移动应用开发实践

项目实战3 数据存储

本节内容

- 沙箱目录
- 文件管理器
- 文件处理器
- 使用SQLite3数据库
- 使用Core Data框架

数据持久化

- 不同的数据持久化方式适用于不同场合。
  1. 基于字符串的文本文件　txt等
  2. 属性列表文件 plist
  3. 归档与解档
  4. 关系型数据库管理系统SQLite
  5. Core Data框架

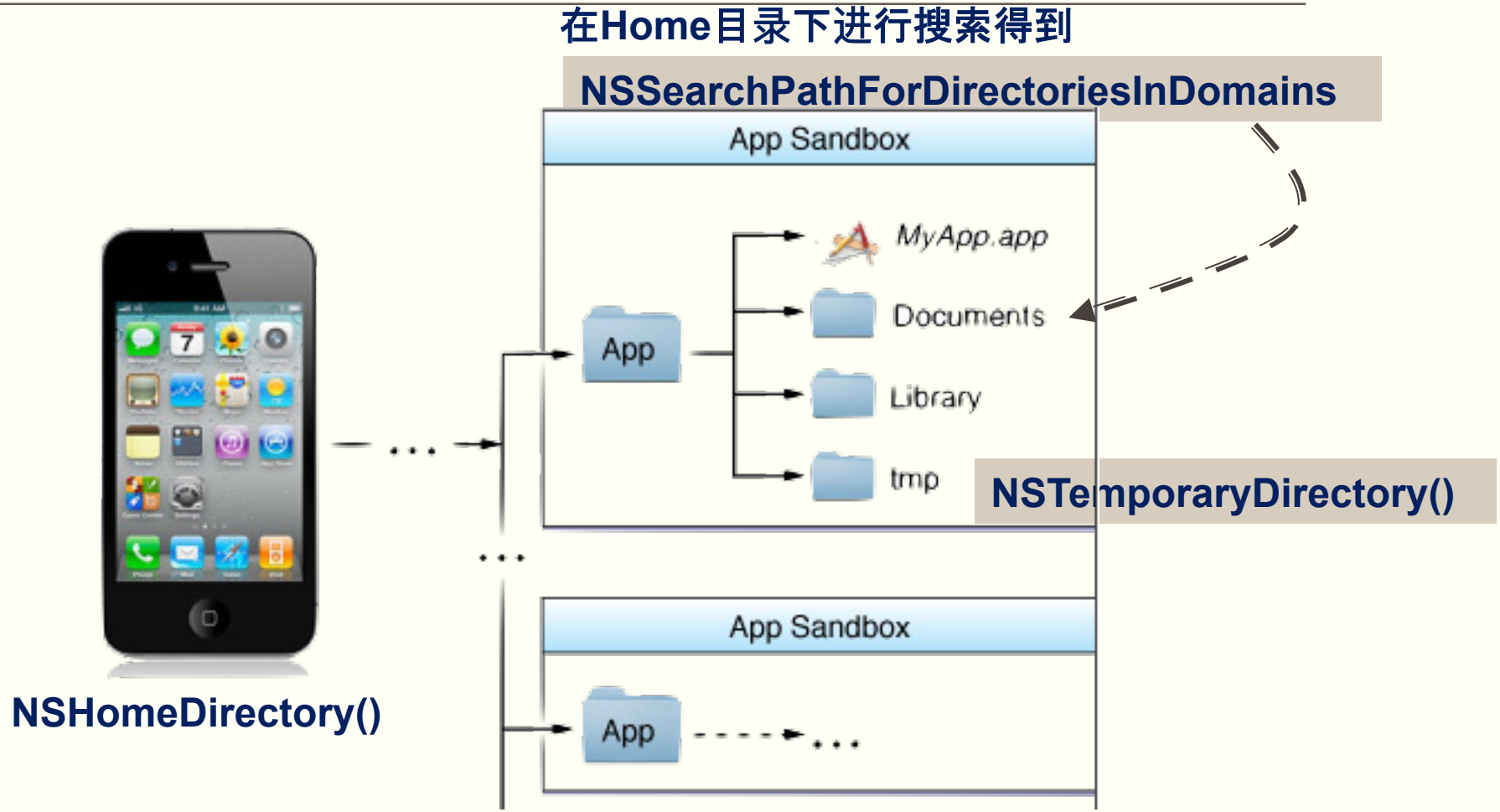
沙箱

安全策略

① Document目录

② Library目录

③ tmp目录



文件管理器

- **NSFileManager类**
  - ① 判断文件是否存在，`fileExistsAtPath`方法
  - ② 创建目录`createDirectoryAtPath`方法
  - ③ 创建文件`createFileAtPath`方法
  - ④ 删除文件`removeItemAtPath`方法
  - ⑤ 查看文件属性`attributesOfItemAtPath`方法

