



LẬP TRÌNH iOS

Module 3

☞ Click vào phụ lục để chuyển tới bài cần đọc

Phụ lục

Bài 1 Quản lý dữ liệu ứng dụng với SQLite	2
Bài 2 Truy vấn và tìm kiếm dữ liệu cho ứng dụng	13
Bài 3 Core Data Framework.....	24
Bài 4 Tiến trình và xử lý đa tiến trình	38
Bài 5 Core Animation.....	49
Bài 6 Touch & Gesture.....	62
Bài 7 Search Field	73
Bài 8 Split View Controller.....	86
Bài 9 Storyboard.....	95



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

[Go Screen Capture](#)

Lập trình iOS

Bài 1. Quản lý dữ liệu ứng dụng với SQLite

Ngành Mạng & Thiết bị di động



2014

5014



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. Ví dụ minh họa



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

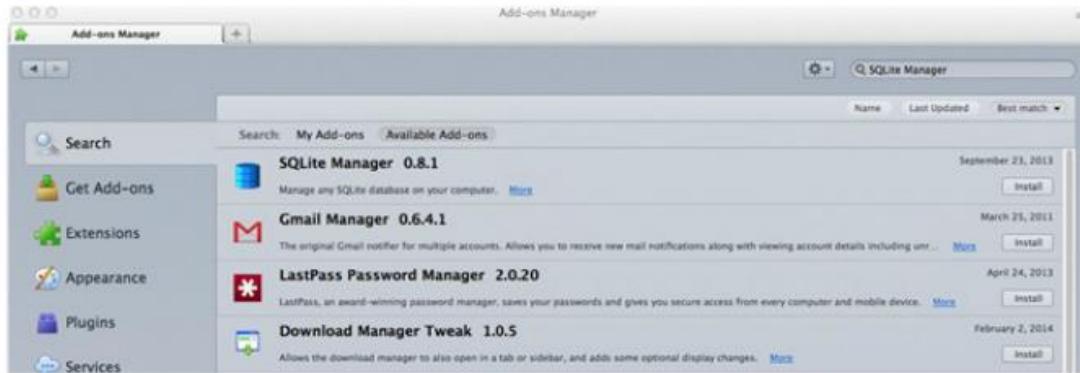
❑ Việc quản lý SQLite rất đơn giản, dễ thuận tiện nhất bạn chỉ cần quản lý thông qua một plugin của FireFox là `SQLite Manager.



1.3 SQLite Manager cho firefox



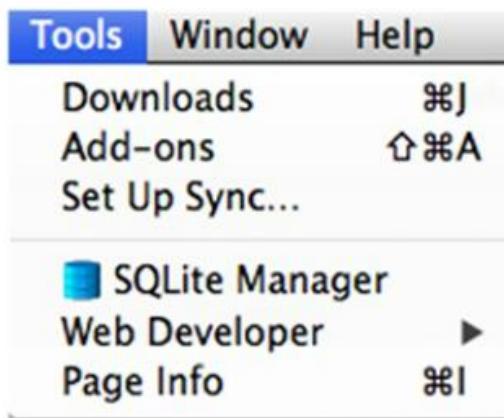
- Để cài đặt plugin quản lý SQLite cho Firefox, bạn vào phần Add-on của trình duyệt Firefox và tìm plugin SQLite Manager rồi Add to Firefox.



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 libssqlite3.dylib vào project.

Name	Status
 libssqlite3.dylib	Required ▾
 CoreGraphics.framework	Required ▾
 UIKit.framework	Required ▾
 Foundation.framework	Required ▾
+ -	



1.5 Các hàm cơ bản trong SQLite

- ❑ Trong SQLite có một số hàm cơ bản cho phép bạn thực hiện các lệnh:
 - sqlite3_open()
 - sqlite3_close()
 - sqlite3_prepare_v2()
 - sqlite3_step(): thực thi lệnh truy vấn được tạo bởi hàm sqlite3_prepare_v2().
 - sqlite3_column_<type>()
 - sqlite3_finalize(): xoá câu lệnh truy vấn SQL được khởi tạo bởi hàm sqlite3_prepare_v2() trong bộ nhớ.



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
 - Khởi tạo và thực thi lệnh truy vấn
 - Truy xuất dữ liệu Database
 - Đóng kết nối Database
3. Ví dụ minh họa



2.1 Khởi tạo đối tượng SQLite



- Trước khi thực hiện các thao tác với cơ sở dữ liệu SQLite, bạn cần phải tạo đối tượng dạng này bằng cách khai báo một biến có kiểu sqlite3 trong tập tin .h .
- Khai báo sqlite3 và dataPath.

```
sqlite3 *contactDB;
```

```
NSString *dataPath;
```





2.2 Kết nối hoặc tạo Database

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

```
const char *dbpath = [dataPath UTF8String];  
  
if (sqlite3_open(dbpath, &contactDB) == SQLITE_OK) {  
  
....  
  
}
```



2.3 Khởi tạo và thực thi lệnh truy vấn

- ❑ Câu lệnh truy vấn được khởi tạo và lưu trữ vào đối tượng `sqlite_stmt()` và truyền vào hàm `sqlite_prepare_v2()` để thực thi

```
sqlite3_stmt *statement;  
  
const char *query_stmt = [query UTF8String];  
  
if(sqlite3_prepare_v2(contactDB, query_stmt, -1, &statement, NULL)  
==SQLITE_OK) {  
  
if (sqlite3_step(statement)==SQLITE_ROW) {  
  
dem = [[NSString alloc] initWithUTF8String:  
       (const char *) sqlite3_column_text(statement, 0)];  
  
} else{  
  
NSLog(@"No data");  
  
}  
  
}
```





2.4 Truy xuất dữ liệu Database

- ❑ Khởi tạo câu lệnh truy vấn.

```
NSString *query = [NSString stringWithFormat:@"SELECT Dem FROM  
DemSo where So = '%@'", self.tfSo.text];  
const char *query_stmt = [query UTF8String];
```



2.4 Truy xuất dữ liệu Database

- ❑ Xử lý dữ liệu truy vấn được.

```
if (sqlite3_step(statement)==SQLITE_ROW) {  
    dem = [[NSString alloc] initWithUTF8String:  
        (const char *) sqlite3_column_text(statement, 0)];  
}
```





2.5 Đóng kết nối Database

- Sau khi hoàn tất quá trình tuân thủ tác cơ sở dữ liệu, bạn nên đóng lại kết nối cơ sở dữ liệu.

```
sqlite3_finalize(statement);
```



3 Ví dụ minh họa



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 2. *Truy vấn và tìm kiếm dữ liệu cho ứng dụng*

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



1. Các thao tác truy vấn dữ liệu

- Tạo bảng
- Thêm dòng dữ liệu
- Bind Variables
- Đọc dữ liệu

2. Sắp xếp dữ liệu



1.1 Tạo bảng



- ❑ Khi cơ sở dữ liệu sqlite đã được tạo, ta có thể thực hiện các thao tác thêm bảng, trường dữ liệu cho bảng...
- ❑ **Tạo câu truy vấn:**
 - CREATE TABLE '**Tên bảng**' ('<Cột 1>' TEXT PRIMARY KEY NOT NULL , '<Cột 2>' TEXT)
- ❑ **Sử dụng phương thức:**
 - `sqlite3_exec(<database>, <query_stmt>, NULL, NULL, <error_message>)`





1.1 Tạo bảng

□ Ví dụ:

```
- (BOOL)CreateTableName:(NSString *)pTableName  
    withField1:(NSString *)pField1 withField2:(NSString *  
    )pField2{  
    NSString *query = [NSString stringWithFormat:@"CREATE  
        TABLE '%@' (%@\\" TEXT PRIMARY KEY NOT NULL ,  
        \"%@\\" TEXT)", pTableName, pField1, pField2];  
    const char *query_stmt = [query UTF8String];  
    char *err;  
    [self Open]; // tạo phương thức open để mở database  
    if (sqlite3_exec(database, query_stmt, NULL, NULL, &  
        err) != SQLITE_OK) {  
        return false;  
    }  
    [self Close]; // tạo phương thức close để đóng  
    database  
    return true;  
}
```



1.2 Thêm dòng dữ liệu

□ Sau khi tạo bảng dữ liệu đã hoàn tất, ta có thể thêm các dòng dữ liệu vào đó.

□ Tạo câu truy vấn:

- INSERT INTO '<Tên bảng>' ('<tên cột 1>', '<tên cột 2>', ...) VALUES ('<Giá trị cột 1>', '<Giá trị cột 2>', ...)

□ Sử dụng phương thức:

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



1.2 Thêm dòng dữ liệu



□ Ví dụ:

```
- (BOOL)InsertRowWithSo:(NSInteger)pSo withDem:(NSString *)pDem{
    NSString *query = [NSString stringWithFormat:@"insert into
        DemSo (So,Dem) values (%d,%@)", pSo, pDem];
    const char *query_stmt = [query UTF8String];
    char *err;
    if ([self Open]) {
        if (sqlite3_exec(database, query_stmt, NULL, NULL, &err)
            == SQLITE_OK) {
            NSLog(@"Da thêm dòng.");
            [self Close];
            return true;
        }
        [self Close];
        NSLog(@"Không thể thêm dữ liệu");
    }else{
        NSLog(@"Không thể mở database");
    }
    return false;
}
```



1.3 Bind Variable



□ Ta sử dụng Bind Variable để truyền dữ liệu vào câu truy vấn

□ Tạo câu truy vấn:

- INSERT OR REPLACE INTO '<Tên bảng>'
(<tên cột 1>,'<tên cột 2>', ...) 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);





1.4 Đọc dữ liệu

- ❑ Để đọc tất cả dữ liệu trong bảng ta làm như sau.
- ❑ Tạo câu truy vấn:
 - `SELECT * FROM <Tên bảng>`
- ❑ Sử dụng phương thức:
 - `sqlite3_prepare_v2(<database>, <query_stmt>, -1, <statement>, Nil)`
 - `sqlite3_step(<statement>)`



1.4 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 stringWithFormat:@"AllRow: %d", sqlite3_step(statement)];
            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;
}
```



Nội dung



1. Các thao tác truy vấn dữ liệu
2. Sắp xếp dữ liệu
 - Giới thiệu Sort Descriptors
 - Khảo sát lớp NSSortDescriptor
 - Cách sử dụng NSSortDescriptor



2.1 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.





2.2 Khảo sát lớp NSSortDescriptor

❑ Khởi tạo NSSortDescriptor

- [sortDescriptorWithKey:ascending:](#)
- [initWithKey:ascending:](#)
- [sortDescriptorWithKey:ascending:selector:](#)
- [initWithKey:ascending:selector:](#)
- [sortDescriptorWithKey:ascending:comparator:](#)
- [initWithKey:ascending:comparator:](#)



2.2 Khảo sát lớp NSSortDescriptor

❑ Lấy thông tin về Sort Descriptor

- [ascending](#)
- [key](#)
- [selector](#)





2.2 Khảo sát lớp NSSortDescriptor

Sử dụng Sort Descriptor

- [compareObject:toObject:](#)
- [reversedSortDescriptor](#)
- [allowEvaluation](#)



2.2 Khảo sát lớp NSSortDescriptor

Tạo NSComparator cho Sort Descriptor

- [comparator](#)





2.3 Cách sử dụng Sort Descriptor

- Ví dụ: ta có một mảng *Employee* với các thuộc tính sau:

- NSString - First name
- NSString - Last name
- NSDate - DateOfHire
- NSNumber - Age



2.3 Cách sử dụng Sort Descriptor

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

```
NSDate *date1 = [NSDate new]; // ngay hiện tại
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];
```



2.3 Cách sử dụng Sort Descriptor



- Ta tạo đối tượng NSSortDescriptor:

```
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];
```



2.3 Cách sử dụng Sort Descriptor



- Dùng NSSortDescriptor 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





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

[Go Screen Capture](#)

Lập trình iOS

Bài 3. Core Data Framework

Ngành Mạng & Thiết bị di động



2014

2014



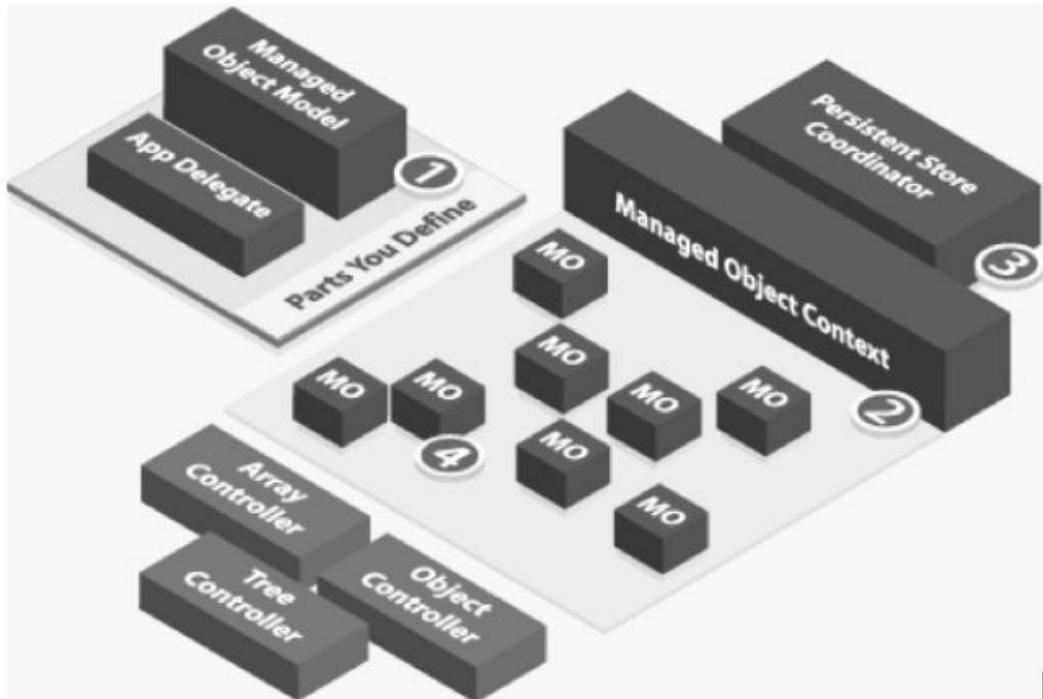
Nội dung



1. Giới thiệu
 - Tổng quan
 - Các tính năng
2. Tích hợp Core Data Framework
3. Model entity
4. Tạo các lớp đối tượng
5. Tương tác với dữ liệu dùng Core Data Manage Context



1 Giới thiệu Core Data Framework



1.1 Tổng quan



- Core Data Framework là một trong những lựa chọn tối ưu cho việc lưu trữ trong ứng dụng, đặc biệt là các ứng dụng cỡ trung và lớn.
- Cung cấp một công cụ trực quan để thiết kế cơ sở dữ liệu cùng các quan hệ (Data Relationship)
- Tiết kiệm đáng kể thời gian viết code.
- Giúp giảm đáng kể tài nguyên vùng nhớ (Memory) bởi khả năng thu hồi bộ nhớ.



1.2 Các tính năng



	Core Data	NSKeyedArchiver
Entity Modeling	Yes	No
Querying	Yes	No
Speed	Fast	Slow
Serialization Format	SQLite, XML, or NSData	NSData
Migrations	Automatic	Manual
Undo Manager	Automatic	Manual

Bảng so sánh Core Data và NSKeyedArchiver





1.2 Các tính năng

❑ Tính năng:

- Cho phép tương tác, xử lý lưu trữ đối tượng tối ưu.
- Cung cấp phương thức cho việc quản lý đối tượng cũng như việc quản lý Undo-Redo.
- Tối ưu hoá bộ nhớ ứng dụng.
- Hỗ trợ việc lưu trữ theo version và tích hợp dữ liệu khi cần thiết.



Nội dung

1. Giới thiệu
2. Tích hợp Core Data Framework
 - Tích hợp trong lúc tạo project
 - Tích hợp vào project có sẵn
3. Model entity
4. Tạo các lớp đối tượng
5. Tương tác với dữ liệu dùng Core Data Manage Context





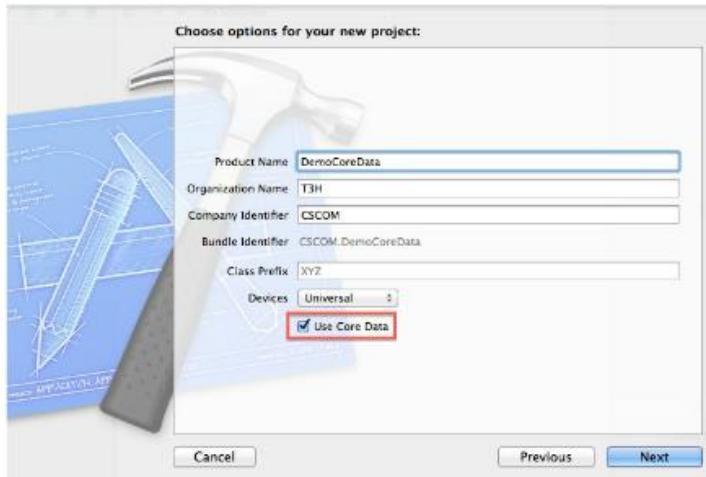
2 Tích hợp Core Data Framework

- ❑ Ta có thể tích hợp Core Data Framework vào dự án theo 2 cách sau:
 - Tích hợp trong lúc tạo project.
 - Tích hợp vào project có sẵn.



2.1 Tích hợp trong lúc tạo project

- ❑ Tích hợp trong lúc tạo project tương đối đơn giản. Khi tạo project ta chỉ việc check vào dòng use core data hệ thống sẽ tự tạo những thông tin cần thiết.





2.1 Tích hợp vào project có sẵn

□ Để tích hợp và project có sẵn ta thực hiện những bước sau.

- Bước 1: Thêm Framework Core Data vào project.
- Bước 2: Tạo Database Model.
- Bước 3: Thêm các đoạn mã cần thiết sau:

▪ Trong tập tin ..Prefix.pch

```
#import <CoreData/CoreData.h>
```



2.1 Tích hợp vào project có sẵn

▪ Trong tập tin AppDelegate.h:

```
@property (readonly, strong, nonatomic) NSManagedObjectContext  
*managedObjectContext;  
  
@property (readonly, strong, nonatomic) NSManagedObjectModel  
*managedObjectModel;  
  
@property (readonly, strong, nonatomic) NSPersistentStoreCoordinator  
*persistentStoreCoordinator;
```

```
- (void)saveContext;  
- (NSURL *)applicationDocumentsDirectory;
```





2.1 Tích hợp vào project có sẵn

- Trong AppDelegate.m:

```
@synthesize managedObjectContext = _managedObjectContext;
@synthesize managedObjectModel = _managedObjectModel;
@synthesize persistentStoreCoordinator = _persistentStoreCoordinator;

...
- (void)saveContext{
    NSError *error = nil;
    NSManagedObjectContext *managedObjectContext =
    self.managedObjectContext;
    if (managedObjectContext != nil) {
        if ([managedObjectContext hasChanges] && ![managedObjectContext
            save:&error]) {
            NSLog(@"Unresolved error %@", error, [error userInfo]);
            abort();
        }
    }
}
```



2.1 Tích hợp vào project có sẵn

- Trong AppDelegate.m:

```
- (NSManagedObjectContext *)managedObjectContext
{
    if (_managedObjectContext != nil) {
        return _managedObjectContext;
    }

    NSPersistentStoreCoordinator *coordinator = [self
        persistentStoreCoordinator];
    if (coordinator != nil) {
        _managedObjectContext = [[NSManagedObjectContext alloc] init];
        [_managedObjectContext setPersistentStoreCoordinator:coordinator];
    }
    return _managedObjectContext;
}
```





2.1 Tích hợp vào project có sẵn

- Trong AppDelegate.m:

```
- (NSManagedObjectModel *)managedObjectModel
{
    if (_managedObjectModel != nil) {
        return _managedObjectModel;
    }
    NSURL *modelURL = [[NSBundle mainBundle]
URLForResource:@"DemoCoreData" withExtension:@"momd"];
    _managedObjectModel = [[NSManagedObjectModel alloc]
initWithContentsOfURL:modelURL];
    return _managedObjectModel;
}
```



2.1 Tích hợp vào project có sẵn

- Trong AppDelegate.m:

```
- (NSPersistentStoreCoordinator *)persistentStoreCoordinator{
    if (_persistentStoreCoordinator != nil) {
        return _persistentStoreCoordinator;
    }
    NSURL *storeURL = [[self applicationDocumentsDirectory]
URLByAppendingPathComponent:@"DemoCoreData.sqlite"];
    NSError *error = nil;
    _persistentStoreCoordinator = [[NSPersistentStoreCoordinator alloc]
initWithManagedObjectModel:[self managedObjectModel]];
    if (![_persistentStoreCoordinator
addPersistentStoreWithType:NSSQLiteStoreType configuration:nil
URL:storeURL options:nil error:&error]) {
        NSLog(@"Unresolved error %@", error, [error userInfo]);
        abort();
    }
    return _persistentStoreCoordinator;
}
```





2.1 Tích hợp vào project có sẵn

- Trong AppDelegate.m:

