# Wiley：移动开发工程师实验报告

云课堂昵称：MartinZhou　　实验日期：2015.12.27

## 一、实验题目

使用 Event Kit 框架创建日历和提醒

## 二、实验要求

使用 `EKEventStore` 类来为用户日历数据库创建事件。

1. 访问日历数据
2. 使用EventKitUI框架创建事件
3. 为日历事件创建一个提醒

## 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.将项目命名为MyEvents。Language下拉列表中选择Swift。点Next推进到下一个界面

4.在下一个页面上,在对应位置选择MyEvents项目,然后点Create。这会在Xcode中打开Swift项目

5.下面,打开ViewController.swift并导入EventKitUI，如图1

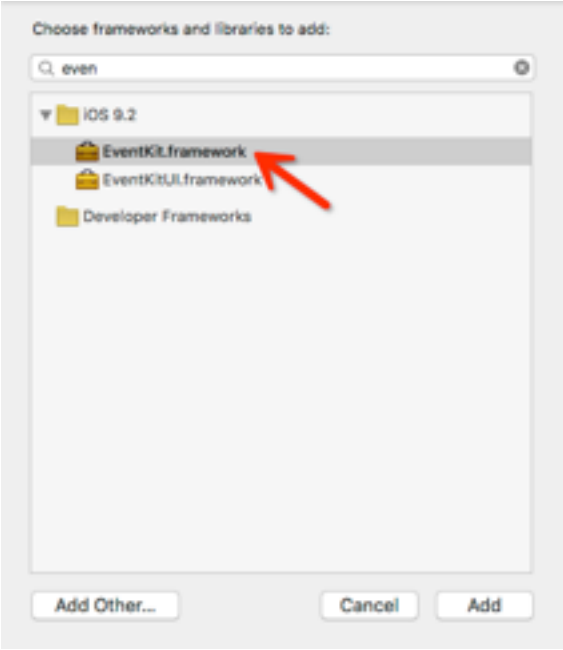


图 1

6.苹果 iOS 不允许任何应用访问其个人数据库。要访问用户的 Event Store,你需要从 iOS

获得权限。你需要一个用户界面来请求用户授权你的应用访问数据库的特定部分,例如日历。要获得访问权限,你需要使用 EKEventStore 类的 authorizationStatusForEntityType 类方法,并传递参数 EKEntityTypeEvent,以获得用户授权。在 AppDelegate.swift 文件中使用如下代码:

```
import UIKit
import EventKit
import EventKitUI

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

var window: UIWindow?

        func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {

    window = UIWindow(frame: UIScreen.mainScreen().bounds)

    if let Display = window
    {
       Display.backgroundColor = UIColor.whiteColor()
       Display.rootViewController = ViewController()
       Display.makeKeyAndVisible()
    }

    return true
  }
```

7.下面运行模拟器,做法是点 Project 菜单中的 Run,或是使用 Command+Run 路径。你会得到如图 2 所示的输出结果。图 3 显示了 MyEvents 应用如何请求用户许可来访问集中式日历数据库。点 OK 来允许访问。

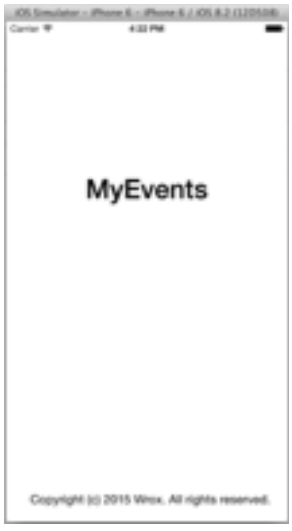


图 2


图 3

8.有了权限之后,你可以从Settings中改变MyEvents应用的配置
9.访问日历数据库后,从Event Store读取数据。为此,使用calendarsForEntityType:实例方法来遍历日历对象。要在 Event Store 中添加事件,使用

