Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình iOS

Bài 1. *Quản lý dữ liệu ứng dụng với SQLite*

Ngành Mạng & Thiết bị di động

2015

Nội dung

1. Giới thiệu SQLite

- SQLite
- Ưu điểm và hạn chế của SQLite
- SQLite Manager cho firefox
- Cấu hình ứng dụng để tương tác với SQLite
- Các hàm cơ bản trong SQLite

2. Các thao tác quản lý dữ liệu trong SQLite

3. Sắp xếp dữ liệu

1.1 SQLite

- SQLite là hệ thống cơ sở dữ liệu quan hệ nhỏ gọn, hoàn chỉnh, có thể cài đặt bên trong các trình ứng dụng khác.

- SQLite được Richard Hipp viết dưới dạng thư viện bằng ngôn ngữ lập trình C. Việc quản lý SQLite rất đơn giản, bạn chỉ cần quản lý thông qua một plugin của FireFox là SQLite Manager.

1.2 Ưu điểm và hạn chế của SQLite

- **Ưu điểm:**
  - Đảm bảo đầy đủ 4 đặc tính ACID của các giao tác.
  - Không cần cấu hình.
  - SQLite có gần như toàn bộ các đặc tính phổ biến của SQL theo chuẩn SQL92.
  - Toàn bộ Database được lưu trữ trong 1 tập tin trên đĩa duy nhất.
  - Hỗ trợ CSDL lên tới hằng TetraByte.
  - Bộ thư viện quản lý rất nhỏ, gọn.
  - Chạy nhanh hơn.
  - Đơn giản và dễ sử dụng.
  - Mã nguồn mở.

1.2 Ưu điểm và hạn chế của SQLite

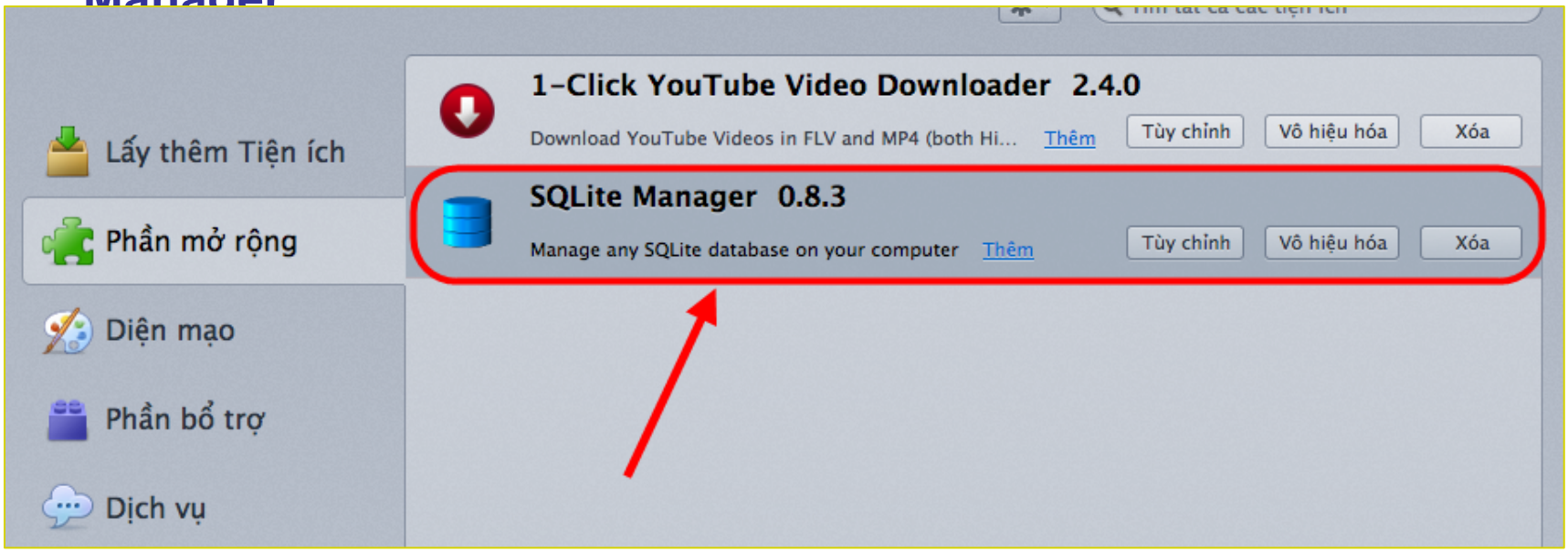
- **Ưu điểm:**
  - SQLite có thể được tải về và nhúng vào các dự án khác nhau dưới hình thức một Single ANSI-C source-code file.
  - Tự tổ chức lưu trữ (self-contained) nhúng vào các thiết bị di động mà không cần phải điều chỉnh cấu hình hệ thống.
  - Client đơn giản giao tiếp theo chế độ dòng lệnh (Command-Line Interface – CLI).

