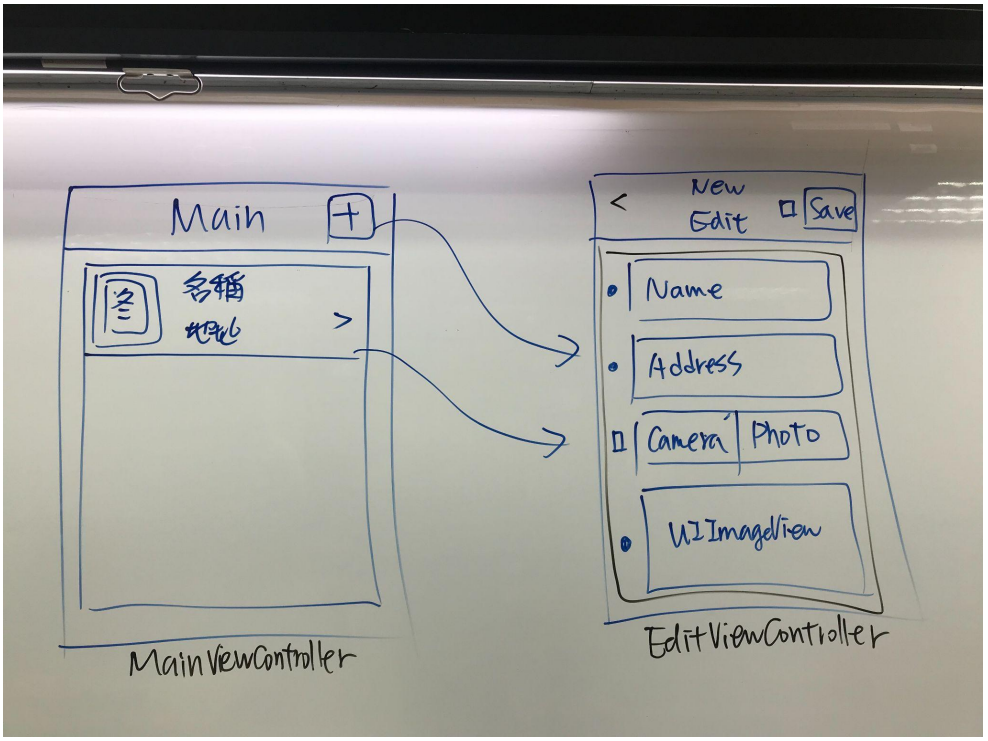
App開發步驟

1. UI/UX Flow design (Wireframe/Prototype/POC) => **UI Spec** => UI Components => Target-Action, Delegation
    a. iPhone or iPad
    b. Portrait or Landscape
    c. i18n?
    d. Data Analysis from UI & Design{DB Tables, Data Class(Entity)}(sqlite administration tool)



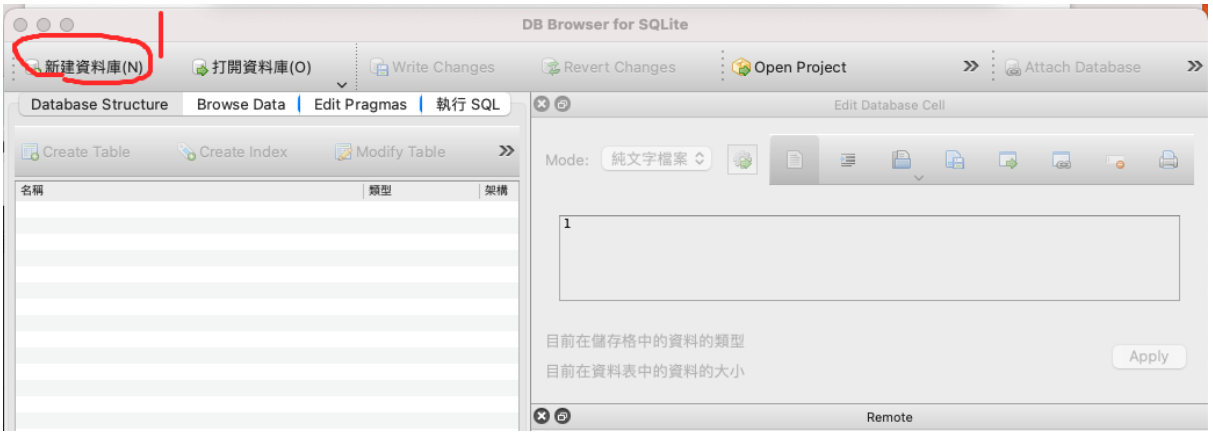
- Create Projects
- Create group: controller,model,fmdb
- Download resources and unzip
- Set up Library, Copy fmdb files to project, set up bridge file
- Database tables design