主要是对文件进行的操作以及文件信息的获取

文件处理器

- NSFileHandle类
- 主要是对文件内容进行读取和写入操作

① init?(forReadingAtPath path: String)打开一个文件准备读取

② init?(forWritingAtPath path: String)打开一个文件准备写入

③ init?(forUpdatingAtPath path: String)打开一个文件准备更新

返回NSFileHandle对象

④ 属性availableData从设备或通道返回可用的数据

⑤ seekToEndOfFile()跳到文件末尾

⑥ seekToFileOffset(offset: UInt64)跳到指定文件的偏移量

⑦ writeData(data: NSData)写入数据

⑧ closeFile()关闭文件

实战——调查信息保存及显示

- 创建单视图工程
- 在StoryBoard中画界面如右图所示：UILabel、UITextField、UITextView、UIButton
- 创建文本框、文本视图的outlet引用**firstName、lastemail、resultsView**
- 创建按钮事件分别为**storeResults、showResults**

实战——调查信息保存及显示

```
var _fileManager:ViewController = ViewController()

//存储信息
@IBAction func storeResults(sender: UIButton) {
    //从界面提取信息
    let csvLine = String(format: "%@,%@,%@\n", self.firstName.text!,self.lastName.text!,self.email.
        text!)

    let documentPath = _fileManager.dirDoc()    //获取Documents路径
    let surveyFile = documentPath + "/surveyresults.csv"
    let filemanager = NSFileManager.defaultManager()

    if !filemanager.fileExistsAtPath(surveyFile) {
        filemanager.createFileAtPath(surveyFile, contents: nil, attributes: nil)
    }

    //文件处理器
    let fileHandle = NSFileHandle(forUpdatingAtPath: surveyFile)
    fileHandle?.seekToEndOfFile()
    fileHandle?.writeData(csvLine.dataUsingEncoding(NSUTF8StringEncoding)!)
    fileHandle?.closeFile()

    self.firstName.text="";
    self.lastName.text="";
    self.email.text="";

}
```

实战——调查信息保存及显示

```
@IBAction func showResults(sender: UIButton) {
    let documentPath = _fileManager.dirDoc()
    let surveyFile = documentPath + "/surveyresults.csv"
    let filemanager = NSFileManager.defaultManager()

    if  filemanager.fileExistsAtPath(surveyFile) {
        let fileHandle = NSFileHandle(forReadingAtPath: surveyFile)
        let data = fileHandle?.availableData
        self.resultsView.text = String(data: data!, encoding: NSUTF8StringEncoding)
        fileHandle?.closeFile()
    }


}
//UITextFieldDelegate协议,按回车键时。。。
func textFieldShouldReturn(textField: UITextField) -> Bool {
    self.firstName.resignFirstResponder()
    self.lastName.resignFirstResponder()
    self.email.resignFirstResponder()
    return true
}
```

