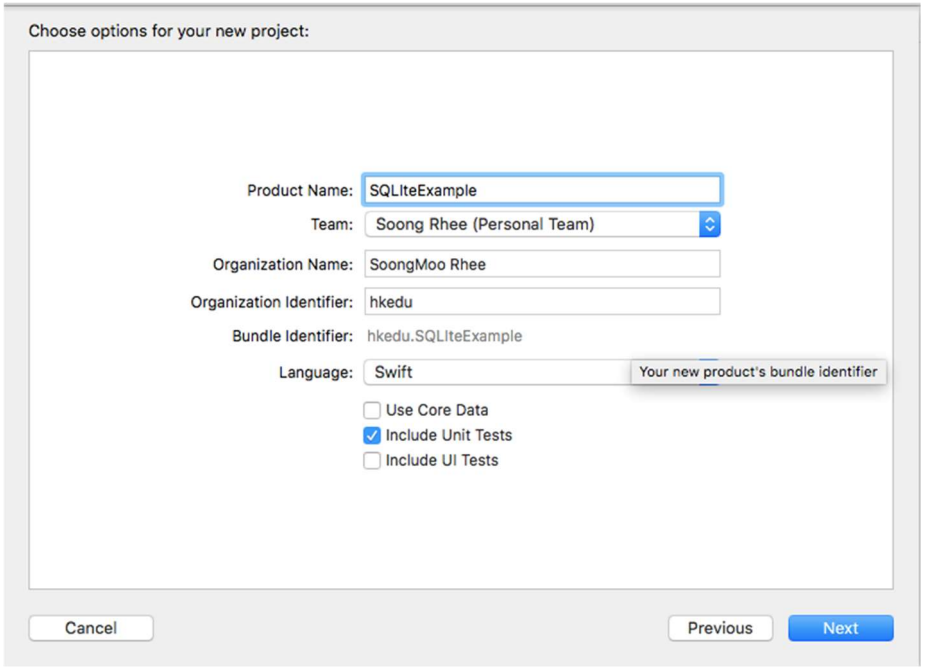
SQLite 데이터베이스 다루기

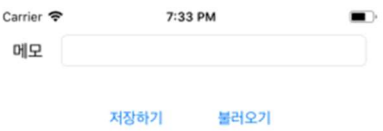
SQLite 를 통해 쿼리문을 이용하여 데이터베이스를 다루는 방법에 대해서 보도록 한다.

1. Xcode 를 실행한 후 Single View Application 을 선택한다.

프로젝트명은 ''SQLIteExample" 라는 이름으로 새 프로젝트를 만든다



2. 화면 구성

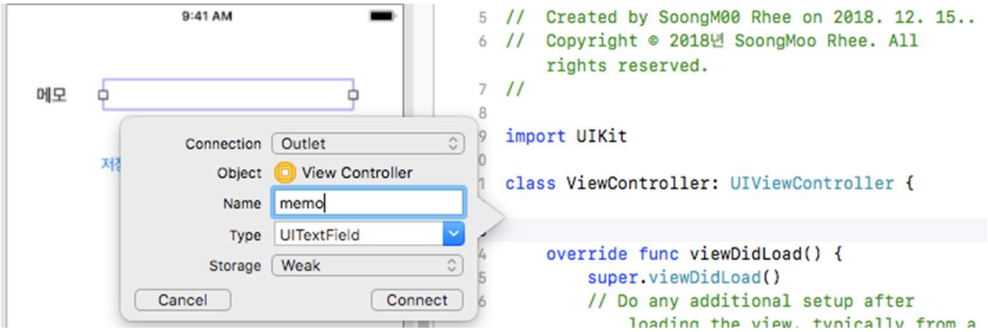


Main.storyboard 파일을 선택해서 오브젝트 라이브러리에 있는 [Label] 하나, [Text Field] 하나, [Button] 두개를 그림과 같이 배열한다.

버튼의 text 를 저장하기, 불러오기로 각각 부여해 준다.

3. 아웃렛 변수와 Action 함수를 추가한다.

Assistant editor 를 열어 [Text Field]의 이름은 "memo"로 주고 "저장하기" 버튼의 이름은 "saveMemo"라고 주고, "불러오기" 버튼은 "loadMemo"라고 이름을 지정해 준다.