```
- (NSURL *)applicationDocumentsDirectory
{
    return [[[NSFileManager defaultManager]
URLsForDirectory:NSDocumentDirectory inDomains:NSUserDomainMask]
lastObject];
}
```



Nội dung

1. Giới thiệu
2. Tích hợp Core Data Framework
3. Model entity
4. Tạo các lớp đối tượng
5. Tương tác với dữ liệu dùng Core Data Manage Context





3 Module Entity

Module Entity là đối tượng mô tả những thực thể mà ta cần lưu trữ.

Có thể hiểu đối tượng này giống như các bảng dữ liệu trong database.

- Để tạo một entity ta cần chọn tập tin *.xcdatamodelId sẽ xuất hiện màn hình để thao tác.
- Chọn “Add Entity” để thêm một entity.
- Trong Entity còn có các thuộc tính:
 - Attributes: lưu trữ các cột của bảng.
 - Relationships: để lưu trữ những liên kết giữa các bảng.



Nội dung

1. Giới thiệu
2. Tích hợp Core Data Framework
3. Model entity
4. Tạo các lớp đối tượng
5. Tương tác với dữ liệu dùng Core Data Manage Context



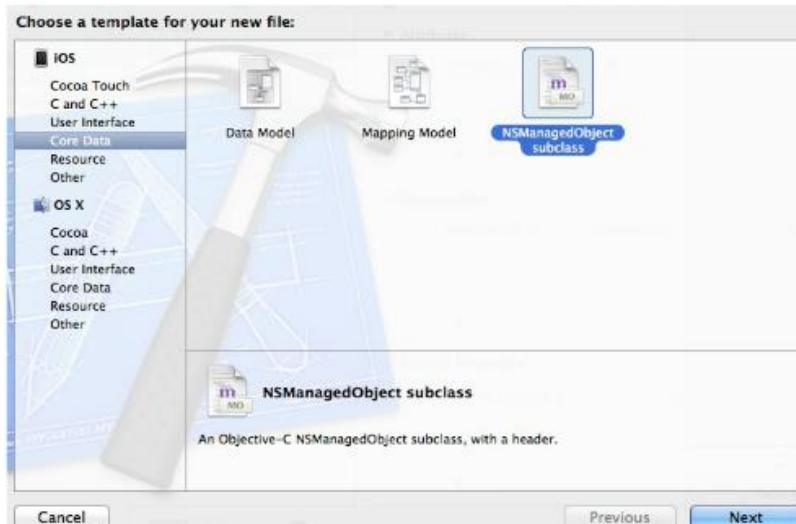


4 Tạo các lớp đối tượng

❑ Manage Object Class là lớp đối tượng được tạo ra cho các Entity.

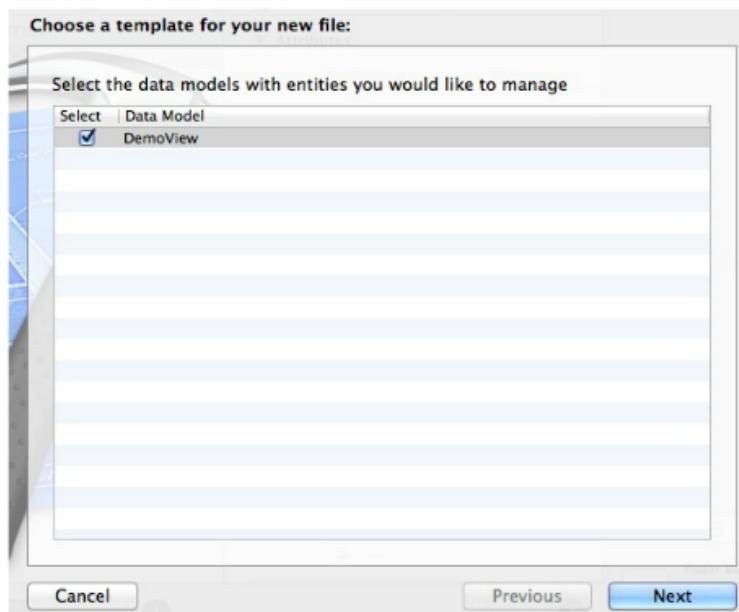
Việc phát sinh mã là hoàn toàn tự động. Ta thực hiện như sau.

- New -> New File... -> Core Data -> NSManagedObject subclass.



4 Tạo các lớp đối tượng

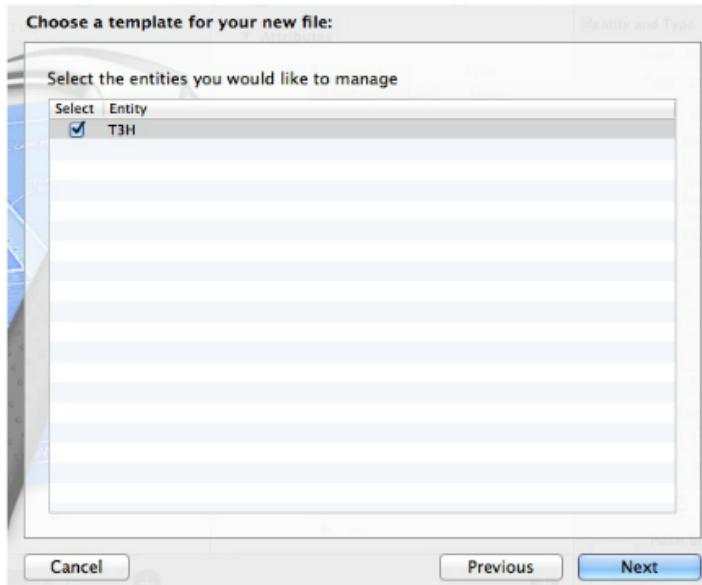
- Chọn Data Model có chứa đối tượng cần phát sinh tập tin quản lý.





4 Tạo các lớp đối tượng

- Chọn entity cần phát sinh tập tin quản lý. Xong ta chọn nơi lưu trữ tập tin.



4 Tạo các lớp đối tượng

- Sau khi hoàn thành ta sẽ có lớp với các trường dữ liệu của đối tượng entity.

File structure:

```

    DemoView
        2 targets, iOS SDK 7.1
        Entity
            Entity.h
            Entity.m
            NavigationViewController.h
            NavigationViewController.m
            NavigationViewController.xib
            WindowDemoViewController.h
            WindowDemoViewController.m
            WindowDemoViewController.xib
            SecondViewController.h
            SecondViewController.m
            SecondViewController.xib
            MainViewController.h
            MainViewController.m
            MainViewController.xib
        DemoView
            AppDelegate.h
            AppDelegate.m
            Images.xcassets
    
```

Content of Entity.h:

```

1 // 
2 // Entity.h
3 // DemoView
4 //
5 // Created by HocVien on 6/3/14.
6 // Copyright (c) 2014 HocVien. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10 #import <CoreData/CoreData.h>
11
12 @interface Entity : NSManagedObject
13
14 @property (nonatomic, retain) NSString * laptrinhiOS;
15 @property (nonatomic, retain) NSString * laptrinhAndoird;
16 @property (nonatomic, retain) NSString * laptrinhWindowPhone;
17
18 @end
19
20
    
```



Nội dung



1. Giới thiệu
2. Tích hợp Core Data Framework
3. Model entity
4. Tạo các lớp đối tượng
5. Tương tác với dữ liệu dùng Core Data Manage Context



5 Tương tác với dữ liệu dùng Core Data Manage Context



- ❑ Core Data Framework cung cấp cho ta đối tượng **Manage Object Context** tạo môi trường để thực hiện các hành động thêm, xoá, sửa cơ sở dữ liệu.
- ❑ Sử dụng lớp **NSEntityDescription** với phương thức **insertNewObjectForEntityForName:inManagedObjectContext:** để thêm dữ liệu:



5 Tương tác với dữ liệu dùng Core Data Manage Context



❑ Các bước cần thiết để thực hiện việc truy vấn với Manage Context:

- Tạo ra request
- Tạo Entity cần request
- Gán Entity cho request
- Gọi hàm đọc data với request
- Kiểm tra lỗi (optional)



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 4. Tiến trình và xử lý đa tiến trình

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



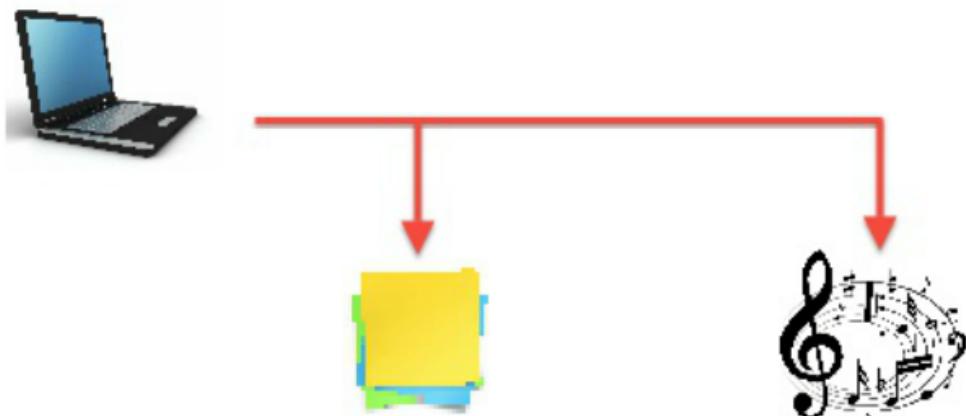
1. Định nghĩa tầm quan trọng của Multithreading

- Định nghĩa Multithreading
 - Multitasking (đa nhiệm)
 - Multithreading (đa luồng)
- Ý nghĩa và ví dụ

2. Multithreading trong Objective - C



1 Định nghĩa tầm quan trọng của Multithreading





1.1 Định nghĩa Multithreading

❑ Multitasking (đa nhiệm)

- Multitasking là thực thi hai hay nhiều tác nhiệm cùng một lúc. Gần như tất cả các hệ điều hành đều có khả năng multitasking và sử dụng một trong hai kỹ thuật là : multitasking dựa trên process(xử lý) hay multitasking dựa trên thread(phân tuyến).
- Multitasking dựa trên process là chạy hai chương trình cùng một lúc.
- Multitasking dựa trên thread là có một chương trình thực hiện hai hay nhiều tác nhiệm tại cùng một thời điểm.



1.1 Định nghĩa Multithreading

❑ Multitasking (đa nhiệm)

- Mục tiêu của multitasking là tận dụng thời gian nghỉ của CPU. Cũng giống như việc bạn vừa nghe nhạc và làm việc chẵng hạn. Trong quá trình làm việc có những lúc chúng ta không hoạt động, hay không suy nghĩ thì chúng ta sẽ nghe nhạc.
- Việc nghe nhạc xảy ra đồng thời song song với công việc trong tình huống ấy chúng ta chính là CPU.
- Việc sử dụng multitasking cũng giống như việc chúng ta muốn CPU của bạn xoay vòng để xử lý các lệnh và dữ liệu và đạt được hiệu quả một cách tốt nhất.



1.1 Định nghĩa Multithreading



Multithreading (đa luồng)

- Multithreading chính là multitasking thread như định nghĩa ở trên. Multithreading là hai hoặc nhiều phần của chương trình được chạy đồng thời cùng một lúc và mỗi phần được gọi là một Thread.



1.2 Ý nghĩa và ví dụ



- ❑ Vì tận dụng được CPU của máy. Nên về sử dụng multithreading sẽ giúp bạn điều khiển các xử lý đòi hỏi tính toán nhanh hơn.
- ❑ Đặc biệt trong trường hợp tốc độ truyền tải dữ liệu là chậm hoặc trong trường hợp xử lý các dữ liệu lớn, xử lý các hành động phức tạp cần thời gian.
- ❑ Xử lý các thao tác đồng thời nhằm giảm thiểu thời gian và đạt hiệu quả tốt đà.





1.2 Ý nghĩa và ví dụ

- ❑ **Ví dụ cụ thể :** Trong trường hợp ứng dụng cần load số lượng hình ảnh lớn từ server và hiển thị. Trong trường hợp này multithreading sẽ giúp hành động load dữ liệu hình ảnh và hiển thị diễn ra đồng thời. Giúp cho ứng dụng không bị block (khóa).



Nội dung

1. Định nghĩa tầm quan trọng của Multithreading
2. Multithreading trong Objective – C
 - Operation Queues
 - Dispatch Queues
 - Dispatch Sources





2.1 Operation Queues

❑ Operation Object

- Một Operation objects là một thể hiện của lớp NSOperation được sử dụng để gói gọn công việc mà bạn muốn ứng dụng thực hiện.
- Lớp NSOperation là một abstract class base(căn bản) bạn phải định nghĩa một subclass để có thể sử dụng. Mặc dù là abstract class nhưng class này không cung cấp các hàm mà bạn có thể sử dụng trong subclass.



2.1 Operation Queues

❑ Liệt kê các lớp và cách sử dụng Operation Object

Class	Mô tả
NSInvocationOperation	Là một subclass của NSOperation dùng để tạo một operation object dựa trên một object và selector từ ứng dụng của bạn. Được sử dụng trong trường hợp đã có một hàm tồn tại trong ứng dụng của bạn và bạn có nhu cầu sử dụng lại. Bởi vì nó không đòi hỏi phải cái một subclass. Sử dụng class này để tránh việc định nghĩa quá nhiều hàm void sử dụng trong app.
NSBlockOperation	Là một class sử dụng để thực hiện một hoặc nhiều block một cách đồng thời. Lớp này cung cấp một wrapper hướng đối tượng cho ứng dụng sử dụng operation queues và không muốn tạo ra một dispatch queues.
NSOperation	Là một class dùng để tùy chỉnh các operation objects. Bạn có thể thêm bất cứ hành vi nào bạn muốn, bạn có thể thay đổi phương pháp thực hiện và theo dõi tình trạng của operation





2.1 Operation Queues

❑ Thủ thuật khi sử dụng Operation Object

- Mặc dù Operation Object là rất dễ sử dụng. Nhưng trong quá trình sử dụng bạn cần lưu ý những phần sau:
 - Quản lý bộ nhớ trong Operation Objects
 - Kiểm tra kiểu mã lỗi
 - Kiểm tra lỗi phát sinh từ hàm nào
 - Bắt ngoại lệ trong code của bạn hoặc framework khác
 - Bắt ngoại lệ NSOperation

- Lưu ý : Không được sửa đổi tương operation khi đã thêm vào hàng đợi.



2.2 Dispatch Queues

❑ Giới thiệu về Dispatch Queues

- Grand Central Dispatch (GCD), dispatch queues là công cụ mạnh mẽ để thực hiện công việc hiệu quả.
- Với dispatch queues bạn có thể giải quyết tất cả các công việc thực hiện trên các thread khác nhau.
- Ưu điểm của dispatch queues là rất dễ sử dụng, hiệu quả và đơn giản.



2.2 Dispatch Queues



□ Giới thiệu về Dispatch Queues

- Dispatch queues là một cách dễ dàng để thực hiện công việc không đồng bộ (asynchronously) và đồng thời (concurrently) trong ứng dụng của bạn.
- Ví dụ bạn có thể tạo ra các công việc như: tính toán, đọc ghi dữ liệu tập tin, truyền tải dữ liệu,... Sau đó định nghĩa các công việc đó trong các hàm và thêm nó vào dispatch queues.



2.2 Dispatch Queues



□ Một số Dispatch Queues có sẵn và cách dùng

Loại	Mô tả
Serial	Hàng đợi nối tiếp (còn gọi là private dispatch queues) thực hiện công việc trong một thời gian và thứ tự mà chúng đợi thêm vào hàng đợi. Mỗi công việc được chạy trên một thread riêng biệt. Thường được sử dụng để đồng bộ hóa truy cập vào resource Bạn có thể tạo ra nhiều serial queue và chạy cùng lúc.
Concurrent	Concurrent queues (còn được gọi là global dispatch queue) thực hiện một hoặc nhiều công việc tại cùng một thời điểm và bắt đầu theo đúng thứ tự khi thêm vào hàng đợi. Công việc đang thực hiện chạy trên các thread khác nhau được quản lý bởi dispatch queue. Số công việc làm cùng lúc tại một thời điểm có thể thay đổi phụ thuộc vào điều kiện của hệ thống.
Main dispatch queue	Main dispatch queue là một globally available serial queue có sẵn thực hiện trên main thread của ứng dụng





2.2 Dispatch Queues

- ❑ Ngoài dispatch queues, Grand Central Dispatch cung cấp một số công nghệ để có thể quản lý code của bạn.

Công nghệ	Mô tả
Dispatch groups	Một dispatch group là cách để quản lý tập hợp một block (Bạn có thể theo dõi block synchronously hoặc asynchronously tùy thuộc vào nhu cầu của bạn).
Dispatch semaphores	Dispatch semaphores tương tự như semaphores nhưng hoạt động hiệu quả hơn
Dispatch sources	Dispatch sources tạo ra các thông báo để đáp ứng các loại sự kiện đặc biệt của hệ thống. Bạn có thể sử dụng Dispatch sources để theo dõi các sự kiện như: xử lý, thông báo, tín hiệu,... Khi có một sự kiện dispatch source sẽ gửi mã công việc vào hàng đợi không đồng bộ specified dispatch queue để xử lý.



2.2 Dispatch Queues

- ❑ Ví dụ:

```
dispatch_queue_t myQueue = dispatch_queue_create("My
Queue",NULL); // tạo một dispatch queue
dispatch_async(myQueue, ^{
    // Thực thi công việc
    dispatch_async(dispatch_get_main_queue(), ^{
        // cập nhật UI
    });
});
```





2.3 Dispatch Source

□ Giới thiệu về Dispatch Source

- Dispatch source là một loại dữ liệu cơ bản điều phối việc xử lý của các sự kiện của hệ thống. Grand Central Dispatch hỗ trợ các loại dispatch sources như sau :
- Timer dispatch sources : tạo ra các notification định kì.
- Signal dispatch sources : thông báo tín hiệu khi có UNIX đến.
- Descriptor sources: thông báo về các hoạt động của socket cơ bản của hệ thống.
- Process dispatch source : thông báo cho bạn các sự kiện liên quan đến quá trình xử lý.
- Mach port dispatch sources: thông báo cho bạn sự kiện liên quan tới mach.
- Custom dispatch sources: Cho bạn khả năng tùy chỉnh.



2.3 Dispatch Source

□ Bảng các hàm có thể gọi trong Event

Hàm	Mô tả
dispatch_source_get_handle	Hàm này trả về kiểu dữ liệu hệ thống cơ bản là dispatch source + Đối với descriptor dispatch source: trả về kiểu int chứa các mô tả kết hợp với dispatch source + Đối với signal dispatch source : trả về kiểu int quy định tín hiệu số cho sự kiện gần nhất + Đối với process dispatch source : trả về một cấu trúc dữ liệu cho quá trình theo dõi + Đối với Mach port dispatch source: trả về một cấu trúc dữ liệu mach_port_t + Đối với other dispatch sources: Không có giá trị trả về cho hàm này.
dispatch_source_get_data	Hàm này trả về dữ liệu đang chờ sự kết hợp với sự kiện.
dispatch_source_get_mask	Trả về flag cho sự kiện được dùng để khởi tạo dispatch source





2.3 Dispatch Queues

❑ Ví dụ:

```
dispatch_source_t source =  
    dispatch_source_create(DISPATCH_SOURCE_TYPE_READ,  
                           myDescriptor, 0, myQueue);  
  
dispatch_source_set_event_handler(source, ^{
    // Get some data from the source variable, which is captured
    // from the parent context.
    size_t estimated = dispatch_source_get_data(source);
    // Continue reading the descriptor...
});  
  
dispatch_resume(source);
```



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 5. Core Animation

Ngành Mạng & Thiết bị di động



2014

5014



Nội dung



1. Giới thiệu về Animation
2. UIView Animation
3. Core Animation



1 Giới thiệu về Animation



- ❑ Animation trong mobile là một tính năng rất quan trọng, tùy theo góc nhìn người ta sẽ có các ý nghĩa khác nhau về animation.
- ❑ Dưới đây là hình minh họa về animation đơn giản trong một ứng dụng chạy trên nền iOS:



Nội dung



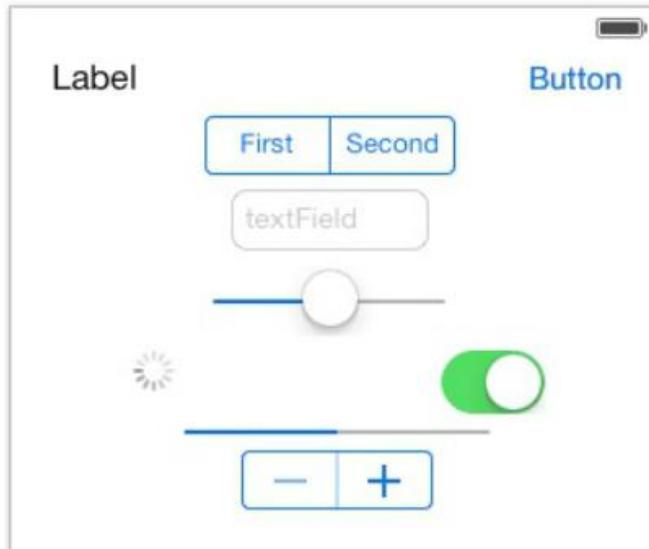
1. Giới thiệu về Animation
2. UIView Animation
 - UIView trong IOS
 - Các thuộc tính của UIView
 - Animation trong UIViews
 - Một số ví dụ về Animation trong UIView
3. Core Animation



2.1 UIView trong IOS



- ❑ UIView là một lớp cha cho các object được thể hiện trên màn hình trong iPhone (ở đây không bao gồm Core Animation và OpenGL). Button, text fields, images... đều là các đối tượng con của UIView.





2.1 UIImageView trong IOS

- ❑ Một UIImageView có thể chứa các view khác. Ta có thể coi UIImageView như một superview dùng để chứa các view khác và việc thêm hoặc xóa các view vào superview tương đối đơn giản như sau:

// Thêm một view

```
[viewObj addSubview:otherViewObj];
```

// Xóa một view

```
[otherViewObj removeFromSuperview];
```



2.2 Các thuộc tính của UIImageView

Properties	Tính năng
frame	Di chuyển, thay đổi kích thước của view
center	Di chuyển view
bounds	Thay đổi khung bao của view
transform	Rotate, scale, translate, skew
alpha	Thay đổi mức độ transparency trong đoạn [0,1]
backgroundColor	Thay đổi màu nền





2.3 Animation trong Views

- ❑ Thao tác hàm cơ bản để gọi thực hiện animation trong UIView:

```
[UIView animateWithDuration:[NSTimeInterval]]
```

```
animations:^{
    // Các hành động
}];
```



2.3 Animation trong Views

- ❑ Ngoài ra ta còn có thao tác hàm với trigger nhận sự kiện completion sau khi animation được thực hiện sau như sau:

```
[UIView animateWithDuration:[NSTimeInterval]]
```

```
animations:^{
    completion:^(BOOL finished)completion{
        }];
}
```





2.4 Một số ví dụ về Animation trong UIView

❑ Move:

```
[CGRect newFrame = theView.frame;  
  
newFrame.origin.x += 500;  
  
// dich chuyen sang phai 500pts  
  
[UIView animateWithDuration:1.0  
  
animations:^{
  
  
theView.frame = newFrame;  
  
}];
```



2.4 Một số ví dụ về Animation trong UIView

❑ Rotation:

```
float degrees = 45.0;  
  
float radians = (degrees/180.0) * M_PI;  
  
[UIView animateWithDuration:0.25  
  
animations:^{
  
  
theView.transform  
  
= CGAffineTransformMakeRotation(radians);  
  