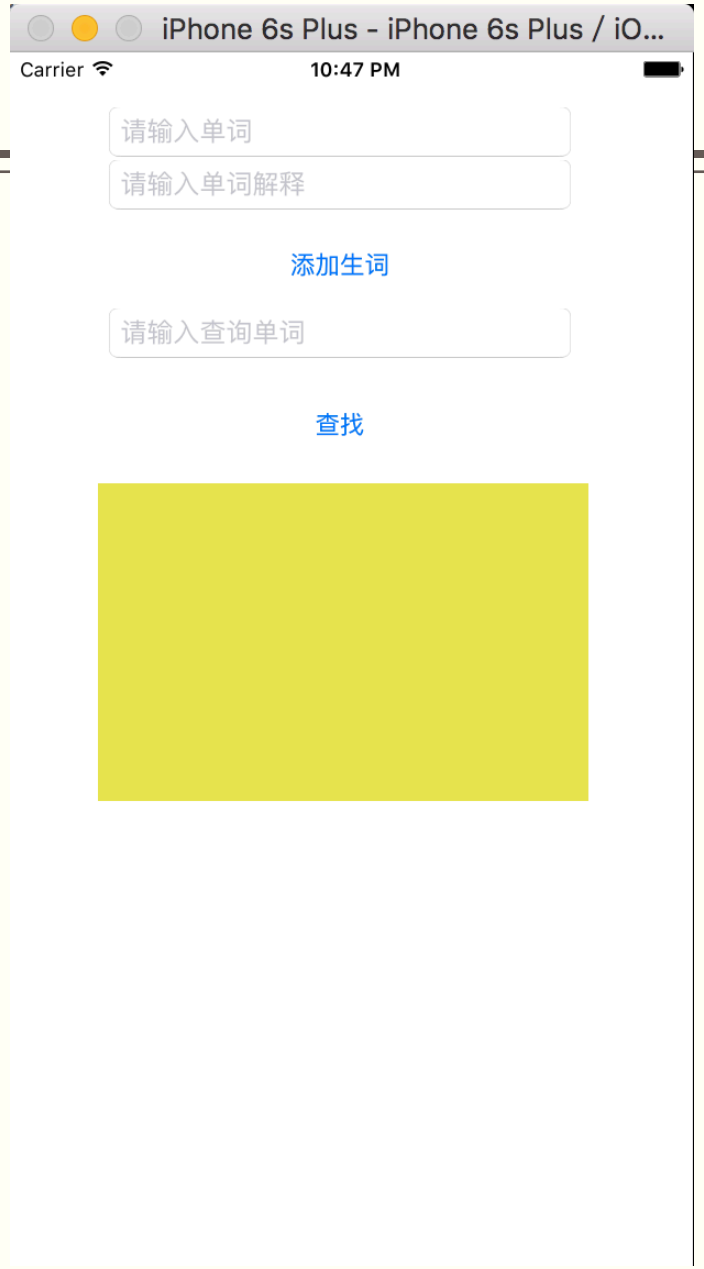
使用SQLite3数据库

- 打开或者创建数据库
- 关闭数据库
- 创建一个表

- 查询操作
- 更新操作
- 插入操作
- 删除操作

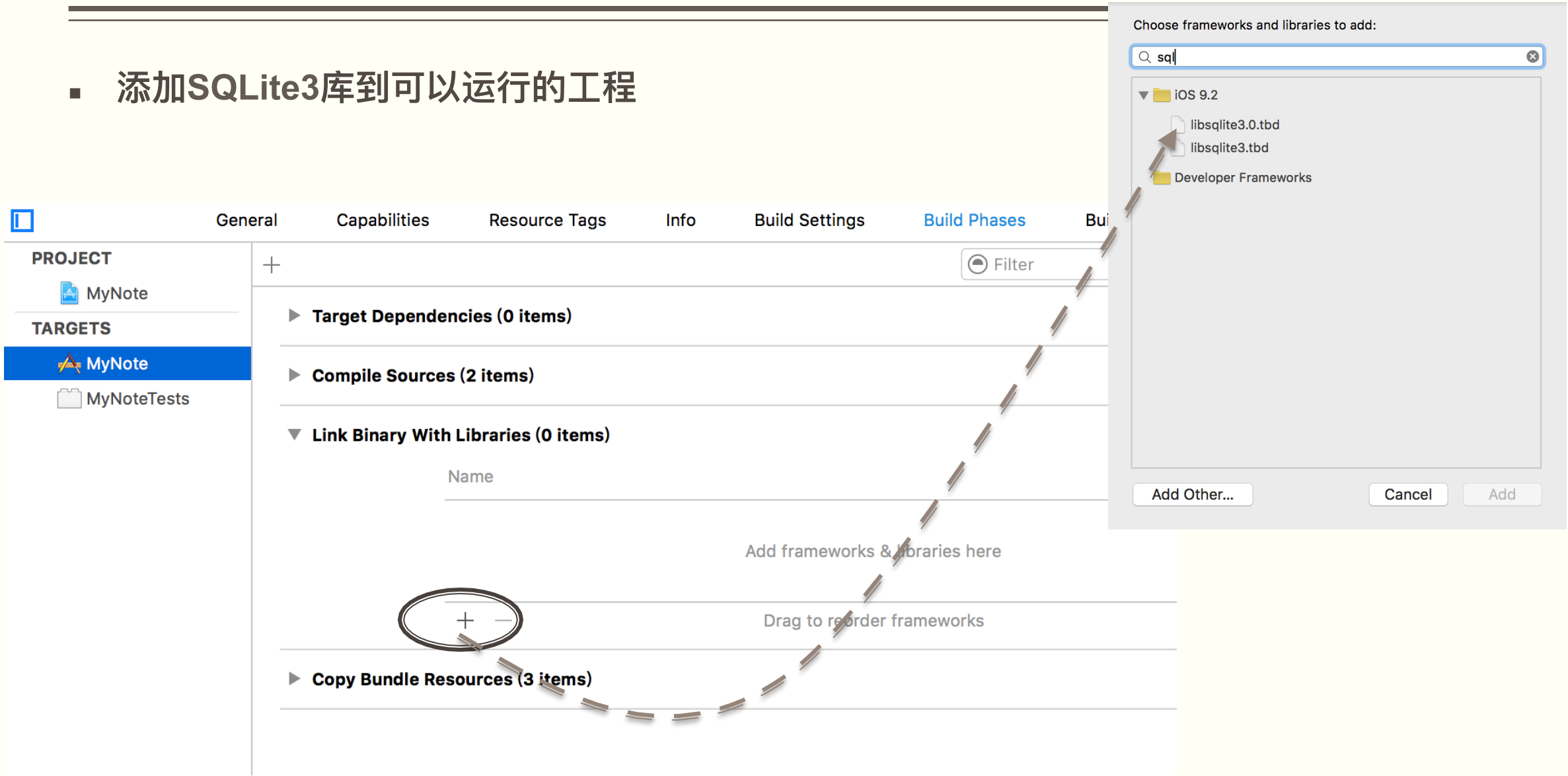
实战——SQLite3