sourceInEventStore:sourceType:sourceTitle:方法。该方法被用于查找特定类型和标题的事件源,同时你还需要 EKSourceTypeLocal 来查找日历类型。找到日历并添加事件后,在 ViewController.swift 添加如下代码,用于创建 NewEvent,并将新事件创建到日历 Event Store。如下面代码所示:

```
import UIKit
import EventKit
import EventKitUI

class ViewController: UIViewController {

override func viewDidLoad() {
   super.viewDidLoad()

   let eventStore = EKEventStore()

   switch EKEventStore.authorizationStatusForEntityType(EKEntityTypeEvent){

   case.Authorized:readEventsFromCalendar(eventStore)

   case.Denied:displayAccessDenied()

   case.NotDetermined:

   eventStore.requestAccessToEntityType(EKEntityTypeEvent, completion:
      {[weak self](granted: Bool, error: NSError!) -> Void in

      if granted {
               self!.insertEventIntoStore(eventStore)
      } else {
               self!.displayAccessDenied()
         }
         })

   case.Restricted:displayAccessRestricted()

      }
         }

   func displayAccessDenied(){

      print("Access to the event store is denied.")

   }

   func displayAccessRestricted(){

      print("Access To Event store is restricted.")

   }

   func readEventsFromCalendar(EventStore: EKEventStore){

         let CalType = ["Local","Exchange","Birthday"]
         let CalData = EventStore.calendarsForEntityType(EKEntityTypeEvent) as
[EKCalendar]
```

```
            for cal in CalData{

                print("Calendar Title = \(cal.title)")
                print("Calendar type = \(CalType[Int(cal.type.value)])")

                let color = UIColor(CGColor: cal.CGColor)
                print("Calendar color = \(color)")

                if cal.allowsContentModifications == false {
                    print("Calendar cannot be modified")
                }
            }
        }

        override func didReceiveMemoryWarning() {
            super.didReceiveMemoryWarning()
            // Dispose of any resources that can be recreated.
        }

        func sourceInEventStore(
            eventStore: EKEventStore,
            type: EKSourceType,
            title: String) ->EKSource? {

            for source in eventStore.sources as [EKSource]{

            if
//              source.sourceType.value == type.value &&
                source.title.caseInsensitiveCompare(title) ==
NSComparisonResult.OrderedSame{

                return source
                }
            }

            return nil
        }

        func AddEvent(title: String, startDate: NSDate, endDate: NSDate, inCalendar:
EKCalendar, inEventStore: EKEventStore, notes: String) ->Bool{

            if inCalendar.allowsContentModifications == false
                {
                    print("Calendar does not allow modification”)
                    return false
                }

            //event created
            var EventSet = EKEvent(eventStore: inEventStore)
            EventSet.calendar = inCalendar
            EventSet.title = title      //set title
            EventSet.notes = notes      //set note
            EventSet.startDate = startDate  //set start date
            EventSet.endDate = endDate      //set end date

            var error:NSError?
            // save event into the calendar
            let EventSetResult = inEventStore.saveEvent(EventSet,
```

```
        span: EKSpanThisEvent,
        error: &error)

    if EventSetResult == false
        {
            if let CalError = error
        {
    print("An error occurred \(CalError)")
        }
    }
    return EventSetResult
}

func insertEventIntoStore(store: EKEventStore){

    let eventSource = sourceInEventStore(store,
        type: EKSourceTypeLocal,
        title: "MyEvent title")

        if eventSource == nil{

            print("Cloud Calendar not configured with device")

            return
        }

    let cal = calWithTitle("Calendar",
        type:EKCalendarTypeLocal,
        source: eventSource!,
        eventType: EKEntityTypeEvent)

    if cal == nil{

        print("Calendar not found.")

        return
        }

    //Start event from now
    let startDate = NSDate()

    //event will end after 60 min, 60 sec
    let endDate = startDate.dateByAddingTimeInterval(0 * 60 * 60)

    if AddEvent("MyEvent",
        startDate: startDate,
        endDate: endDate,
        inCalendar: cal!,
        inEventStore: store,
        notes: ""){

    print("Event created Successfully.")

    } else {

    print("Failed to create the event.")

    return
```

```
    }
}
```