}];
```



Nội dung



1. Giới thiệu về Animation
2. UIView Animation
3. Core Animation
 - Giới thiệu
 - Cơ bản về Core Animation trong IOS
 - Ví dụ



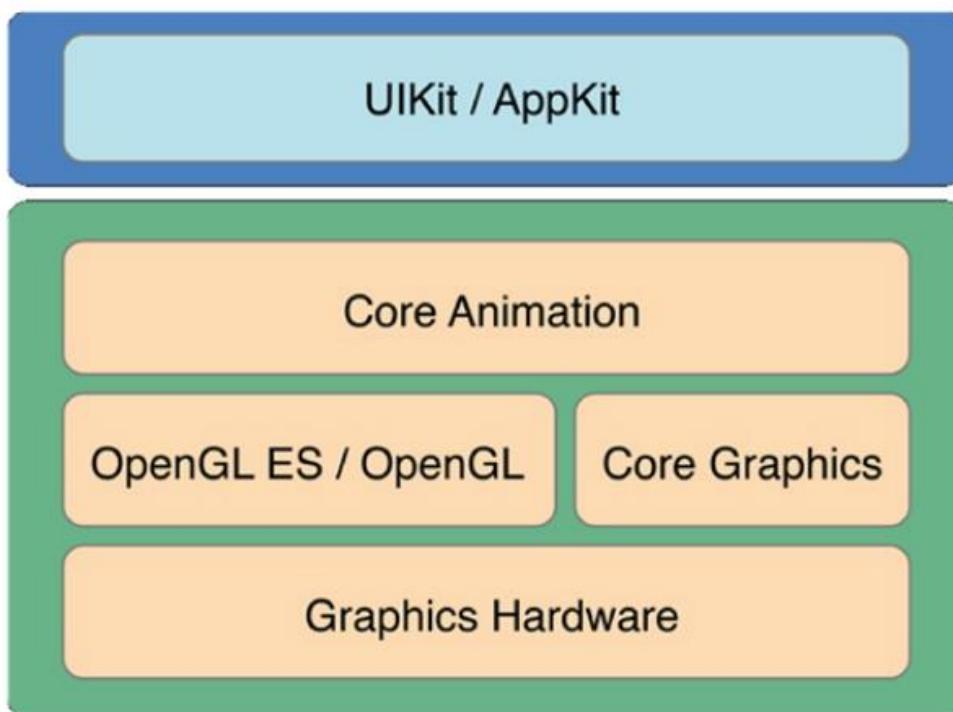
3.1 Giới thiệu



- ❑ Core Animation là một nền tảng trong việc thiết kế đồ họa và các chuyển động hình ảnh được tích hợp sẵn trong iOS và OS X.
- ❑ Với Core Animation có thể giúp ta tạo ra các hiệu ứng, chuyển hoạt hình ảnh trên ứng dụng.



3.1 Giới thiệu



3.2 Cơ bản về Core Animation trong iOS

- ❑ Layout Object
- ❑ Cách Core Animation thể hiện một đối tượng
- ❑ Các animation cơ bản trong iOS
- ❑ Các thuộc tính hình học của Layout Object



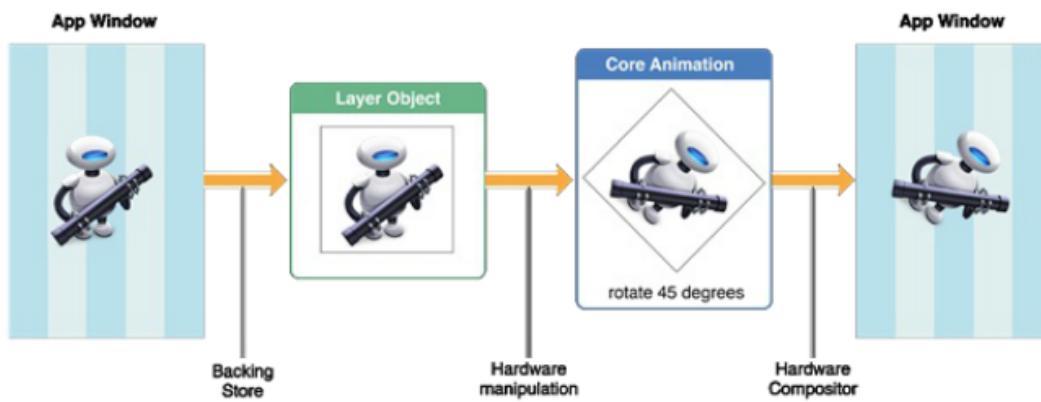


3.2.1 Layout Object

- Layout Object là một mặt phẳng 2 chiều được vẽ trong không gian (3D) và đây là một đối tượng cực kỳ quan trọng khi làm việc với Core Animation.



3.2.2 Cách Core Animation thể hiện một đối tượng



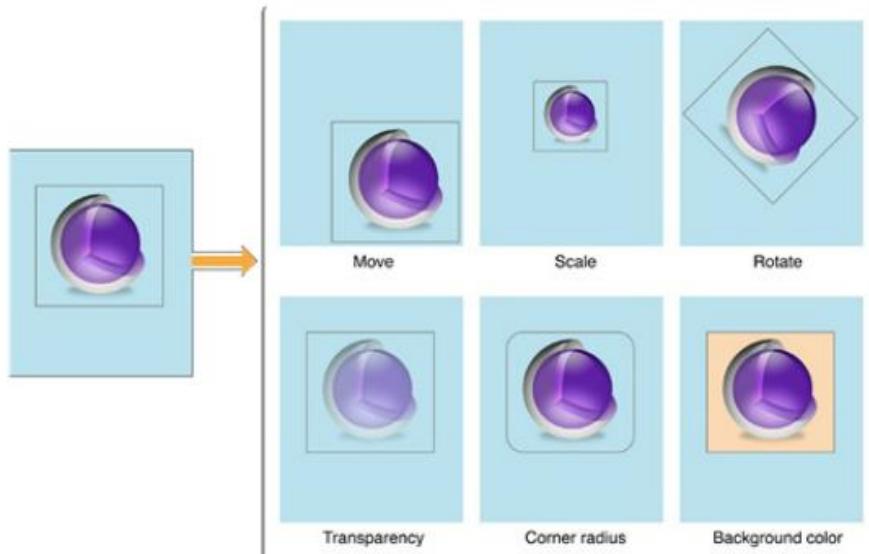
- Backing Store: hầu hết các layout không thực hiện bất cứ hành động vẽ nào lên màn hình, thay vào đó một layout muốn thể hiện một content lên màn hình thì nó sẽ thực hiện việc “capture” và “cache” đối tượng cần thể hiện vào bitmap.
- Hardware manipulation: quá trình thay đổi các thông tin, thuộc tính của layout tạo nên sự chuyển động của đối tượng.
- Hardware Compositor: tiếp nhận các thay đổi và chuyển các thông tin mới xuống phần cứng đồ họa.





3.2.3 Các animation cơ bản trong iOS

- Để thể hiện được các thao tác chuyển hoạt, bản thân Core Animation sẽ thực hiện vẽ từng khung ảnh (frame-by-frame) lên phần cứng đồ họa.

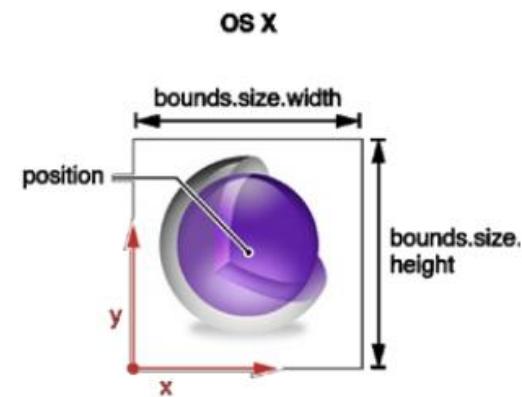
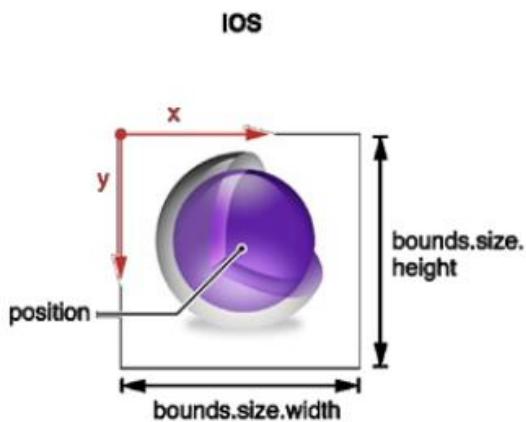


3.2.4 Các thuộc tính hình học của Layout Object

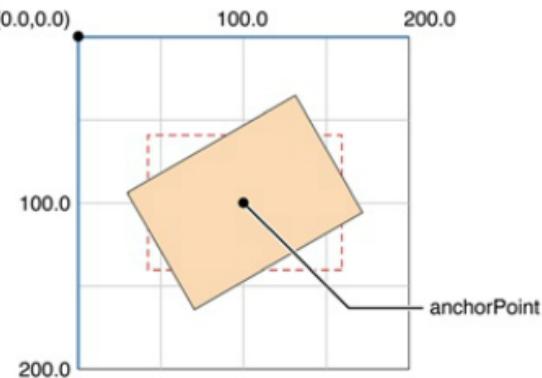
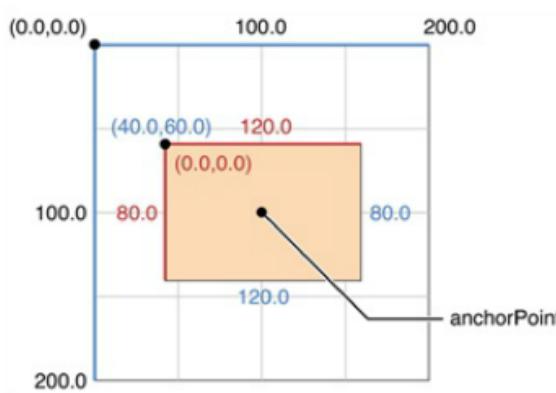
- Các hệ qui chiếu tọa độ:**
 - Point-based Coordinate systems: sử dụng cho các thuộc tính mà giá trị của nó được thể hiện trực tiếp thông qua giá trị của góc tọa độ: position, bound...
 - Unit Coordinate systems: sử dụng cho các thuộc tính mà giá trị của nó được thể hiện gián tiếp thông qua một giá trị khác (không nhất thiết là góc tọa độ): anchorPoint.



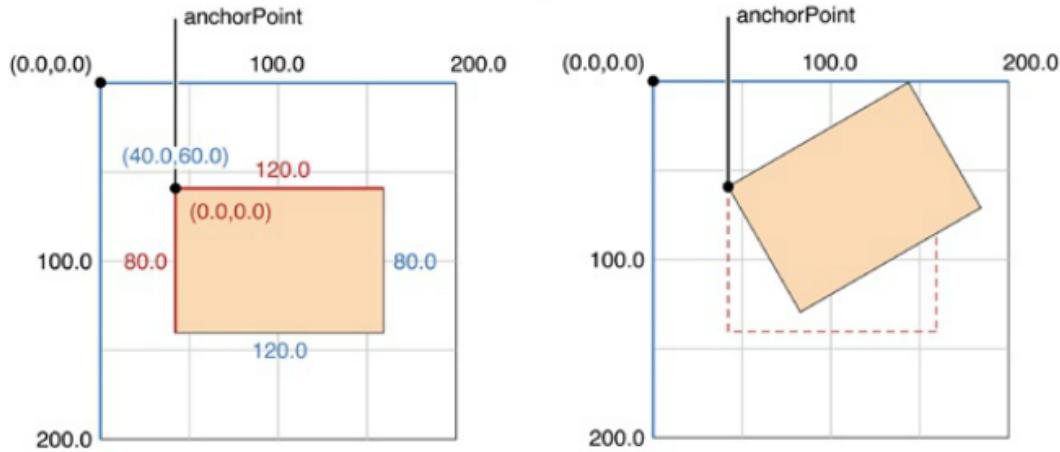
3.2.4 Các thuộc tính hình học của Layout Object



3.2.4 Các thuộc tính hình học của Layout Object



3.2.4 Các thuộc tính hình học của Layout Object



3.3 Ví dụ



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 6. *Touch & Gesture*

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



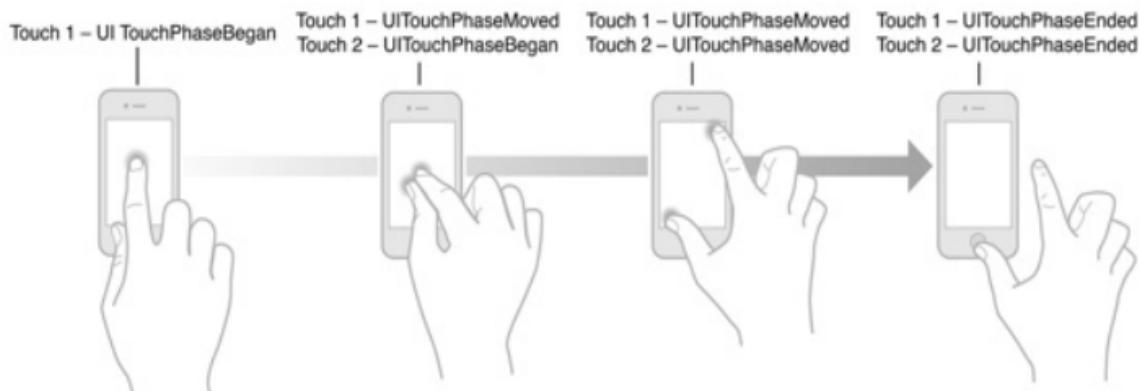
1. Touch

- Touch - Event
- Phase và Location của sự kiện Touch

2. Gesture



1 Touch





1.1 Touch - Event

- ❑ Touch trong iOS được ghi nhận như là một phần trong chuỗi sự kiện MultiTouch mà thông qua đó ứng dụng sẽ gửi chuỗi các event messages cho hệ thống nhận và xử lý tương ứng.