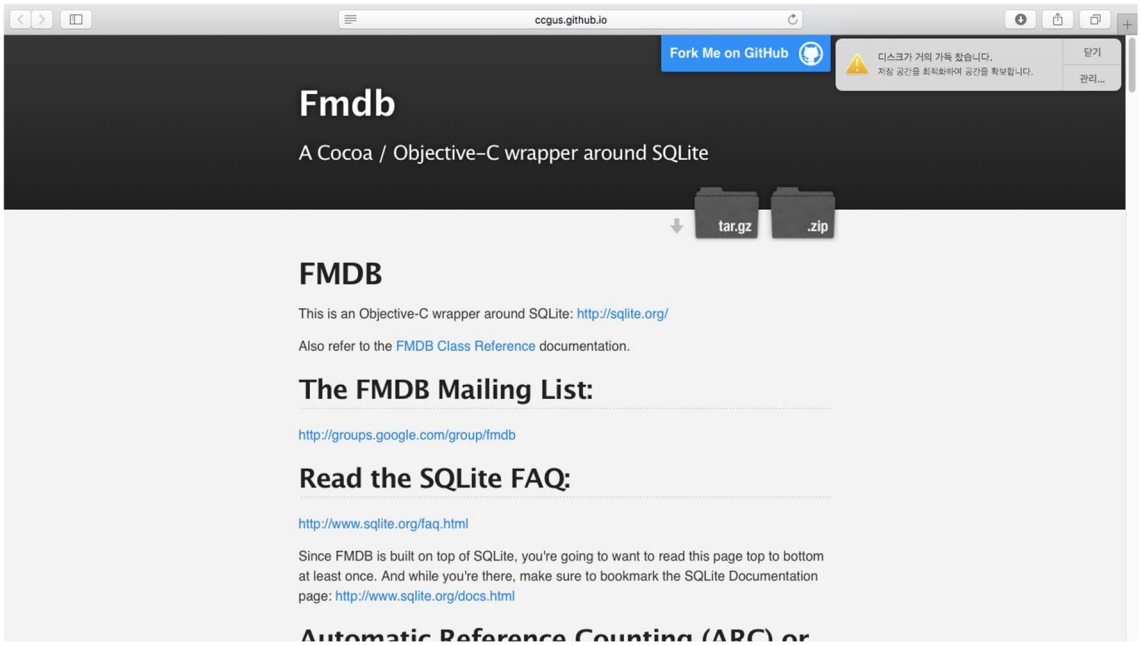



```
1  import UIKit
2
3  class ViewController: UIViewController {
4
5      @IBOutlet weak var memo: UITextField!
6
7      override func viewDidLoad() {
8          super.viewDidLoad()
9          // Do any additional setup after loading the view, typically from a nib.
10     }
11
12     @IBAction func saveMemo(_ sender: UIButton) {
13     }
14     @IBAction func loadMemo(_ sender: Any) {
15     }
16 }
17
```

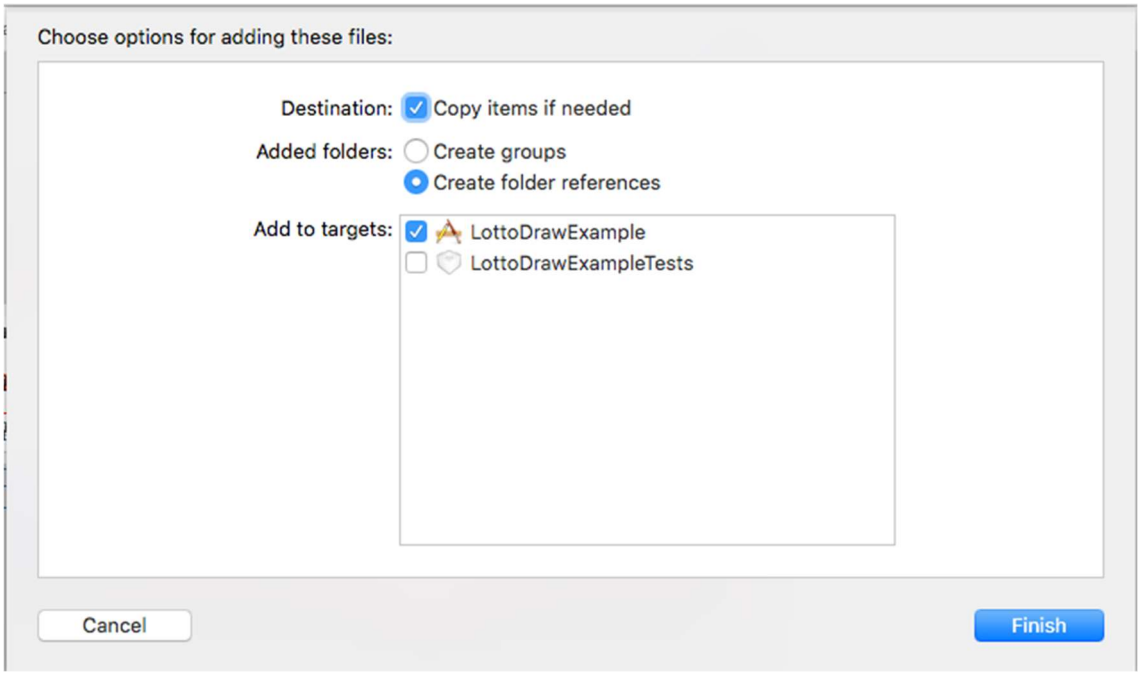
4. 모바일 앱에서는 SQLite 라는 데이터베이스를 많이 사용한다. 안드로이드는 기본으로 사용할 수 있도록 되어 있고 iOS 도 기본으로 사용할 수 있지만 별도의 클래스 파일을 만들어 사용하는 것이 더 편리하다.