10.再次运行模拟器。输出启动界面如图4所示。这里,你的MyEvent被创建了。

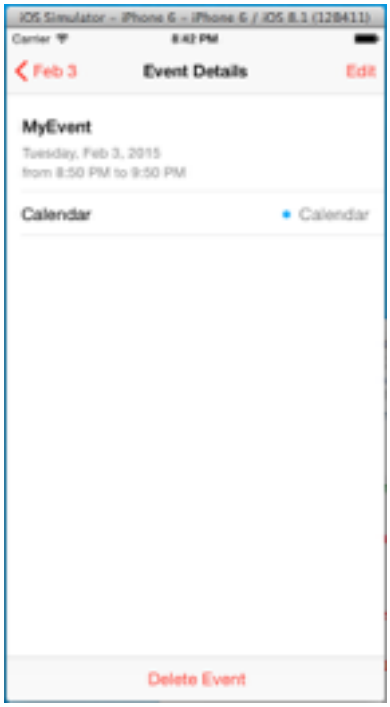


图 4

11.要添加提醒给日历事件,你可以添加如下代码到ViewController.swift文件:

```
func EventAlarm(store: EKEventStore, cal: EKCalendar){

    // start alarm from now
    let startDate = NSDate(timeIntervalSinceNow: 60.0)
    let endDate = startDate.dateByAddingTimeInterval(20.0)
    let EventAlarm = EKEvent(eventStore: store)

        EventAlarm.calendar = cal
        EventAlarm.startDate = startDate
        EventAlarm.endDate = endDate

    let alarm = EKAlarm(relativeOffset: -5.0)

        EventAlarm.title = "MyEvent Alarm"
        EventAlarm.addAlarm(alarm)

    var error:NSError?

    //End Alarm 5 sec before event
    if store.saveEvent(EventAlarm,
        span: EKSpanThisEvent,
        error: &error){

        print("Event saved with alarm")

    } else if
```

```
            let theError = error{

            print("Error : Failed to save event = \(theError)")

            }
        }
    }
```

12.再次运行模拟器,你会得到输出启动画面,提醒会设置成功，如图五所示。

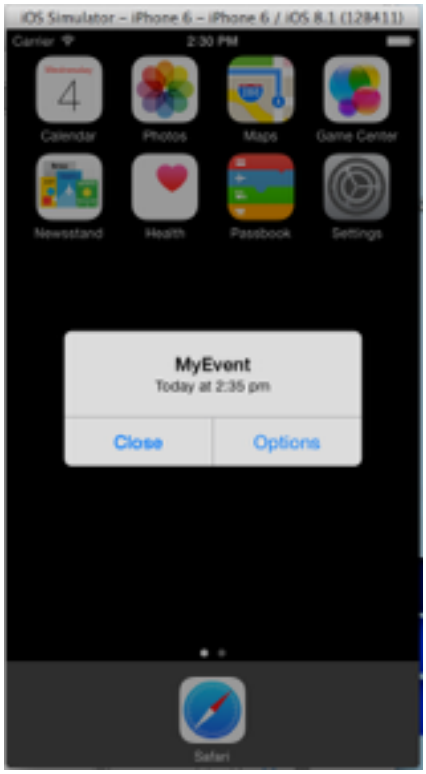


图 5

## 四、实验结果

实验结果见附件工程文件

## 五、总结反思

1.通过使用代码完成UI的呈现，了解了完成UI的另一种方式。

2.诚实的来讲，这次的实验我有很多不解的地方，代码大部分是Copy实验示例中的，但就算如此静态编译器中还是有很多错误，部分原因在于课程代码是用Swift1.0写的，而现在Swift的版本是2.0，语法有一些变化，在此基础上我已经排除了一些BUG，但是剩下的由于我个人对Swift还没有掌握，故暂未找到问题所在。

## 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