- 创建单视图工程SqlDict
- Stroyboard布局如右图所示
- 三个文本框，两个按钮，一个文本视图
- 创建外部连接和事件



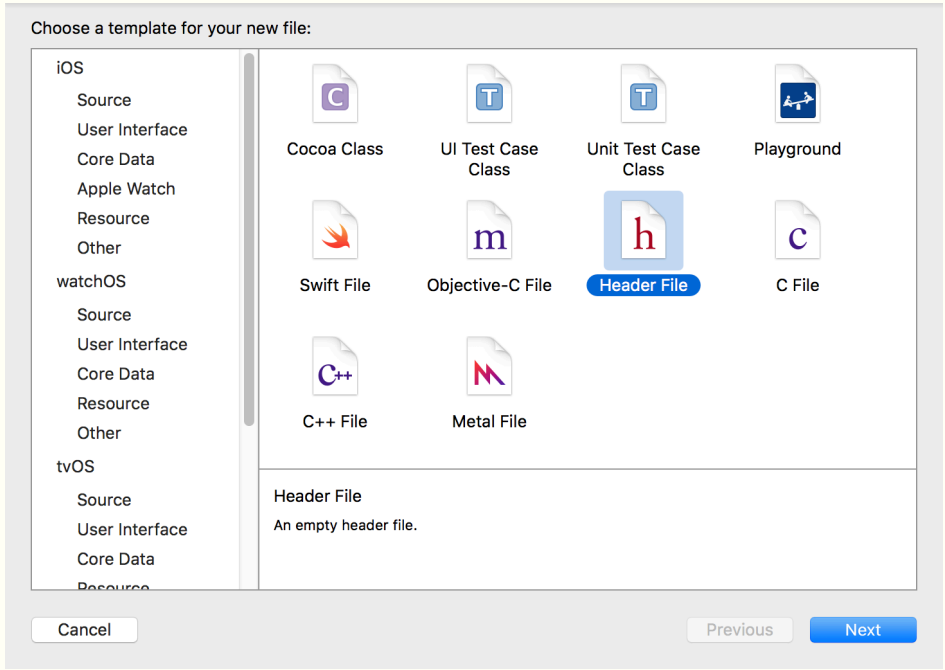
使用SQLite3数据库

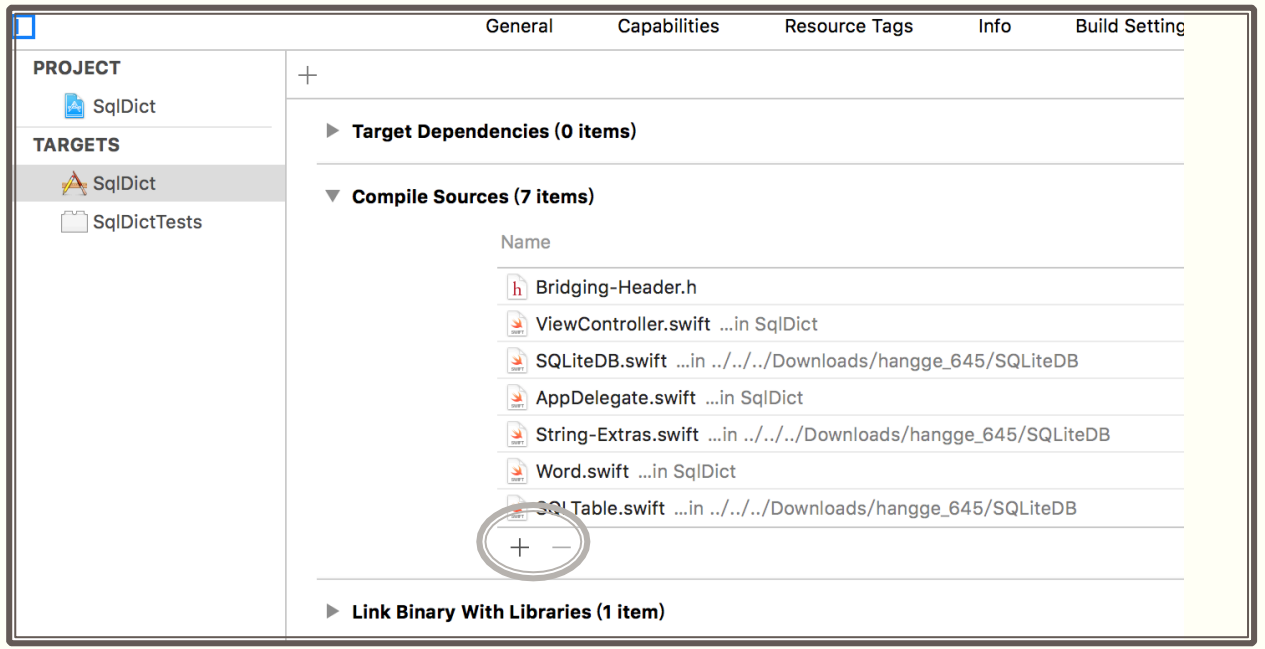