이러한 별도의 클래스 파일은 직접 만들 수도 있지만 기존에 공개된 유명한 라이브러리를 사용할 수도 있다. 새로 만들기에는 너무 버거운 작업이므로 SQLite 기반으로 이미 만들어져 있는 FMDB 라는 클래스 파일을 다운로드한다. http://ccgus.github.io/fmdb/ 라는 웹사이트에 접속해 화면 위 오른쪽에 있

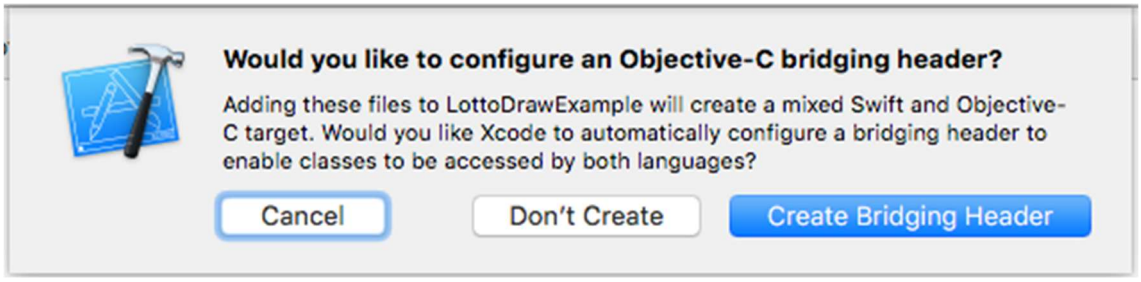
는 .zip 링크를 클릭한 후 FMDB 압축파일을 다운로드한다.



다운로드한 압축 파일의 압축을 해제하면 다양한 파일들을 볼 수 있다.
그 중 src/fmdb 폴더에 있는 *.h 파일과 *.m 파일 전체를 [프로젝트 이름] 폴더로 드래그 앤 드롭해 복사를 해준다.
해당 폴더를 추가하는 것과 관련된 옵션창이 나타난다. [Destination]항목의 [Copy item if needed]에 체크를 켜고 나머지는 기본 설정 그대로 두고 <Finish>를 클릭한다.



추가한 FMDB 관련 폴더는 Objective-C 로 만들어져 있다. 우리가 마드는 프로젝트는 Swift 로 만들기 때문에 2 개의 언어를 썩어서 사용하려면 중간다리가 필요하다. 이때 사용하는 것이 Objective-C Bridging Header 이다.
그러므로 <Create Bridging Header> 를 클릭한다.

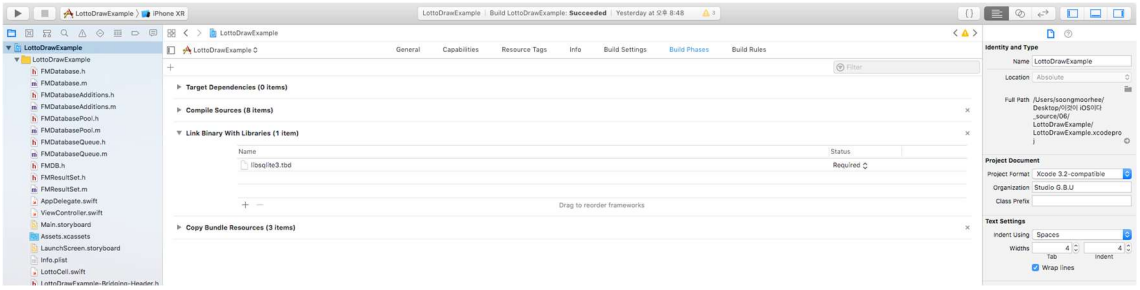


SQLIteExample-Bridging-Header.h 라는 파일이 생성된다.

