WILEY

访问日历数据库

日历项和提醒

创建、编辑或删除事件


从程序上访问日历数据库

从程序上访问日历数据库

你可以从程序上访问用户日历数据库，用于创建、编辑、保存和删除数据库中的事件，步骤如下：

1. 在头文件YDAppDelegate.h中声明navigation controller，并在YDAppDelegate.m文件中实现。

2. 在YDViewController.xib文件中添加界面元素。

3. 导入EventKit和EventKitUI框架。

4. 包含实现文件中需要实现的代码。

从程序上访问日历数据库

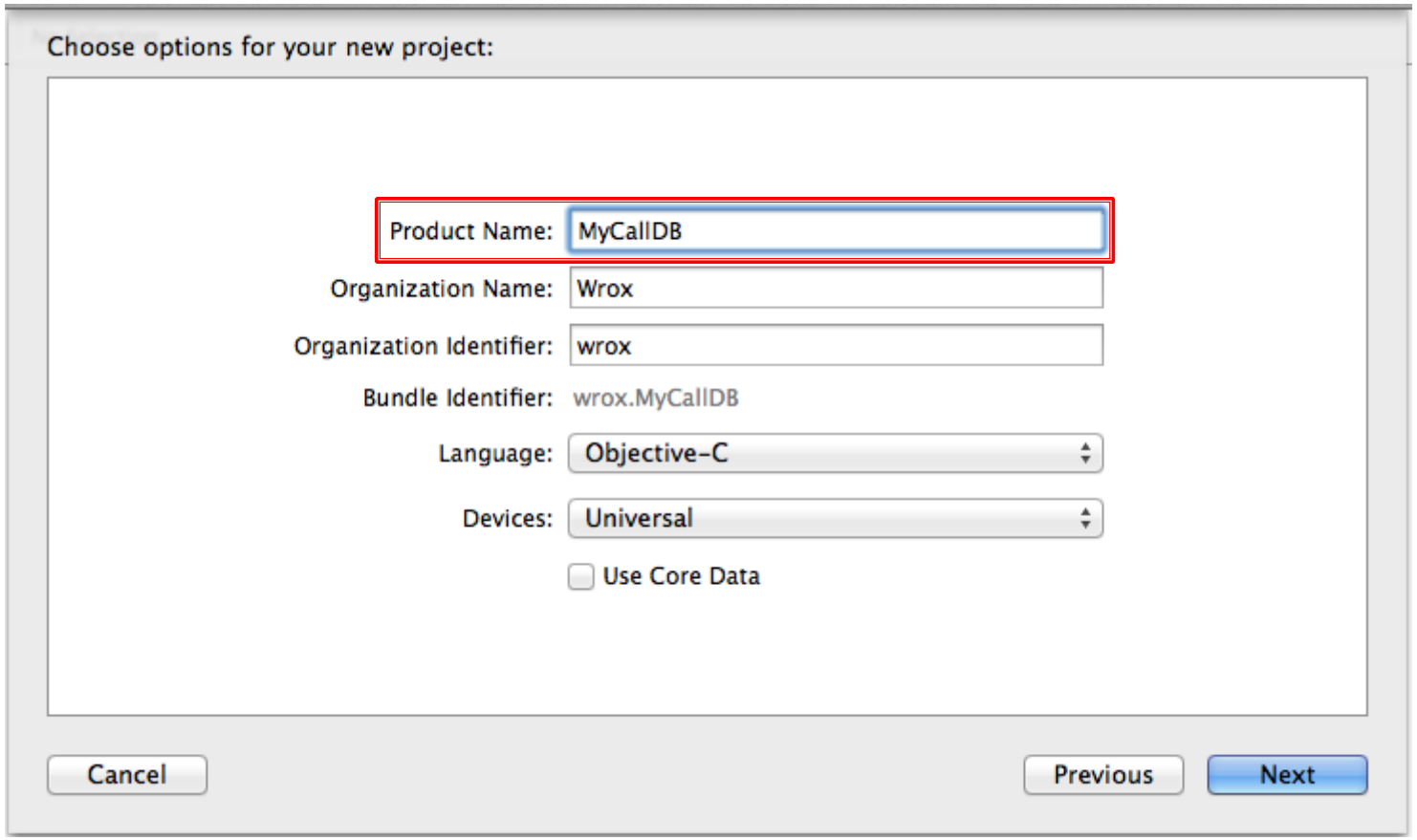
5. 加载用户日历数据库中已经存在的事件。

6. 按需要添加、编辑和保存事件。

7. 需要的话，使用removeEvent:span:commit:error:方法删除事件。

创建事件

- 开启**Xcode，并使用Single View Application模板**创建一个新项目

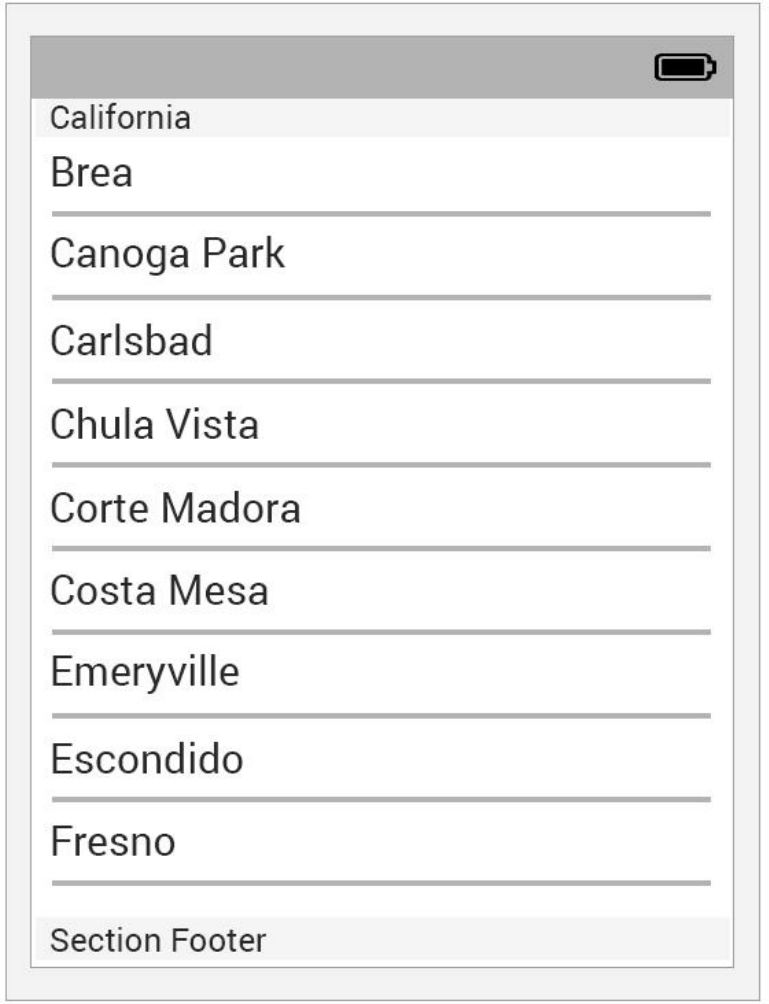


创建事件

```
- (BOOL)application: (UIApplication *)application
didFinishLaunchingWithOptions: (NSDictionary *)launchOptions
{
self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
self.viewController = [[YDViewController alloc]
initWithNibName:@"YDViewController" bundle:nil];
self.navController = [[UINavigationController alloc]
initWithRootViewController:self.viewController];
self.window.rootViewController = self.navController;
self.window.backgroundColor = [UIColor clearColor];
[self.window makeKeyAndVisible];
return YES;
}
```

创建事件

- 打开**YDViewController.xib文件**
- 使用**UITableView**设置用户界面



创建事件

EventKit和**EventKitUI**框架包含操作和处理日历需要的所有类。

```
#import <UIKit/UIKit.h>
#import <EventKit/EventKit.h>
#import <EventKitUI/EventKitUI.h>
@interface YDViewController : UIViewController
@property (weak, nonatomic) IBOutlet UITableView
    *mTableView;
@property(nonatomic,strong) EKEventStore
    *myEventStore;
@property (nonatomic, strong) NSArray *events;
@property(nonatomic,strong)EKCalendar*
    myCalendar;
@end
```

创建事件

```
#import "YDViewController.h"