- ❑ Trong đó đối tượng UIResponder sẽ được kế thừa và xử lý các messages như sau:
 - (void)touchesBegan:(NSSet *) touches withEvent:(UIEvent *)event;
 - (void)touchesMoved:(NSSet *) touches withEvent:(UIEvent *)event;
 - (void)touchesEnded:(NSSet *) touches withEvent:(UIEvent *)event;
 - (void)touchesCancelled:(NSSet *) touches withEvent:(UIEvent *)event;



1.1 Touch - Event

- ❑ Như vậy ta thấy mỗi một chu trình Touch bao gồm các giai đoạn ta gọi là các phase: Began, Moved, Ended, Cancelled





1.2 Phase và Location của sự kiện Touch

- ❑ Đối tượng touch lưu trữ các thông tin của các giai đoạn (phase). Đồng thời nó cũng sẽ lưu trữ lại vị trí của sự kiện touch tương ứng theo một trong 3 cách sau:
 - The window: vị trí window nhận sự kiện touch
 - The view: vị trí view nhận sự kiện touch
 - Location: vị trí chính xác theo tọa độ trên view



Nội dung

1. Touch

2. Gesture

- Khái niệm và thuật ngữ
- UITapGestureRecognizer
- UIPanGestureRecognizer
- UIPinchGestureRecognizer và UIRotationGestureRecognizer





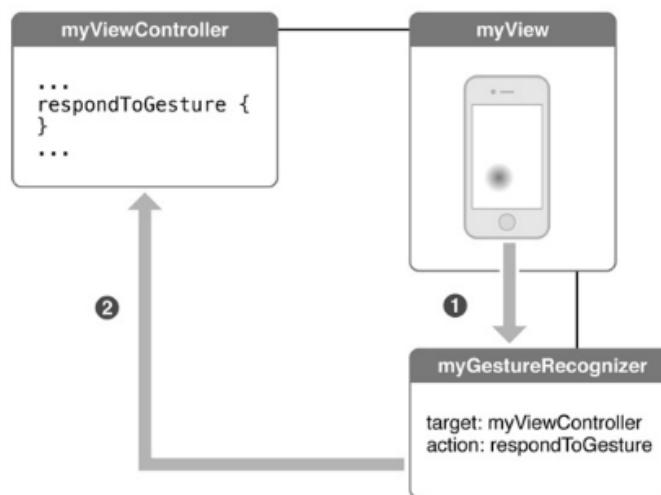
2.1 Khái niệm và thuật ngữ

- ❑ **Gesture:** là một chuỗi các sự kiện xảy ra được tính từ thời gian ta chạm vào màn hình với một hoặc nhiều ngón tay đến khi ta bỏ các ngón tay ra khỏi màn hình.
- ❑ **Tap:** là sự kiện ta chạm vào màn hình với một ngón tay và lập tức rút tay ra khỏi màn hình. Thiết bị iOS có thể xác định được a tapped, double-tapped, triple-tapped ... hoặc thậm chí 20-tapped.



2.1 Khái niệm và thuật ngữ

- ❑ **Gesture Recognizer:** dùng để quản lý, kiểm soát các dòng sự kiện gesture do người dùng tạo ra đồng thời ghi nhận lại các hành động touching và dragging.



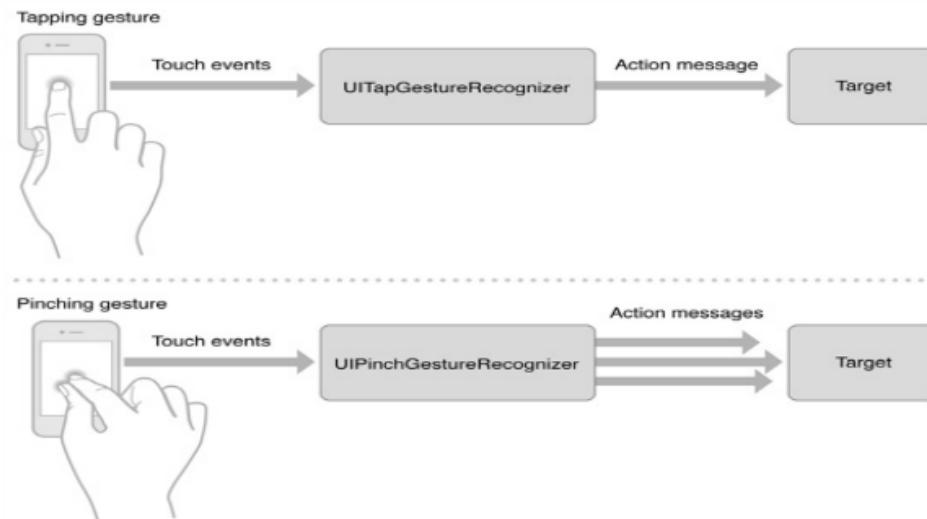
Hình: Luồng quản lý của Gesture Recognizer trên view





2.1 Khái niệm và thuật ngữ

- Ung với từng sự kiện touch gesture hay pinch gesture mà đối tượng Gesture Recognizer sẽ có các luồng sự kiện khác nhau để quản lý các event, action messages.



2.2 UITapGestureRecognizer

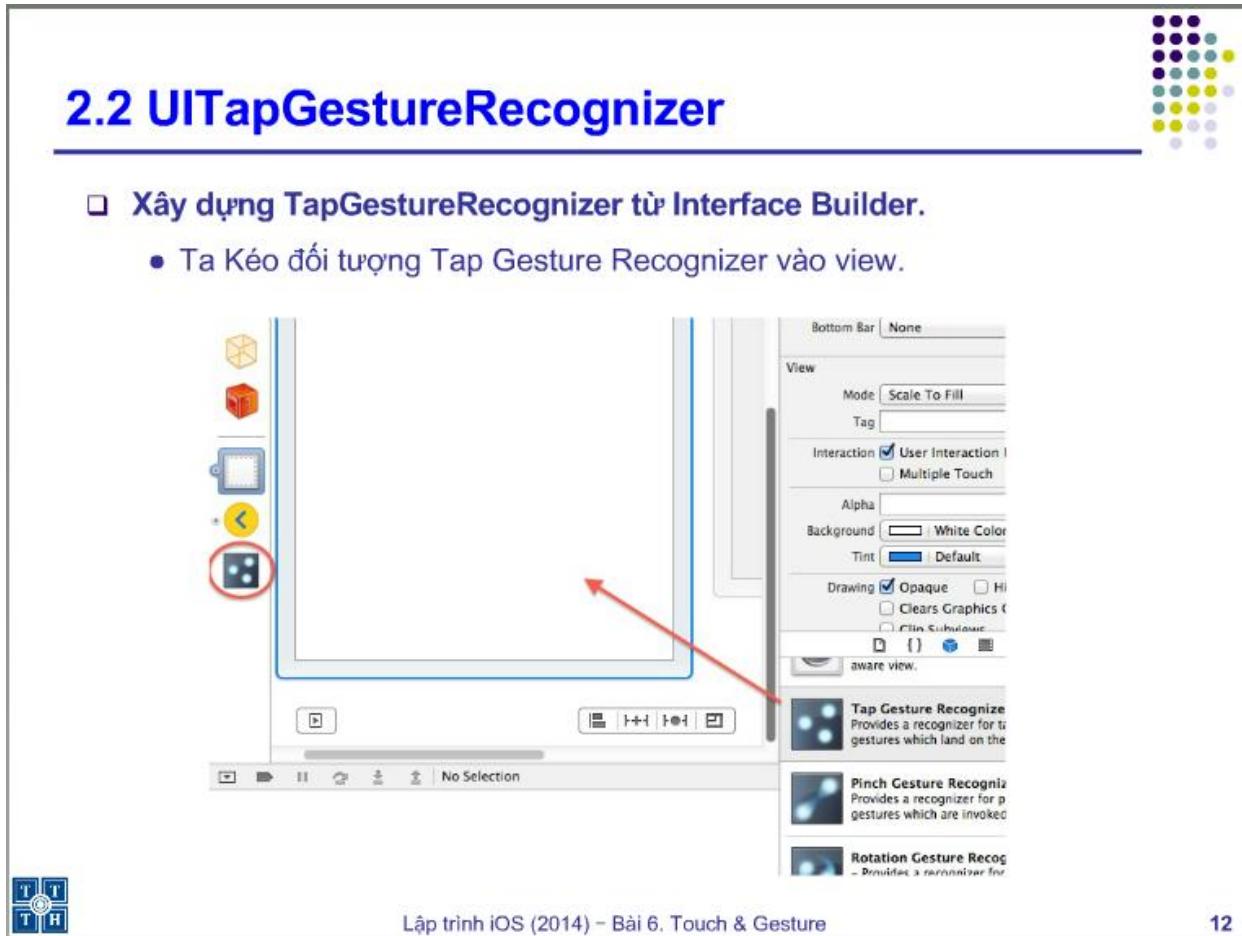
- UITapGestureRecognizer là một lớp con của lớp UIGestureRecognizer nó cho ta biết có một hay nhiều tap được thực hiện.
- Thuộc tính của UITapGestureRecognizer:
 - `numberOfTapsRequired` property
 - `numberOfTouchesRequired` property



2.2 UITapGestureRecognizer

- Xây dựng TapGestureRecognizer từ Interface Builder.

- Ta Kéo đổi tượng Tap Gesture Recognizer vào view.

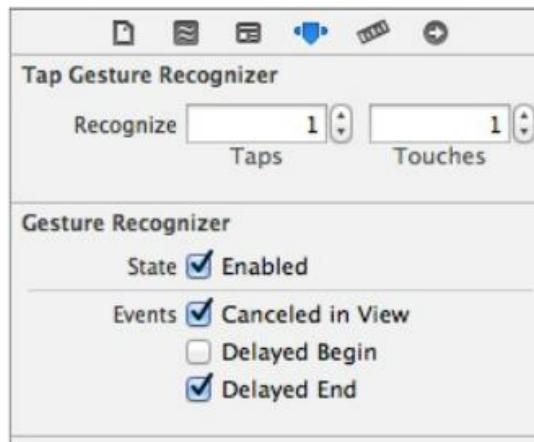


Lập trình iOS (2014) – Bài 6. Touch & Gesture

12

2.2 UITapGestureRecognizer

- Sau đó ta hiệu chỉnh thuộc tính của nó.



Lập trình iOS (2014) – Bài 6. Touch & Gesture

13

2.2 UITapGestureRecognizer



- Tạo kết nối outlet và xử lý sự kiện.