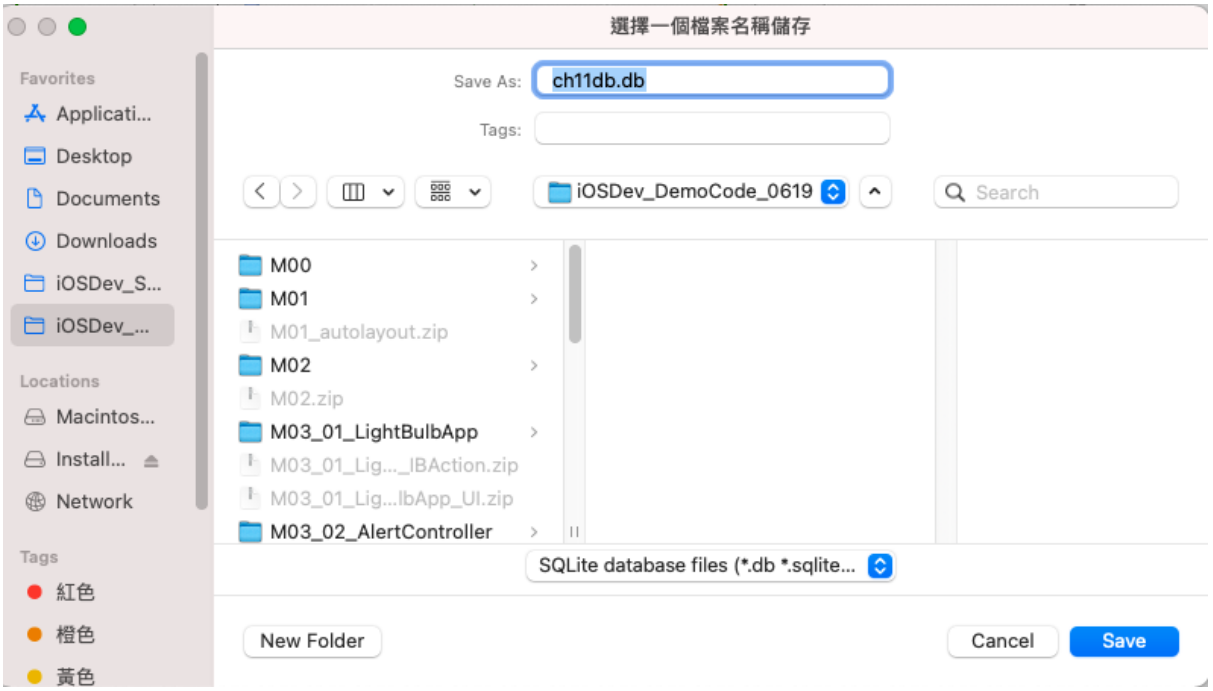
Restaurants