1.2 Ưu điểm và hạn chế của SQLite

- **Hạn chế:**
  - Tính đồng thời.
  - Nối kết mạng.
  - Phù hợp với các ứng dụng có qui mô dữ liệu nhỏ.

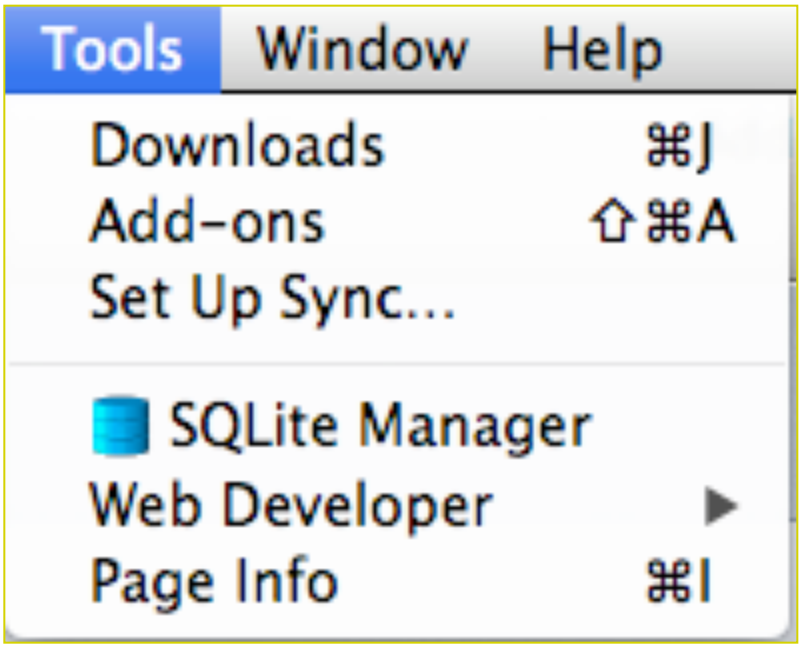
1.3 SQLite Manager cho firefox

- Việc quản lý SQLite rất đơn giản, để thuận tiện nhất bạn chỉ cần quản lý thông qua một plugin của FireFox là SQLite Manager



1.3 SQLite Manager cho firefox

- Sau khi cài đặt xong, trong phần Tool bạn sẽ thấy như hình.



1.4 Cấu hình ứng dụng để tương tác với SQLite

- Để ứng dụng có thể thao tác với cơ sở dữ liệu của SQLite, bạn cần bổ sung thêm thư viện hỗ trợ vào project.

- Trong phần Build Phase, mục Link to Library, bạn thêm vào thư viện libsqlite3.dylib vào project.

1.5 Các hàm cơ bản trong SQLite

- **Trong SQLite có một số hàm cơ bản cho phép bạn tương tác dễ dàng với cơ sở dữ liệu.**
  - sqlite3_open()
  - sqlite3_close()
  - sqlite3_prepare_v2()
  - sqlite3_step()
  - sqlite3_column_<type>()
  - sqlite3_finallize()

Nội dung

1. Giới thiệu SQLite

2. Các thao tác quản lý dữ liệu trong SQLite
   - Khởi tạo đối tượng SQLite
   - Kết nối hoặc tạo Database
   - Tạo bảng
   - Thêm dòng dữ liệu
   - Đọc dữ liệu
   - Đóng kết nối Database

3. Sắp xếp dữ liệu

2.1 Khởi tạo đối tượng SQLite

- Đối tượng SQLite là một đối tượng có kiểu dữ liệu là sqlite3. Đối tượng này đại diện cho database. Ta khai báo đối tượng này tại tập tin ViewController.h như sau:

```
#import <UIKit/UIKit.h>
#import <sqlite3.h>
@interface ViewController : UIViewController
@property (nonatomic) sqlite3 *database;
@end
```

2.2 Kết nối hoặc tạo Database

- Dùng hàm sqlite3_open() để mở kết nối đến cơ sở dữ liệu sqlite.

```
NSString *databaseDir =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES)[0];

NSString *filePath = [databaseDir
stringByAppendingPathComponent: @"database.sqlite"];

if(sqlite3_open([filePath UTF8String], &_database)==SQLITE_OK){

    NSLog(@"Kết nối database thành công");

    sqlite3_close(_database);

}else{

    NSLog(@"Kết nối thất bại!");

}
```

2.3 Tạo bảng

- Câu truy vấn để tạo bảng được viết theo dạng:

```
CREATE TABLE database_name.table_name(
   column1 datatype  PRIMARY KEY(one or more columns),
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
);
```