@interface YDViewController ()<UITableViewDataSource,UITableViewDelegate>
@end
@implementation YDViewController
- (void)viewDidLoad
{
[super viewDidLoad];
[self requestAccessToCalendar];
//Create an Add Event button
UIBarButtonItem *addEventButtonItem =
[[UIBarButtonItem alloc] initWithBarButtonSystemItem:
UIBarButtonSystemItemAdd
target:self
action:@selector(addEvent:)];
self.navigationItem.rightBarButtonItem = addEventButtonItem;
}
```

创建事件

```
-(void)requestAccessToCalendar
{
self.myEventStore = [[EKEventStore alloc] init];
__block BOOL accessGranted = NO;
[self.myEventStore requestAccessToEntityType:EKEntityTypeEvent
completion:^(BOOL granted,
NSError *error) {
// handle access here
accessGranted = granted;
dispatch_async(dispatch_get_main_queue(),^{
accessGranted=YES;
//call loadEvents here now that you know the user has granted access
[self loadEvents];
});
}];
}
```

创建事件

```
-(void)loadEvents
{
EKAuthorizationStatus status = [EKEventStore
authorizationStatusForEntityType:EKEntityTypeEvent];
if (status == EKAuthorizationStatusDenied ||
status == EKAuthorizationStatusRestricted) {
return;
}
```

```
if (self.events)
self.events=nil;
self.myCalendar = [self.myEventStore
    defaultCalendarForNewEvents];
NSDate *startDate = [NSDate date];
NSDate *endDate = [NSDate
    dateWithTimeIntervalSinceNow:86400];
// Create the predicate. Pass it the default calendar.
NSArray *calendarArray = [NSArray
    arrayWithObject:self.myCalendar];
NSPredicate *predicate = [self.myEventStore
predicateForEventsWithStartDate:startDate
endDate:endDate
calendars:calendarArray];
```