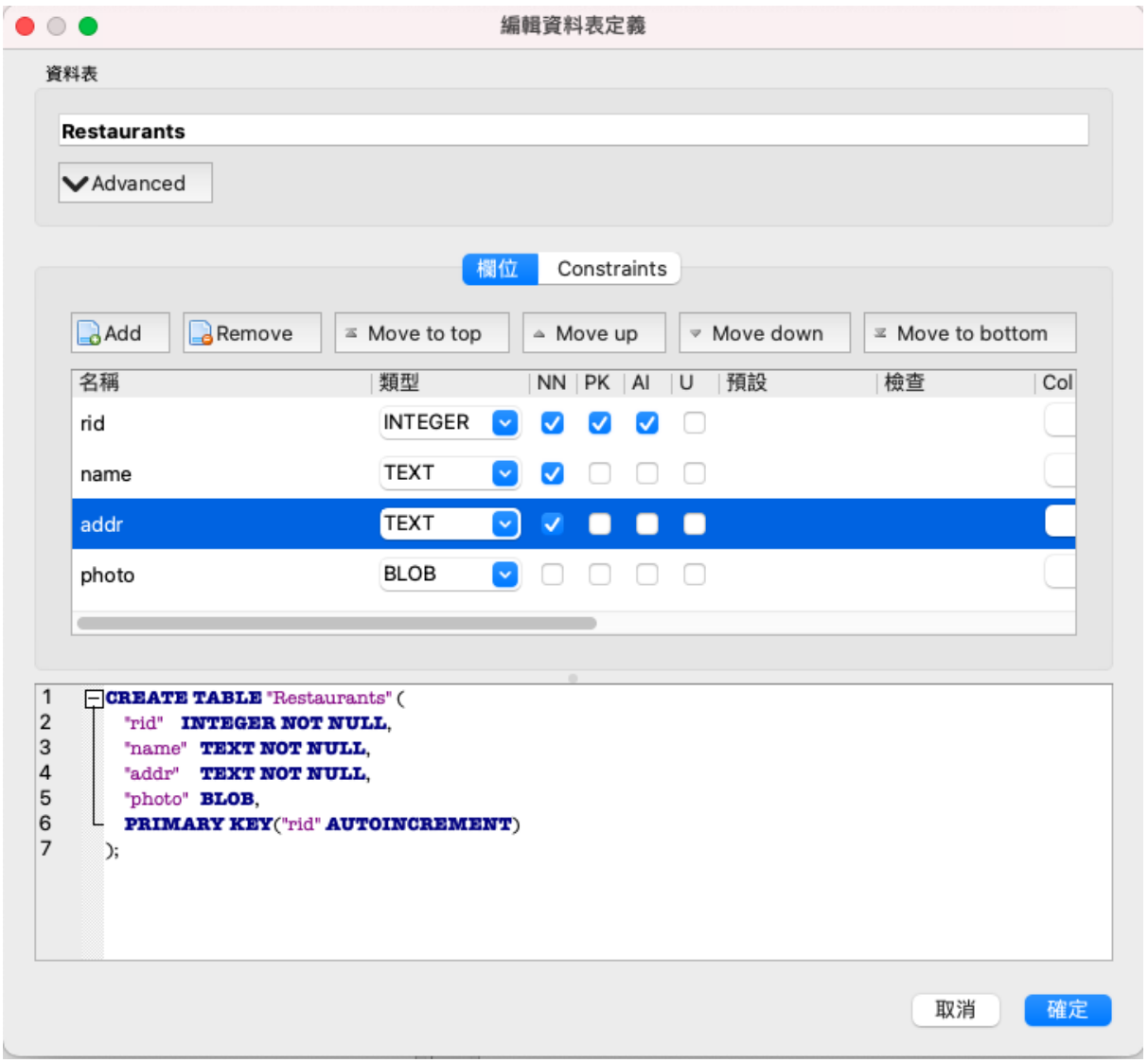
rid(自動遞增,PK)	name	addr	photo
Integer	Text	Text	Blob