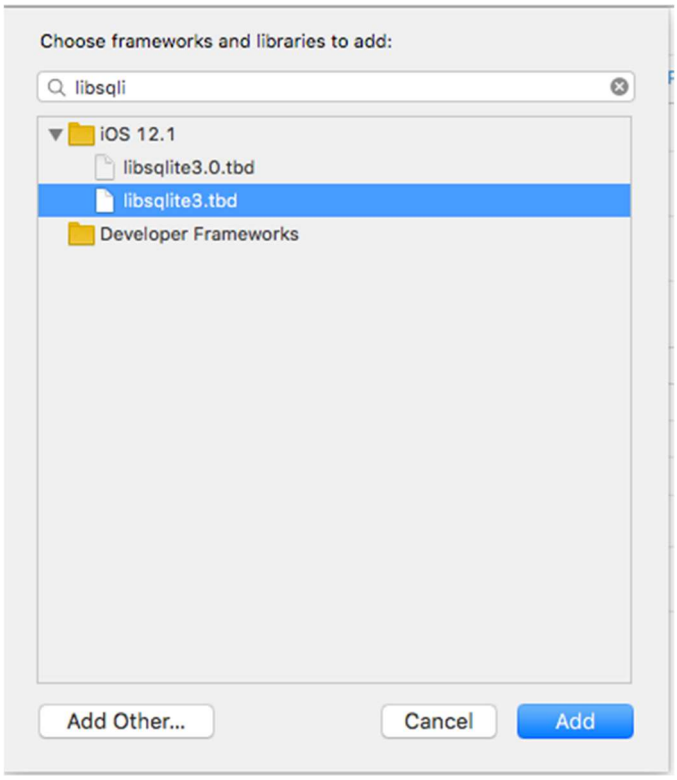
파일의 복사가 완료되고 브리징 헤더 파일도 자동으로 생성되었다.
SQLIteExample-Bridging-Header.h 에 코드를 추가 해준다,

```
#import "FMDB.h"
```

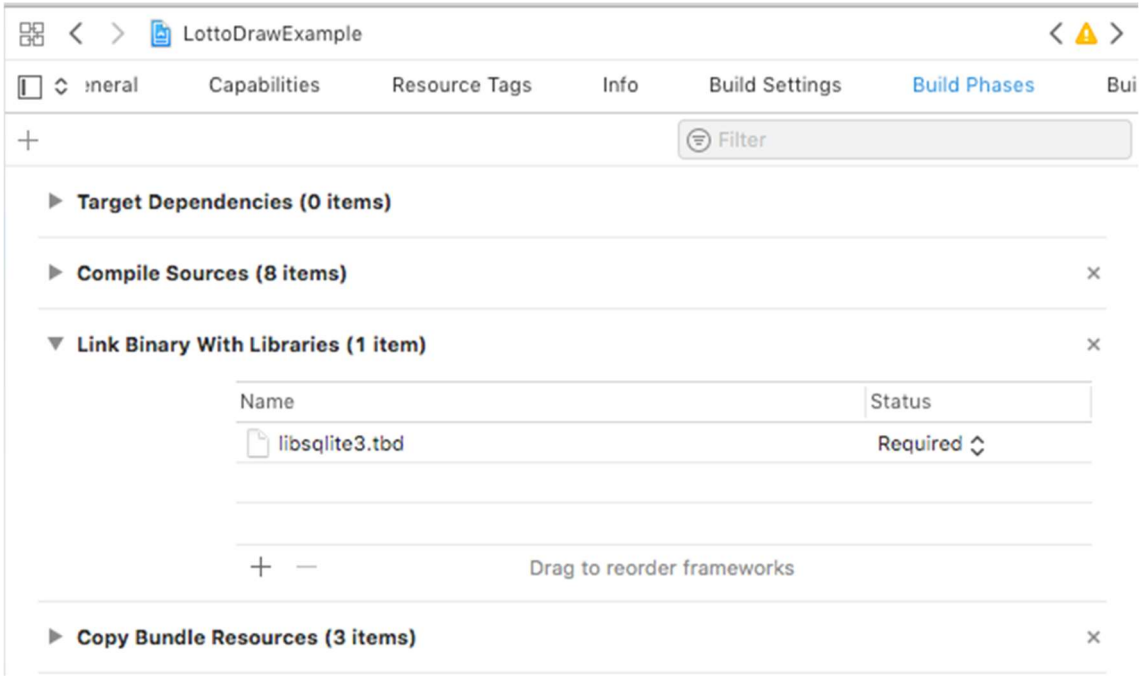
FMDB 클래스 파일들을 사요할 수 있게 설정을 마쳤지만 실제 SQLite 를 사용하기 위한 필수 라이브러리를 추가해야 한다, Project navigator 에서 [프로젝트 이름]을 선택하고 나타난 프로젝트 정보에서 [Build Phases]를 선택한다.