- 添加SQLite3库到可以运行的工程



使用SQLite3数据库

- 创建连接头文件Bridging-Header.h，并导入编译源中

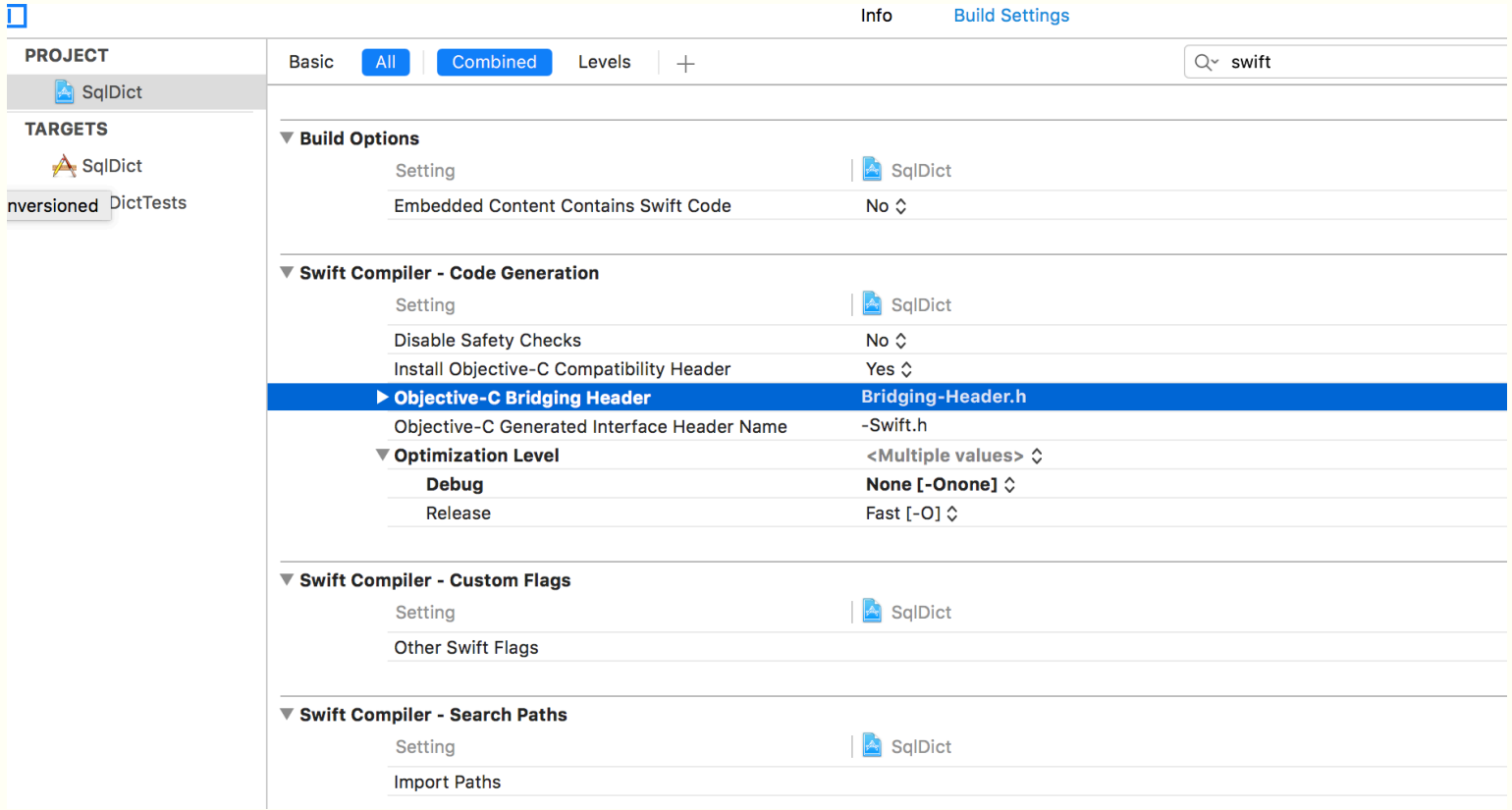





```
#import "sqlite3.h"
#import <time.h>
```