创建事件

未来24小时内已经存在于用户日历中的事件将在你的应用中可见。

```
// Fetch all events that match the predicate and
    store in self.events
self.events = [[NSArray alloc] initWithArray:
[self.myEventStore
    eventsMatchingPredicate:predicate]];
[self.mTableView reloadData];
}
```

创建事件

```
- (void)addEvent:(id)sender
{
//Add an event without UI
EKEvent *newEvent = [EKEvent eventWithEventStore:self.myEventStore];
newEvent.calendar = self.myCalendar;
newEvent.title = @"Finalize chapter 13";
newEvent.notes = @"Finish learning chapter 13 of the book Professional
iOS programming";
newEvent.startDate = [NSDate date];
newEvent.endDate =[[NSDate date]
initWithTimeInterval:600
sinceDate:newEvent.startDate];
```

创建事件

```
NSError *err=nil;
[self.myEventStore saveEvent:newEvent span:EKSpanThisEvent
commit:YES error:&err];
if (err)
{
//Handle errors here
}
[self loadEvents];
}
```

创建事件

```
#pragma mark Table View
- (NSInteger)tableView:(UITableView
    *)tableView
numberOfRowsInSection:(NSInteger)section
{
return [self.events count];
}
- (UITableViewCell *)tableView:(UITableView
    *)tableView
cellForRowAtIndexPath:(NSIndexPath
    *)indexPath
```

```
{
static NSString *CellIdentifier = @"Cell";
UITableViewCellAccessoryType
    editableCellAccessoryType
=UITableViewCellAccessoryDisclosureIndicat
    or;
UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:CellIdenti
    fier];
if (cell == nil) {
cell = [[UITableViewCell alloc]
    initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:CellIdentifier];
}
```

创建事件

```
cell.accessoryType = editableCellAccessoryType;
// Get the event at the row selected and display it's
title
cell.textLabel.text = [[self.events
objectAtIndex:indexPath.row] title];
return cell;
}
```

编辑特定单元格的代码是格式化表格的一部分。

创建事件

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

```
(NSIndexPath *)indexPath {
```

```
// Upon selecting an event, create an
    EKEventViewController to display the
```

```
[self.navigationController
    pushViewController:editController
    animated:YES];
}
```

编辑事件

- 事件标题使用title属性
- 事件开始和结束日期使用startDate和endDate 属性
- 事件关联的日历使用calendar属性
- 事件关联的Alarm使用alarms属性
- 事件重复规则使用recurrenceRules属性

保存事件

- 调用myEventStore实例的方法 **saveEvent:span:commit:**
- 使用EKEventStore方法 **saveEvent:span:commit:error:**
- 为**saveEvent:span:commit:error:**方法的span参数指定EKSpanFutureEvents

```
- (void)editEvent:(EKEvent* )theEvent
{
theEvent.calendar = self.myCalendar;
reminder.calendar = [self. myEventStore
    defaultCalendarForNewReminders];
NSError *err;
[self.myEventStore saveReminder:reminder commit: YES
    error:&err];
if (err)
    {
//Handle errors here
     }
  [self loadReminders];
}
```

删除事件

myEventStore实例的**removeEvent:span:commit:error:**方法

EKSpanFutureEvents，用于**removeEvent:span:commit:error:**方法的span参数

保持同步

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(storeChanged:)
name:EKEventStoreChangedNotification
object:self.myEventStore];
```

要编辑日历事件，你需要调用**myEventStore**实例的哪一个方法？

a) saveEven:commit:

b) Editevent:span:commit:

c) saveEvent:span:commit:

d) saveEvent:span

要编辑日历事件，你需要调用**myEventStore**实例的哪一个方法？

a) saveEven:commit:

b) Editevent:span:commit:

c) saveEvent:span:commit:

d) saveEvent:span

总结

你可以从程序上访问用户日历数据库，用于创建、编辑、保存和删除数据库中的事件，步骤如下：

1. 在头文件YDAppDelegate.h中声明navigation controller，并在YDAppDelegate.m文件中实现。这会在iPhone导航控制器中启动你的应用。

2. 在YDViewController.xib文件中添加界面元素。

3. 在YDViewController.h文件中导入EventKit和EventKitUI框架，它们对日历处理是必不可少的。

总结

4. 包含实现文件YDViewController.m中需要实现的代码。这里，首先需要请求对用户日历数据库的访问。

5. 用户许可访问后，加载用户日历数据库中已经存在的事件。这会在你的应用中显示用户日历的所有事件。

6. 按需要添加、编辑和保存事件。

7. 需要的话，使用removeEvent:span:commit:error:方法删除事件。

WILEY