- Sau khi tạo xong câu truy vấn, ta có thể sử dụng hàm sqlite3_exec() để thực thi câu truy vấn

```
sqlite3_exec(<database>, <query_stmt>, NULL, NULL, <error_message>)
```

2.3 Tạo bảng

- Ví dụ viết phương thức *createTableNamed* dùng để tạo bảng:

```
-(void) createTableNamed:(NSString *) tableName
              withField1:(NSString *) field1
              withField2:(NSString *) field2 {

    char *err;
    NSString *sql = [NSString stringWithFormat:
                     @"CREATE TABLE IF NOT EXISTS '%@' ('%@' "
                     "TEXT PRIMARY KEY, '%@' TEXT);",
                     tableName, field1, field2];

    if (sqlite3_exec(db, [sql UTF8String], NULL, NULL, &err)
        != SQLITE_OK) {
        sqlite3_close(db);
        NSAssert(0, @"Tabled failed to create.");
    }
}
```

2.4 Thêm dòng dữ liệu

- Câu truy vấn để tạo bảng được viết theo dạng:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

- Sau khi tạo xong câu truy vấn, ta có thể sử dụng hàm sqlite3_exec() để thực thi câu truy vấn giống tạo bảng.

2.4 Thêm dòng dữ liệu

- Ví dụ viết phương thức insertRecordIntoTableNamed dùng để thêm dữ liệu vào bảng:

```
-(void) insertRecordIntoTableNamed:(NSString *) tableName
                         withField1:(NSString *) field1
                        field1Value:(NSString *) field1Value
                          andField2:(NSString *) field2
                        field2Value:(NSString *) field2Value {

    NSString *sql = [NSString stringWithFormat:
                     @"INSERT OR REPLACE INTO '%@' ('%@', '%@') "
                     "VALUES ('%@','%@')", tableName, field1, field2,
                     field1Value, field2Value];

    char *err;
    if (sqlite3_exec(db, [sql UTF8String], NULL, NULL, &err)
        != SQLITE_OK) {
        sqlite3_close(db);
        NSAssert(0, @"Error updating table.");
    }
}
```

2.4 Thêm dòng dữ liệu

- Ngoài ra ta còn cách sử dụng Bind Variables để tạo câu truy vấn.

- Tạo câu truy vấn:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (?,?,?,...);
```

- Sử dụng phương thức:
  - sqlite3_bind_<value>(đối số 1, đối số 2, …)
  - Ví dụ:
    - sqlite3_bind_text(statement, 1, [<chuỗi cần truyền> UTF8String], -1, NULL);

2.4 Thêm dòng dữ liệu

- Ví dụ:

```
NSString *sqlStr = [NSString stringWithFormat:
                    @"INSERT OR REPLACE INTO '%@' ('%@', '%@') "
                    "VALUES (?,?)", tableName, field1, field2];
const char *sql = [sqlStr UTF8String];


sqlite3_stmt *statement;
if (sqlite3_prepare_v2(db, sql, -1, &statement, nil) == SQLITE_OK) {
    sqlite3_bind_text(statement, 1, [field1Value UTF8String],
                      -1, NULL);
    sqlite3_bind_text(statement, 2, [field2Value UTF8String],
                      -1, NULL);
}


if (sqlite3_step(statement) != SQLITE_DONE)
    NSAssert(0, @"Error updating table.");
sqlite3_finalize(statement);
```

2.5 Đọc dữ liệu

- Tạo câu truy vấn:

```
SELECT column_name, column_name FROM table_name
```

- Nếu ta muốn lấy ra dữ liệu của tất cả các cột ta có thể dùng "*"

```
SELECT * FROM table_name
```

- Nếu có điều kiện lọc:

```
SELECT * FROM table_name WHERE column_name operator value
```

2.5 Đọc dữ liệu

- **Để thực thi câu truy vấn và đọc dữ liệu ta cần sử dụng phương thức:**
  - sqlite3_prepare_v2(<database>, <query_stmt>, -1, <statement>, Nil)
  - sqlite3_step(<statement>)
  - sqlite3_colum_<kiểu dữ liệu>

2.5 Đọc dữ liệu

- Ví dụ:

```
- (NSMutableString *)getAllNumber{
    NSMutableString *result = [NSMutableString stringWithString:@"Lay du lieu loi"];

    NSString *query = @"Select * from DemSo";
    const char *query_stmt = [query UTF8String];
    if ([self Open]) {
        if (sqlite3_prepare_v2(database, query_stmt, -1, &statement, Nil) == SQLITE_OK) {
            result = [NSMutableString stringWithString:@"AllRow: "];
            while (sqlite3_step(statement) == SQLITE_ROW) {
                char *so = (char *) sqlite3_column_text(statement, 0);
                NSString *soStr = [NSString stringWithUTF8String:so];
                char *dem = (char *) sqlite3_column_text(statement, 1);
                NSString *demStr = [NSString stringWithUTF8String:dem];