```

    - (void)viewDidLoad
    {
        [super viewDidLoad];
        // Do any additional setup after loading the view from its
        // nib.
    }

    - (void)
    didReceiveMemoryWarning
    {
        [super
            didReceiveMemoryWarning];
        // Dispose of any
        // resources that can
        // be recreated.
    }
    - (IBAction)singleTap:(id)
    sender {
}

```

delegate
 Sent Actions
 singleTap → File's Owner
 Referencing Outlets
 New Referencing Outlet
 Referencing Outlet Collections
 gestureRecognizers → View
 New Referencing Outlet Collection

Lập trình iOS (2014) – Bài 6. Touch & Gesture 14

2.2 UITapGestureRecognizer



- Xây dựng TapGestureRecognizer từ mã nguồn.

- Tạo đối tượng TapGestureRecognizer thiết lập hàm xử lý sự kiện sau đó add vào view.

```

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view from its
    // nib.

    UITapGestureRecognizer *tap = [[UITapGestureRecognizer
        alloc] init];
    [tap addTarget:self action:@selector(doTap)];
}

- (void)doTap{
    // Xử lý sự kiện
}

```



2.2 UITapGestureRecognizer



- Ta có thể thiết lập các thuộc tính cho tap.

```
[tap setNumberOfTapsRequired:2];
[tap setNumberOfTouchesRequired:1];
```



2.3 UIPanGestureRecognizer



- UITapGestureRecognizer là một lớp của lớp UIGestureRecognizer nó cho ta biết cử chỉ kéo được thực hiện.
- Các thuộc tính và phương thức của UIPanGestureRecognizer:
 - [maximumNumberOfTouches](#) *property*
 - [minimumNumberOfTouches](#) *propert*
 - [translationInView:](#)
 - [setTranslation:inView:](#)
 - [velocityInView:](#)



2.3 UIPanGestureRecognizer



❑ Sử dụng PanGestureRecognizer.

- Tao tạo đối tượng UIPanGestureRecognizer.