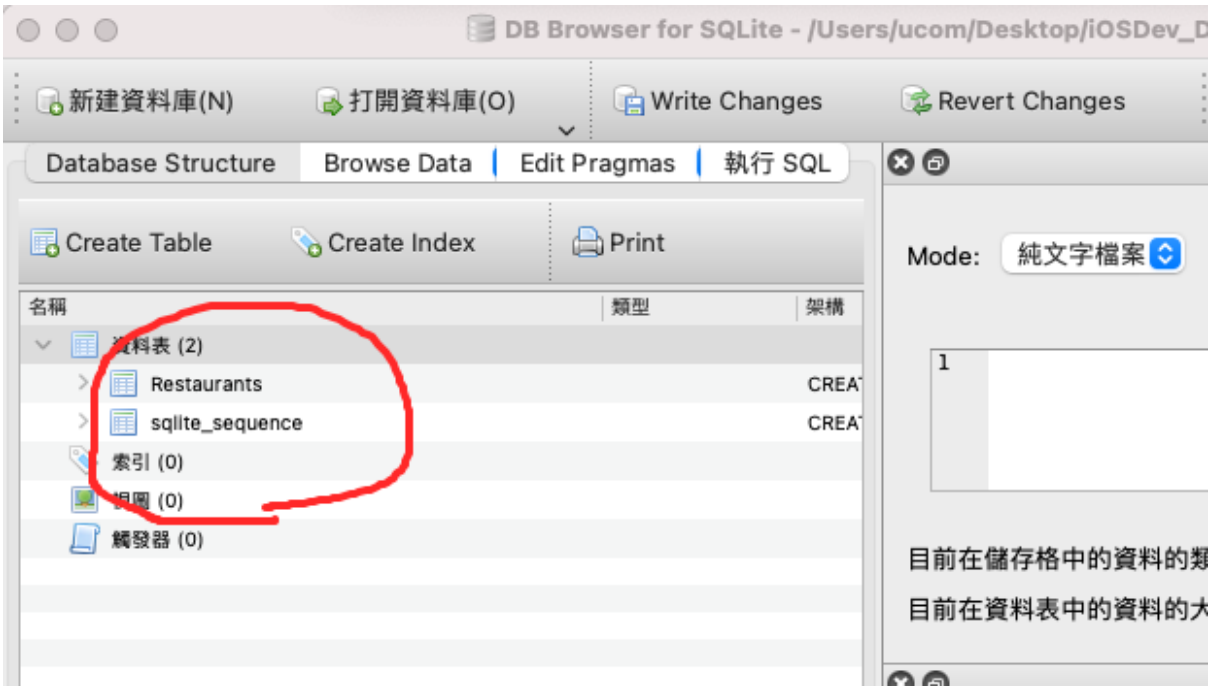
Use DB Browser

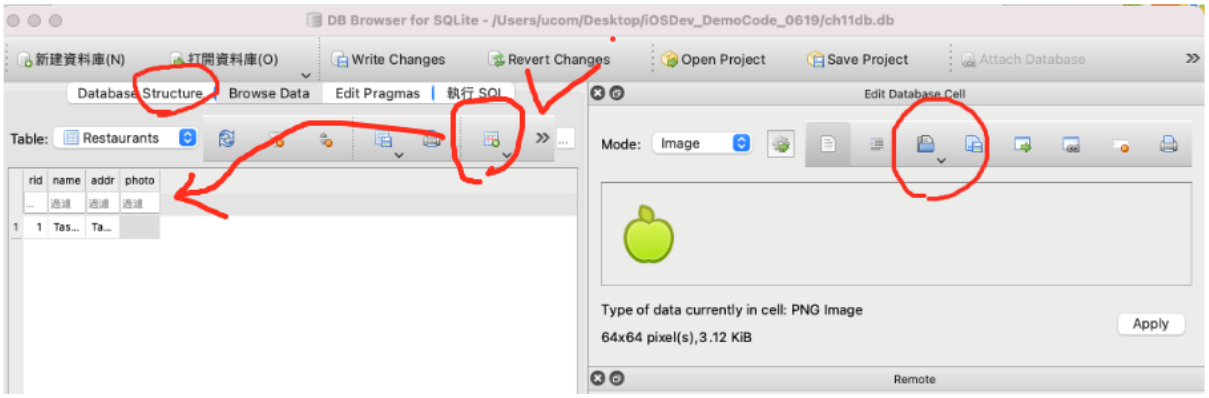
1. Create DB


DB Browser for SQLite
新建資料庫(N)
打開資料庫(O)
Write Changes
Revert Changes
Open Project
Attach Database
Database Structure
Browse Data
Edit Pragmas
執行 SQL
Edit Database Cell
Create Table
Create Index
Modify Table
名稱
類型
架構
Mode:
純文字檔案
1
目前在儲存格中的資料的類型
目前在資料表中的資料的大小
Apply
Remote

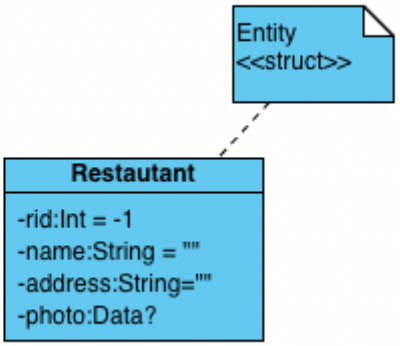

選擇一個檔案名稱儲存
Favorites
Applicati...
Desktop
Documents
Downloads
iOSDev_S...
iOSDev_...
Locations
Macintos...
Install...
Network
Tags
紅色
橙色
黃色
Save As:
ch11db.db
Tags:
iOSDev_DemoCode_0619
Search
M00
M01
M01_autolayout.zip
M02
M02.zip
M03_01_LightBulbApp
M03_01_Lig..._IBAction.zip
M03_01_Lig...lbApp_UI.zip
M03_02_AlertController
SQLite database files (*.db *.sqlite...
New Folder
Cancel
Save


編輯資料表定義
資料表
Restaurants
Advanced
欄位
Constraints
Add
Remove
Move to top
Move up
Move down
Move to bottom
名稱
類型
NN
PK
AI
U
預設
檢查
Col
rid
INTEGER
name
TEXT
addr
TEXT
photo
BLOB
1 CREATE TABLE "Restaurants" (
2 "rid" INTEGER NOT NULL,
3 "name" TEXT NOT NULL,
4 "addr" TEXT NOT NULL,
5 "photo" BLOB,
6 PRIMARY KEY("rid" AUTOINCREMENT)
7);
取消
確定


