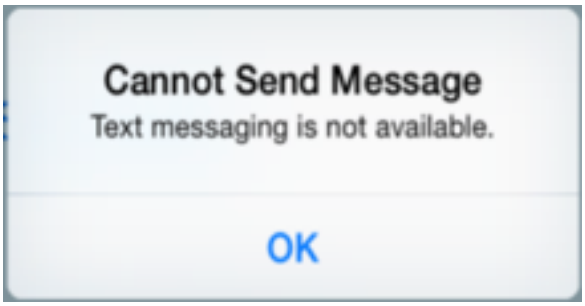

图 3

14.再在 ViewController.m 文件中添加 DialPhone 按钮的代码,这段代码会在 Xcode 中自动呼叫电话号码:

```
- (IBAction)DialPhone:(id)sender
    {
            [[UIApplication sharedApplication]openURL:[NSURL URLWithString:@“tel:
+8612345678"]];
}
```

四、实验结果

实验结果及演示视频见附件，代码如下:

——

ViewController.h文件

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

@interface ViewController : UIViewController <MFMailComposeViewControllerDelegate,
MFMessageComposeViewControllerDelegate>
- (IBAction)SendEmail:(id)sender;
```

```
- (IBAction)SendSMS:(id)sender;
- (IBAction)DialPhone:(id)sender;
@end
```

——

ViewController.m文件

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)SendEmail:(id)sender {

    // Email主题
    NSString *eSub = @"Test Email";
    // Email内容
    NSString *mBody = @"M9d3 EMailer app";
    // Email收件人
    NSArray *recipients = [NSArray arrayWithObject:@"john@gmail.com"];

    MFMailComposeViewController *Ecomp = [[MFMailComposeViewController alloc] init];
    Ecomp.mailComposeDelegate = self;
    [Ecomp setSubject:eSub];
    [Ecomp setMessageBody:mBody isHTML:YES];
    [Ecomp setToRecipients:recipients];
    // 使mail界面显示在屏幕上
    [self presentViewController:Ecomp animated:YES completion:NULL];
}

- (void) mailComposeController:(MFMailComposeViewController *)controller
          didFinishWithResult:(MFMailComposeResult)result
                        error:(NSError *)error
{
    switch (result)
    {
        case MFMailComposeResultCancelled:
            NSLog(@"Mail cancelled");
            break;
        case MFMailComposeResultSaved:
            NSLog(@"Mail saved");
            break;
        case MFMailComposeResultSent:
            NSLog(@"Mail sent");
```

```
            break;
        case MFMailComposeResultFailed:
            NSLog(@"Mail sent failure: %@", [error localizedDescription]);
            break;
        default:
            break;
    }

    // 关闭mail界面
    [self dismissViewControllerAnimated:YES completion:NULL];

}

- (IBAction)SendSMS:(id)sender {

            MFMessageComposeViewController *controller =
[[[MFMessageComposeViewController alloc] init] self];
    if([MFMessageComposeViewController canSendText])
        {
        //设置短信内容
        controller.body = @"This is Test SMS from Emailer app.";
        //设置短信接收号码
        controller.recipients = [NSArray arrayWithObjects:@"12345678", nil];
        controller.messageComposeDelegate = self;

        [self presentViewController:controller animated:YES completion:nil];
    }
    }
- (void)messageComposeViewController:(MFMessageComposeViewController *)controller
             didFinishWithResult:(MessageComposeResult)result
    {
        switch (result) {
            case MessageComposeResultCancelled:
                //测试，点击取消时打印如下内容
                NSLog(@"Cancelled");
                break;
            case MessageComposeResultSent:
                //测试，点击发送时打印如下内容
                NSLog(@"Message sent");
                break;
            default:
                break;
    }
        [self dismissViewControllerAnimated:NO completion:nil];
}

- (IBAction)DialPhone:(id)sender{

            [[UIApplication sharedApplication]openURL:[NSURL
URLWithString:@"tel:+8612345678"]];

}
@end
```

五、总结反思

1.只有在真机测试中才能使用短信和电话功能，第一次让我注意到模拟器并不能处理“大部分”功能，真机测试很重要。
2.虽然是真机测试，但是在代码中添加的NSLog内容还是可以在控制台输出，方便查看Bug
3.代理的使用更加频繁，同时这也加深了我对代理的理解，但对代理的使用也产生了一些新的疑问，需要继续查阅资料去了解。

六、作业提交

1. 请将本文档按照《Wiley移动开发_X章X节_云课堂昵称》的名称命名；
2. 请将本文档、源代码文件打包以附件形式上传到课程作业部分