                [result appendString:[NSString stringWithFormat:@"%@-%@ ",soStr,demStr]];
            }
        }
        [self Close];
        return result;
    }
    return result;
}
```

2.6 Đóng kết nối Database

- **Sau khi hoàn tất quá trình tương tác cơ sở dữ liệu:**
  - Giải phóng câu truy vấn bằng hàm sqlite3_finalize(statement) nếu có sử dụng đối tượng statement.
  - Đóng kết nối cơ sở dữ liệu bằng hàm sqlite3_close(_database).

Nội dung

1. Giới thiệu SQLite

2. Các thao tác quản lý dữ liệu trong SQLite

3. **Sắp xếp dữ liệu**
   - Giới thiệu Sort Descriptors
   - Khảo sát lớp NSSortDescriptor
   - Cách sử dụng NSSortDescriptor

3.1 Giới thiệu Sort Descriptors

- Sort Descriptors mô tả các phương thức so sánh nhằm sắp xếp các dữ liệu cho trước.

- Lưu ý rằng Sort Descriptors không phải là một đối tượng sắp xếp mà nó cung cấp cho ta các cách thức, mô tả để có thể sắp xếp dữ liệu.

3.2 Khảo sát lớp NSSortDescriptor

- **Khởi tạo NSSortDescriptor**
  - sortDescriptorWithKey:ascending:
  - initWithKey:ascending:
  - sortDescriptorWithKey:ascending:selector:
  - initWithKey:ascending:selector:
  - sortDescriptorWithKey:ascending:comparator:
  - initWithKey:ascending:comparator:

3.2 Khảo sát lớp NSSortDescriptor

- **Lấy thông tin về Sort Descriptor**
  - ascending
  - key
  - selector

3.2 Khảo sát lớp NSSortDescriptor

- **Sử dụng Sort Descriptor**
  - compareObject:toObject:
  - reversedSortDescriptor
  - allowEvaluation

3.2 Khảo sát lớp NSSortDescriptor

- **Tạo NSComparator cho Sort Descriptor**
  - comparator

3.3 Cách sử dụng Sort Descriptor

- **Ví dụ: ta có một mảng *Employee* với các thuộc tính sau:**
  - NSString - Fist name
  - NSString - Last name
  - NSDate - DateOfHire
  - NSNumber - Age

3.3 Cách sử dụng Sort Descriptor

- Ta tạo một mảng gồm nhiều phần tử *Employee*:

```
NSDate *date1 = [NSDate new]; // ngay hien tai
NSDate *date2 = [NSDate dateWithTimeIntervalSinceNow:-60*60*24];
NSDate *date3 = [NSDate dateWithTimeIntervalSinceNow:-60*60*24*3];
NSDate *date4 = [NSDate dateWithTimeIntervalSinceNow:-60*60*24*2];

Employee *em = [[Employee alloc] init];
[em setFirstName:@"Van" lastName:@"Tei" dateOfHire:date1 age:15];
Employee *em1 = [[Employee alloc] init];
[em1 setFirstName:@"Minh" lastName:@"Tomet" dateOfHire:date2 age:14];
Employee *em2 = [[Employee alloc] init];
[em2 setFirstName:@"Anh" lastName:@"Gole" dateOfHire:date3 age:14];
Employee *em3 = [[Employee alloc] init];
[em3 setFirstName:@"Tan" lastName:@"Join" dateOfHire:date4 age:15];

NSArray *employeesArray = [[NSArray alloc] initWithObjects:em, em1, em2, em3, nil];
```

3.3 Cách sử dụng Sort Descriptor

- **Ta tạo đối tượng NSSortDesciptor:**

```
NSSortDescriptor *sortKeyDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"<tên của thuộc tính muốn sắp xếp>" ascending:<sắp xếp
tăng hay giảm>];
```

Ví dụ:

```
NSSortDescriptor *ageDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"age" ascending:YES];
```

3.3 Cách sử dụng Sort Descriptor

- Dùng NSSortDesciptor để sắp xếp:

```
NSArray *sortDescriptors = @[ageDescriptor];
NSArray *sortedArray = [employeesArray sortedArrayUsingDescriptors:sortDescriptors];
for (Employee *eml in sortedArray) {
    NSLog(@"%@",eml.age);
}

sortDescriptors = @[ageDescriptor, fNameDescriptor];
sortedArray = [employeesArray sortedArrayUsingDescriptors:sortDescriptors];
for (Employee *eml in sortedArray) {
    NSLog(@"%@ - %@ - %@ ",eml.firstName, eml.age, eml.lastName);
}
```

Thảo luận