DB Browser for SQLite - /Users/ucom/Desktop/iOSDev_D
新建資料庫(N)
打開資料庫(O)
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
執行 SQL
Create Table
Create Index
Print
Mode:
純文字檔案
名稱
類型
架構
資料表 (2)
Restaurants
CREA
sqlite_sequence
CREA
索引 (0)
視圖 (0)
觸發器 (0)
1
目前在儲存格中的資料的類
目前在資料表中的資料的大



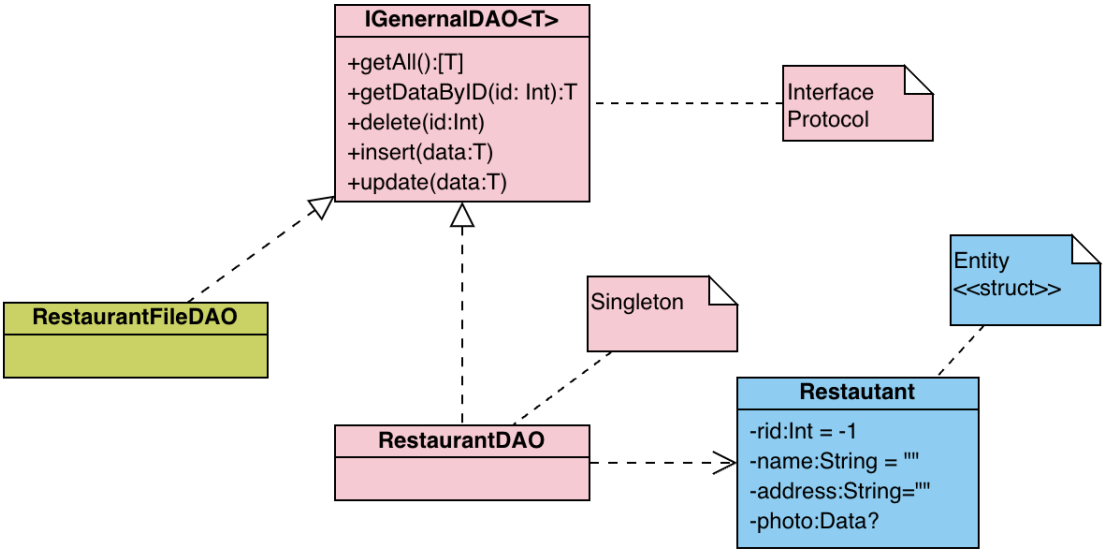
Design and Create Entity(Data class)



Code:

```
struct Restaurant {
    var rid = -1
    var name = ""
    var address = ""
    var photo: Data?
}
```

Design and Create Data Access Object class



create IGenernalDAO<T>
Code:

```
protocol IGenernalDAO {
    associatedtype T
    func getAll() -> [T]
    func getDataByID(id: Int) -> T?
//    func getDataByName(text:String) -> [T]
    func delete(id: Int)
    func insert(data: T)
    func update(data: T)
}
```

create RestaurantDAO with Singleton design pattern
Code:

```
//Singleton
class RestaurantDAO {
    //Data Fields
    var dbPath = ""
    //Singleton
    private static var _inst = RestaurantDAO()
    public static var shared: RestaurantDAO{
        return _inst
    }
    private init(){

    }
```

```
}
```

implements copy logic of db file in initialization method

```
    private init(){
        dbPath = "\(NSHomeDirectory())/Documents/db.db"
        let fileMgr = FileManager.default
        if !fileMgr.fileExists(atPath: dbPath) {
            if let srcPath = Bundle.main.path(forResource: "ch11db",
ofType: "db"){
                print("Copy file\n\(dbPath)")
                try? fileMgr.copyItem(atPath: srcPath, toPath: dbPath)
            }
        }else{
            print("File exists")
        }
    }
```

implements getAll()