Choose framework and libraries to add 창 위에 있는 검색 창에 'libsqli'라고 입력하면 관련 라이브러리들이 나타난다. 이때 libsqlite3.tbd 를 선택하고 <Add>버튼을 클릭한다



해당 라이브러리가 추가된 것을 확인할 수 있다. 다양한 프로젝트에서 라이브러리나 프레임워크를 추가할 때는 이 기능을 이용하면 된다.



데이터베이스를 사용할 준비가 끝났으니 실제 코드를 이용해서 메모를 저장하고 불러와 보자.
기본적으로 데이터베이스 역시 하나의 파일 형태로 이루어져 있다. 꼭 하나의 파일이 아닐 때도 있지만 모바일 앱에서 SQLite 를 이용할 때는 보통 하나의 파일로 데이터베이스를 관리하며, 이 파일에 접근해서 데이터베이스를 읽고 쓰게 된다.

iOS 에서 어떤 파일에 접근할 때는 미리 해당 파일의 경로를 저장해두는 변수가 필요한다. 그럼 데이터베이스 역시 마찬가지다.
ViewController 클래스 안에 databasePath 라는 변수를 하나 추가한다.

```
1  var databasePath = String()
2
3  override func viewDidLoad() {
4          super.viewDidLoad()
5          // Do any additional setup after loading the view, typically from a nib.
6          let fileMgr = FileManager.default
           let dirPaths = NSSearchPathForDirectoriesInDomains(
7  .documentDirectory, .userDomainMask, true)
8          let docsDir = dirPaths[0]
9          databasePath = docsDir + "/test.db"
10
11         if !fileMgr.fileExists(atPath: databasePath as String) {
12             let db = FMDatabase(path:databasePath as String)
13
14             if db == nil {
15                 NSLog("DB 생성 오류")
16             }
17
18             if ((db.open()) != nil) {
19                 let sql_statement = "Create table if not exists      test(id integer
20 primary key autoincrement, memo text)"

21                 if !(db.executeStatements(sql_statement)) {
22                     NSLog("테이블 생성 오류")
23                 }
24
25                 db.close()
26             } else {
27                 NSLog("디비 연결 오류")
28             }
29         }
```

```
30  }
31
32
```

5

6~8

10

12

19~20

Create table if not exists test(id integer primary key autocrement, memo text)

Create table : "테이블을 만들라"는 명령이다.

If not exists test : "test 라는 테이블이 존재하지 않으면" 이라는 조건문이다.

Id integer, memo text : 각 열(Column)의 이름과 타입을 지정한다.

Primary key : 해당 열을 프라이머리 키(주 키)로 지정한다. 관계형 데이터베이스에서 다른 테이블과의 관계를 정의할 때 사용되는 값이다.

autoincrement : 자동 증가 옵션을 사용하겠다는 의미이다, 이 옵션을 사용하면 새로운 열을 삽입할 때 따로 값을 넣지 않아도 자동으로 증가된 값을 할당받게 된다.

```
1   @IBAction func saveMemo(_ sender: UIButton) {
2           let db = FMDatabase(path:databasePath as String)
3
4           if ((db.open()) != nil) {
5               let insertQuery = "insert into test(memo) values(?)"
6               do{
7                   try db.executeUpdate(insertQuery, values: ["\(memo.text!)"])
8               }catch let error as NSError {
9                       print("failed: \(error.localizedDescription)")
10              }
11
12              if ((db.hadError()) != nil) {
13                  NSLog("저장 오류 \(insertQuery)")
14              } else {
15                  NSLog("저장 성공 \(insertQuery)")
16              }
17          } else {
18
```

```
                NSLog("디비 연결 오류")
            }
}
```

```
@IBAction func loadMemo(_ sender: UIButton) {
            let db = FMDatabase(path:databasePath as String)

            if ((db.open()) != nil) {
                let selectQuery = "select memo from test order by id desc limit 1"
                do {
                    let result:FMResultSet = try db.executeQuery(selectQuery,
values: [])
                    if result != nil{
                        while result.next() {
                            memo.text = result.string(forColumn: "memo")
                            NSLog("자료 불러오기 성공")
                        }
                    } else {
                        NSLog("자료 불러오기 실패")
                    }
                }catch let error as NSError {
                    print("failed: \(error.localizedDescription)")
                }

            } else {
                NSLog("디비 연결 오류")
            }
    }
```