使用SQLite3数据库

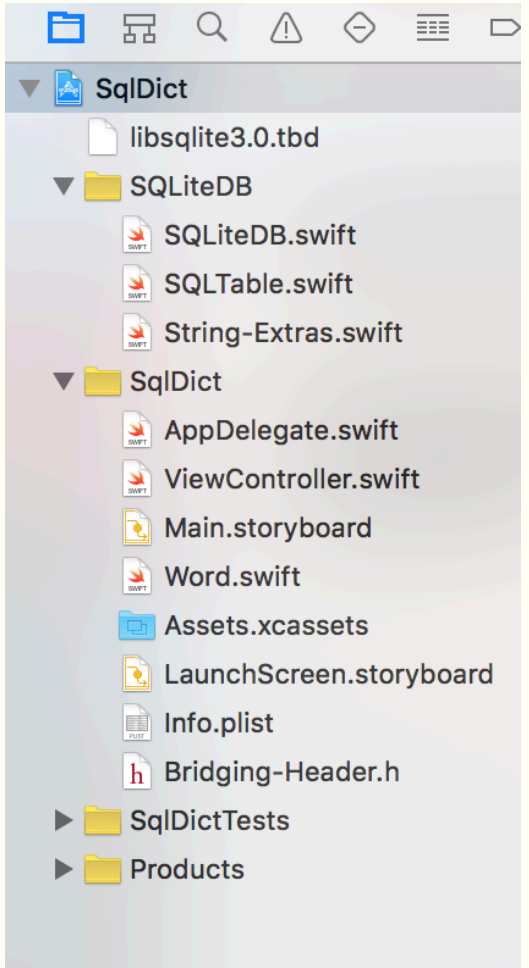
- 在项目编译属性里引用头文件



使用SQLite3数据库

- 导入SQLiteDB的代码（SQLiteDB.swift、SQLTable.swift和String-Extras.swift）

注意：此处导入的代码，是自定义的，不是系统提供的，大家可自行编写自己的数据表处理代码。



使用Core Data框架

- 以面向对象的方式持久化操作SQLite数据库
- 可以将实体保存到持久化存储设备中，也可以在需要的时候将它们取出来
- Core Data底层的持久化存储方式：SQLite、XML文档、内存

> 实体：NSManagedObject类或其子类的实例
>
> 实体间存在1-1，1-N，N-N关系

- 通过NSManagedObjectContext对实体进行增、删、改、查操作