Code:

```
class RestaurantDAO: IGenernalDAO {
    //Database settings
    let TABLE_NAME = "Restaurants"
    let COLUMN_RID = "rid"
    let COLUMN_NAME = "name"
    let COLUMN_ADDR = "addr"
    let COLUMN_PHOTO = "photo"
    //CRUD Methods
    func getAll() -> [Restaurant] {
        var list = [Restaurant]()
        let db = FMDatabase(path: dbPath)
        db?.open()
        let sql = "SELECT * FROM \(TABLE_NAME)"
        if let result = db?.executeQuery(sql, withArgumentsIn: []){
            while result.next() {
                let rid = Int(result.int(forColumn: COLUMN_RID))
                let name = result.string(forColumn: COLUMN_NAME) ?? ""
                let address = result.string(forColumn: COLUMN_ADDR) ??
""
                let photo = result.data(forColumn: COLUMN_PHOTO)
```

```
            list.append(Restaurant(rid: rid, name: name, address:
address, photo: photo))
        }
        result.close()
    }
    db?.close()
    return list
}
```

implements insert(:Restaurant)
Code:
//Version1

```
func insert(data: Restaurant) {
    //Version1
    var dict = [String:Any]()
    dict["n"] = data.name
    dict["a"] = data.address
    dict["p"] = data.photo
    let db = FMDatabase(path: dbPath)
    db?.open()
    let sql = "INSERT INTO \(TABLE_NAME)
(\(COLUMN_NAME),\(COLUMN_ADDR),\(COLUMN_PHOTO)) VALUES
(:n,:a,:p)"
//      print(sql)
    db?.executeUpdate(sql, withParameterDictionary: dict)
    db?.close()
}
```

//Version2

```
func insert(data: Restaurant) {
    var dict = [String:Any]()
    dict["n"] = data.name
    dict["a"] = data.address
    dict["p"] = data.photo
    let sql = "INSERT INTO \(TABLE_NAME)
(\(COLUMN_NAME),\(COLUMN_ADDR),\(COLUMN_PHOTO)) VALUES
(:n,:a,:p)"
    updateDB(sql: sql, parameterDictionary: dict)
}
func updateDB(sql: String, parameterDictionary dict: [String:Any]){
    let db = FMDatabase(path: dbPath)
```

```
        db?.open()
        db?.executeUpdate(sql, withParameterDictionary: dict)
        db?.close()
    }
```

implements getDataByID()
Code:

```
func getDataByID(id: Int) -> Restaurant? {
        var ret: Restaurant?
        let db = FMDatabase(path: dbPath)
        db?.open()
        let sql = "SELECT * FROM \(TABLE_NAME) WHERE
\(COLUMN_RID) = ?"
        if let result = db?.executeQuery(sql, withArgumentsIn: [id]){
            if result.next() {
                let rid = Int(result.int(forColumn: COLUMN_RID))
                let name = result.string(forColumn: COLUMN_NAME) ?? ""
                let address = result.string(forColumn: COLUMN_ADDR) ??
""
                let photo = result.data(forColumn: COLUMN_PHOTO)
                ret = Restaurant(rid: rid, name: name, address: address,
photo: photo)
            }
            result.close()
        }
        db?.close()
        return ret
    }
```

implements update(),delete()
Code:

```
func update(data: Restaurant) {
        var dict = [String:Any]()
        dict["n"] = data.name
        dict["a"] = data.address
        dict["p"] = data.photo
        dict["id"] = data.rid
```

```
        let sql = "UPDATE \(TABLE_NAME) SET \(COLUMN_NAME)=
:n,\(COLUMN_ADDR)=:a,\(COLUMN_PHOTO)=:p WHERE
\(COLUMN_RID)=:id"
        print(sql)
        updateDB(sql: sql, parameterDictionary: dict)
    }
    func delete(id: Int) {
        let db = FMDatabase(path: dbPath)
        db?.open()
        let sql = "DELETE FROM \(TABLE_NAME) WHERE rid = ?"
        db?.executeUpdate(sql, withArgumentsIn: [id])
        db?.close()
    }
```

WKWebView取代UIWebView

```
if let url = URL(string: "https://www.uuu.com.tw") {
            webView.load(URLRequest(url: url))
        }
```

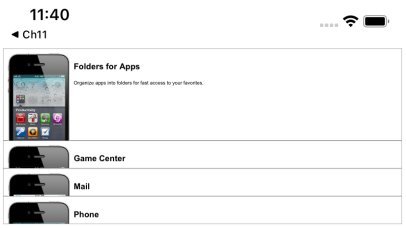
Executing JavaScript

```
func evaluateJavaScript(String, completionHandler: ((Any?, Error?)
-> Void)?)
```

Evaluates the specified JavaScript string.

Delegation

```
var uiDelegate: WKUIDelegate?
var navigationDelegate: WKNavigationDelegate?
```



Folders Game Mail Phone