```
UIPanGestureRecognizer *pan = [[UIPanGestureRecognizer alloc] init];
```

- Thiết lập hành động cho Pan Gesture.

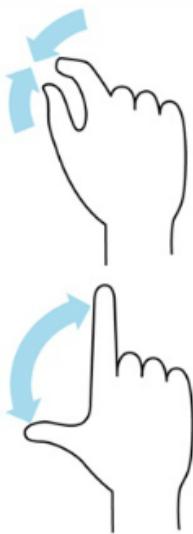
```
[pan addTarget:self action:@selector(doPan)];
```



2.4 UIPinchGestureRecognizer và UIRotationGestureRecognizer



❑ UIPinchGestureRecognizer và UIRotationGestureRecognizer là hai lớp con của GestureRecognizer dùng để nhận biết việc phóng to, thu nhỏ và xoay trên view.



2.4 UIPinchGestureRecognizer và UIRotationGestureRecognizer



- ❑ Để sử dụng hai đối tượng trên ta thực hiện tương tự như **UITapGestureRecognizer** và **UIPanGestureRecognizer**.
 - Tạo ra một Gesture Recognizer.
 - Tạo mối liên kết giữa Gesture Recognizer với View cần tương tác.
 - Xác định một phương thức (action) mà gesture đó sẽ gọi đến khi gesture đó bắt đầu, thay đổi hoặc kết thúc.



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 7. Search Field

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



1. Tại sao phải tìm kiếm trong ứng dụng
2. Giới thiệu về SearchBar
3. Giới thiệu về SearchDisplayController



1 Tại sao phải tìm kiếm trong ứng dụng



- ❑ Khi sử dụng các ứng dụng đôi khi người dùng có nhu cầu lọc dữ liệu theo các tiêu chí sao cho phù hợp với mục đích và những thông tin người dùng tìm kiếm.
- ❑ Trong trường hợp đó việc đưa ra chức năng tìm kiếm trong mỗi ứng dụng là rất cần thiết. Giúp cho người dùng cảm thấy tiện lợi và thoải mái khi sử dụng phần mềm.



Nội dung



1. Tại sao phải tìm kiếm trong ứng dụng

2. SearchBar

- Giới thiệu về SearchBar
- Khảo sát lớp UISearchBar
- Xây dựng SearchBar
- Các tác vụ trong SearchBar

3. SearchDisplayController



2.1 Giới thiệu về SearchBar



- Lớp UISearchBar triển khai một textfield cho những trường hợp tìm kiếm dựa trên văn bản. Quản lý textfield cho thao tác nhập văn bản, một nút tìm kiếm, một nút ghi nhớ các văn bản đã tìm kiếm, một nút hủy bỏ.
- Một đối tượng UISearchBar không thực hiện thao tác tìm kiếm mà các thao tác cần thiết chỉ thông qua các UISearchBarDelegate.





2.2 Khảo sát lớp UISearchBar

❑ Tuỳ chỉnh nội dung văn bản

- placeholder *property*
- prompt *property*
- text *property*

❑ Tuỳ chỉnh thuộc tính văn bản đầu vào

- autocapitalizationType *property*
- autocorrectionType *property*
- keyboardType *property*
- spellCheckingType *property*



2.2 Khảo sát lớp UISearchBar

❑ Tuỳ chỉnh thuộc tính hiển thị

- barStyle *property*
- barTintColor *property*
- searchBarStyle *property*
- tintColor *property*
- translucent *property*





2.2 Khảo sát lớp UISearchBar

❑ Tuỳ chỉnh cấu hình button

- [showsBookmarkButton *property*](#)
- [showsCancelButton *property*](#)
- [setShowsCancelButton:animated:](#)
- [showsSearchResultsButton *property*](#)
- [searchResultsButtonSelected *property*](#)
- [scopeButtonTitles *property*](#)
- [selectedScopeButtonIndex *property*](#)
- [showsScopeBar *property*](#)



2.2 Khảo sát lớp UISearchBar

❑ Tuỳ chỉnh hiển thị bên ngoài của SearchBar

- [backgroundImage *property*](#)
- [backgroundImageForBarPosition:barMetrics:](#)
- [setBackgroundImage:forBarPosition:barMetrics:](#)
- [imageForSearchBarIcon:state:](#)
- [setImage:forSearchBarIcon:state:](#)
- [positionAdjustmentForSearchBarIcon:](#)
- [setPositionAdjustment:forSearchBarIcon:](#)
- [inputAccessoryView *property*](#)





2.2 Khảo sát lớp UISearchBar

❑ Tuỳ chỉnh hiển thị bên ngoài của SearchBar

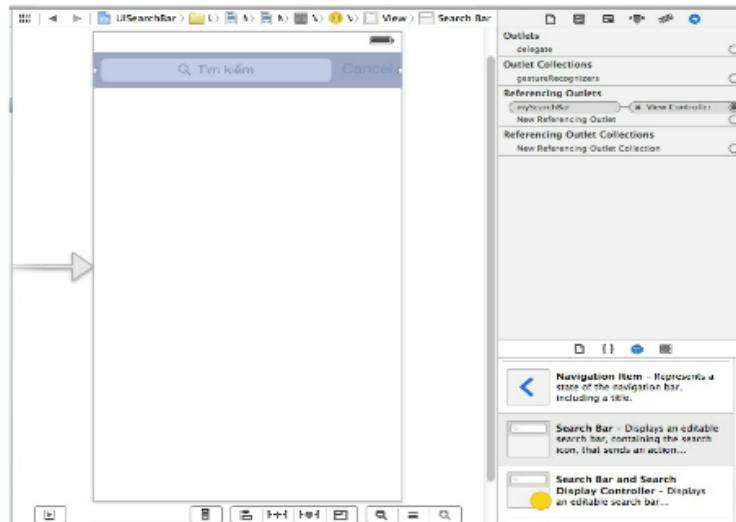
- [scopeBarBackgroundImage *property*](#)
- [scopeBarButtonBackgroundImageForState:](#)
- [setScopeBarButtonBackgroundImage:forState:](#)
- [scopeBarButtonDividerImageForLeftSegmentState:rightSegmentState:](#)
- [setScopeBarButtonDividerImage:forLeftSegmentState:rightSegmentState:](#)
- [scopeBarButtonTitleTextAttributesForState:](#)
- [setScopeBarButtonTitleTextAttributes:forState:](#)
- [searchFieldBackgroundImageForState:](#)
- [setSearchFieldBackgroundImage:forState:](#)
- [searchFieldBackgroundPositionAdjustment *property*](#)
- [searchTextPositionAdjustment *property*](#)



2.3 Xây dựng SearchBar

❑ Xây dựng SearchBar từ Interface Builder.

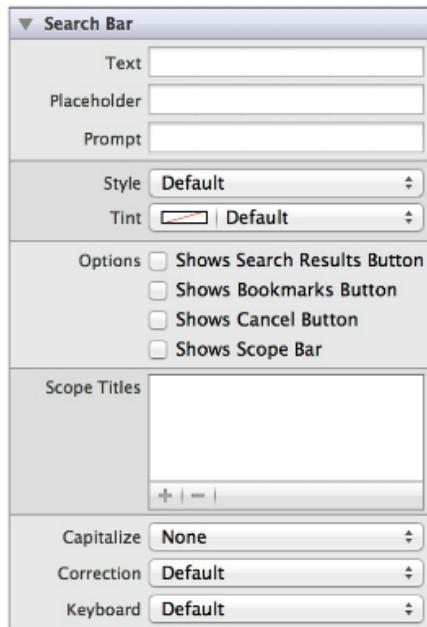
- Thực hiện kéo đối tượng **earchBar** từ Object Library vào xib, sau đó thực hiện tạo outlet cho **earchBar** đến File's Owner.





1.3 Xây dựng SearchBar

- **Tùy chỉnh một số thuộc tính của SearchBar trên Interface Builder.**



2.4 Các tác vụ trong SearchBar

- **Chỉnh sửa văn bản**

- [searchBar:textDidChange:](#)
- [searchBar:shouldChangeTextInRange:replacementText:](#)
- [searchBarShouldBeginEditing:](#)
- [searchBarTextDidBeginEditing:](#)
- [searchBarShouldEndEditing:](#)
- [searchBarTextDidEndEditing:](#)





2.4 Các tác vụ trong SearchBar

❑ Chọn button

- [searchBarBookmarkButtonClicked:](#)
- [searchBarCancelButtonClicked:](#)
- [searchBarSearchButtonClicked:](#)
- [searchBarResultsListButtonClicked:](#)

❑ Scope Button

- [searchBar:selectedScopeButtonIndexDidChange:](#)



Nội dung

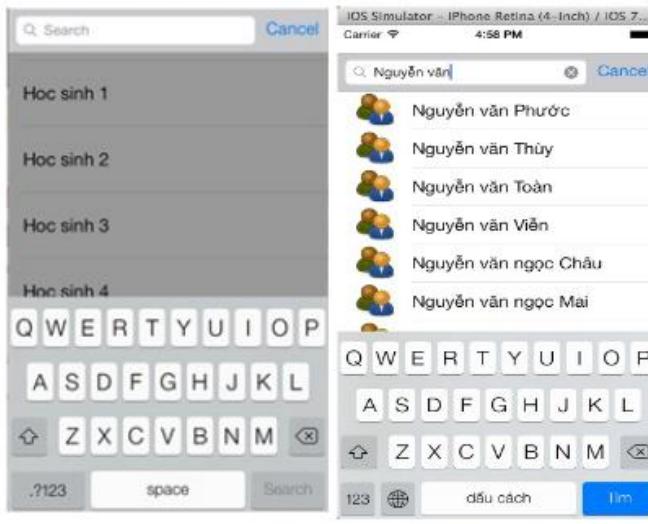
1. Tại sao phải tìm kiếm trong ứng dụng
2. SearchBar
3. **SearchDisplayController**
 - Giới thiệu về SearchDisplayController
 - Khảo sát lớp UISearchDisplayController
 - Xây dựng SearchDisplayController
 - Các tác vụ trong SearchDisplayController





3.1 Giới thiệu về SearchDisplayController

- **SearchDisplayController** bao gồm một **UISearchBar** và một **table** dùng để hiển thị kết quả trả về (bắt buộc phải có).
- **Table** chứa kết quả trả về có cấu trúc tương tự **UITableViewCell**.



Lập trình iOS (2014) – Bài 7. Search Field

16



3.2 Khảo sát lớp UISearchDisplayController

- **Tạo một SearchDisplayController với SearchBar**
 - [initWithSearchBar:contentsController:](#)
- **Hiển thị Search trên Interface**
 - [active property](#)
 - [setActive:animated:](#)



Lập trình iOS (2014) – Bài 7. Search Field

17

3.2 Khảo sát lớp UISearchDisplayController



❑ Tuỳ chỉnh cấu hình SearchBar

- [delegate property](#)
- [searchBar property](#)
- [searchContentsController property](#)
- [searchResultsTableView property](#)
- [searchResultsDataSource property](#)
- [searchResultsDelegate property](#)
- [searchResultsTitle property](#)
- [displaysSearchBarInNavigationBar property](#)
- [navigationItem property](#)

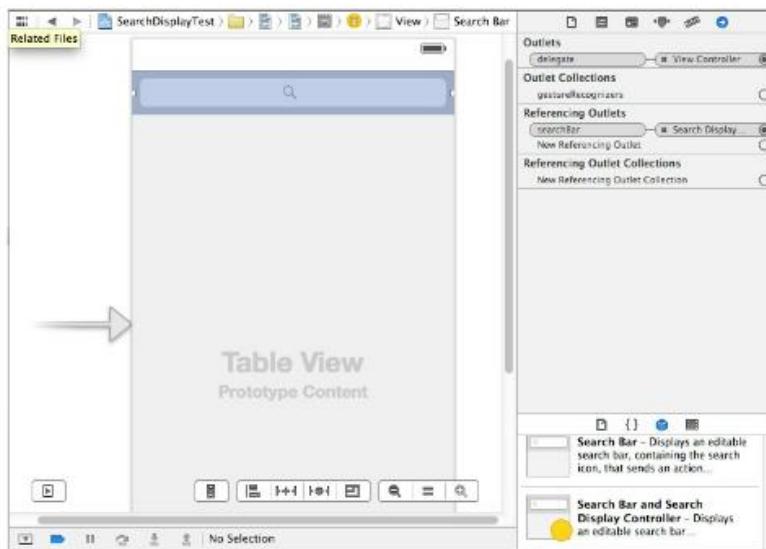


3.3 Xây dựng SearchDisplayController



❑ Xây dựng SearchDisplayController từ Interface Builder.

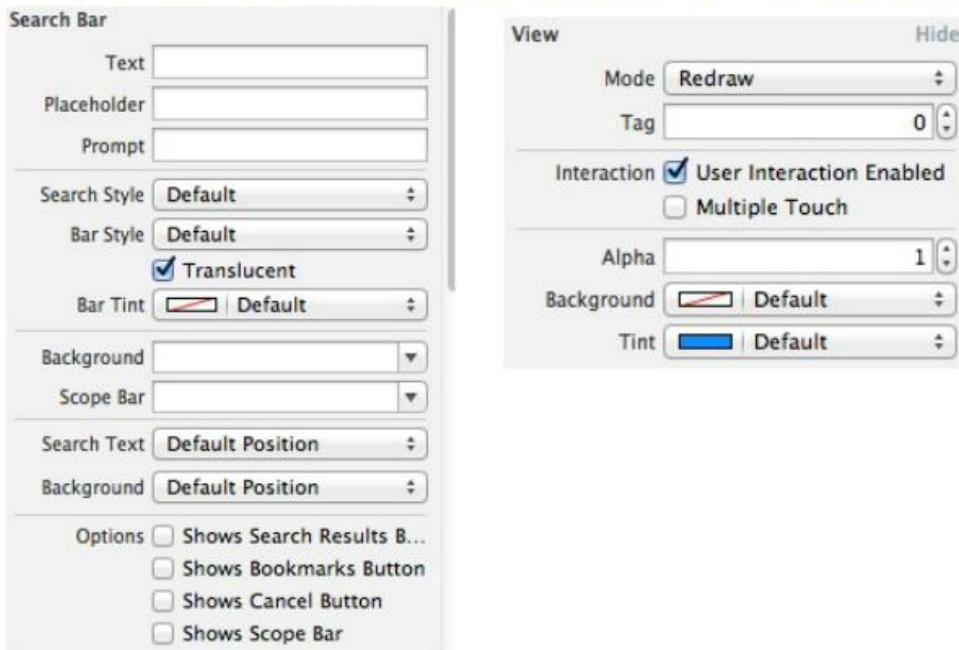
- Thực hiện kéo đối tượng **Search bar and Search Display Controller** từ Object Library vào xib.





3.3 Xây dựng SearchBar

- Tuỳ chỉnh một số thuộc tính của SearchBar trên Interface Builder.



3.4 Các tác vụ trong SearchDisplayController

- Thay đổi trạng thái Search

- [searchDisplayControllerWillBeginSearch:](#)
- [searchDisplayControllerDidBeginSearch:](#)
- [searchDisplayControllerWillEndSearch:](#)
- [searchDisplayControllerDidEndSearch:](#)





3.4 Các tác vụ trong SearchDisplayController

❑ Tải và ngừng tải dữ liệu TableView

- [searchDisplayController:didLoadSearchResultsTableView:](#)
- [searchDisplayController:willUnloadSearchResultsTableView:](#)

❑ Hiển thị và ẩn TableView

- [searchDisplayController:willShowSearchResultsTableView:](#)
- [searchDisplayController:didShowSearchResultsTableView:](#)
- [searchDisplayController:willHideSearchResultsTableView:](#)
- [searchDisplayController:didHideSearchResultsTableView:](#)



3.4 Các tác vụ trong SearchDisplayController

❑ Xử lý khi có sự thay đổi trong tiêu chí Search

- [searchDisplayController:shouldReloadTableForSearchString:](#)
- [searchDisplayController:shouldReloadTableForSearchScope:](#)



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 8. *Split View Controller*

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



1. Split View Controller

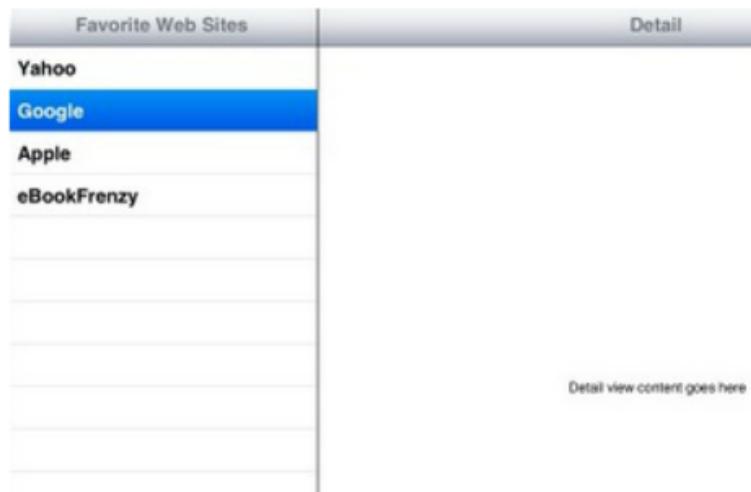
- Mục đích sử dụng trong ứng dụng
- Khảo sát lớp UISplitViewController
- Xây dựng Split View Controller
- Các tác vụ trong Split View Controller
- Demo



1.1 Mục đích sử dụng trong ứng dụng



- Split View được sử dụng nhằm hiển thị nội dung màn hình với một menu bên trái và nội dung bên phải.**



1.1 Mục đích sử dụng trong ứng dụng



❑ Một số hình ảnh về Split View Controller.

Lists for Writers	Occupations
Character	longshoreman
First Names (Male)	mail carrier
First Names (Female)	maître d'
Last Names	medical technician
Clothing	meteorologist
Occupations	midwife
Personality	miner
Phobias	minister
Philias	model
Obsessions	monk
Character Traits	mortician
Hobbies	

More	Subscriptions
My Account	Your subscription
Subscription	You don't have a valid subscription on this device. To make sure you don't already have a subscription or to sign devices please Login or Register.
Download Manager	
Terms of use	
Feedback	
About	Login or Register

Subscriptions
first 2 month access €9.99
WELCOME Lützen's ARCHIVE Digital Access
2 month access €8.49
TWO MONTH Lützen's ARCHIVE Digital Access



1.1 Mục đích sử dụng trong ứng dụng



❑ Tổng quan.

- Mỗi panel được quản lý bởi một controller điều khiển riêng cho mình tùy thuộc vào mục đích sử dụng mà thiết kế cho phù hợp.
- Bên cạnh đó một số giao tác như quản lý quay màn hình được split view điều khiển thông qua delegate của nó.
- Một splitview controller phải là control gốc của các control còn lại. Nói cách khác splitview phải là root view của application's window trong ứng dụng của bạn.
- Mỗi panel có thể là navigation controller, tab bar controllers hoặc bất cứ loại view controller nào bạn cần.





1.1 Mục đích sử dụng trong ứng dụng

- Split view controllers không thể được sử dụng bằng cách presented.
- Cách đơn giản nhất để tạo một split view controller là khi tạo project mới. Chọn Split View-based Application template trong xcode project mặc định sẽ bắt đầu bằng một split view controller. Việc còn lại là chỉ việc sử dụng các view controller khác cho panel left và right.



1.2 Khảo sát lớp UISplitViewController

Thuộc tính quản lý View Controller con

- viewControllers *property*
- presentsWithGesture *property*

Thuộc tính delegate

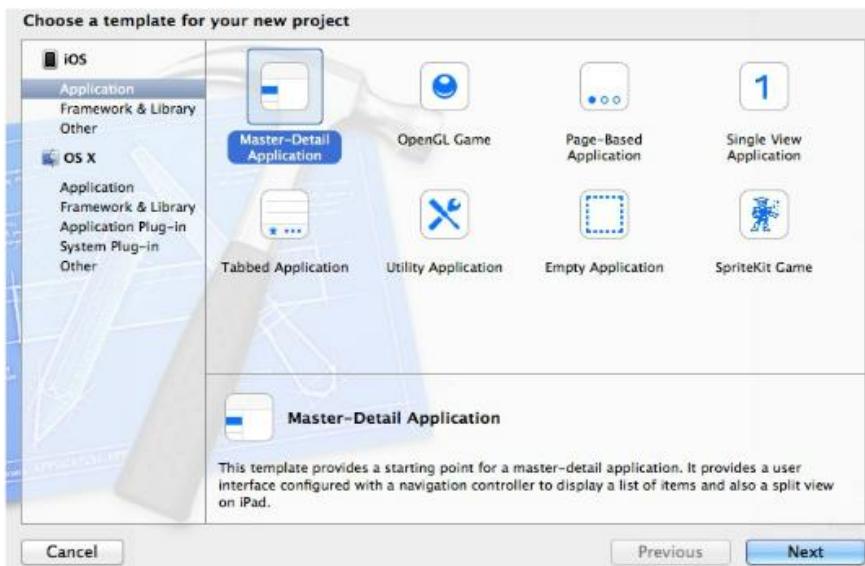
- delegate *property*





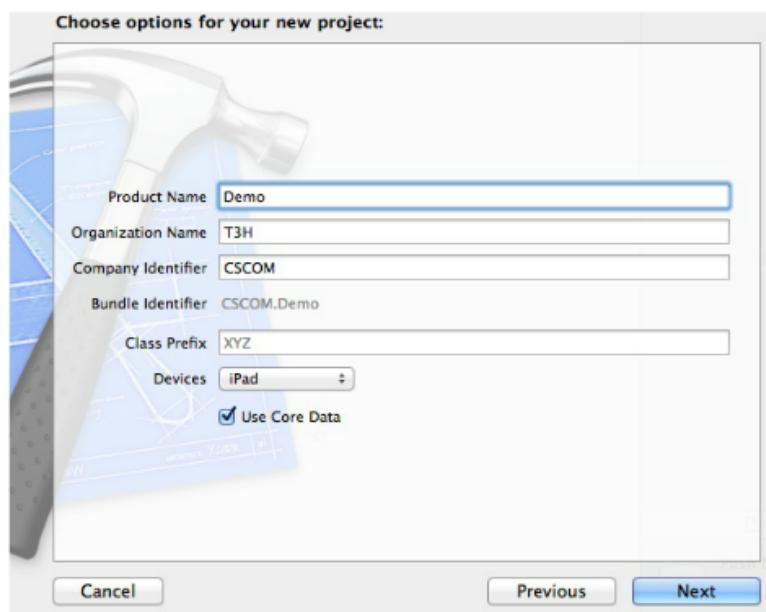
1.3 Xây dựng Split View Controller

- Ta có thể dùng Split View Controller bằng cách chọn Master-Detail Application khi tạo project:



1.3 Xây dựng Split View Controller

- Tiếp theo chọn devive cho ipad :



1.3 Xây dựng Split View Controller



- Cuối cùng project được tạo ra có giao diện gốc là một splitview :

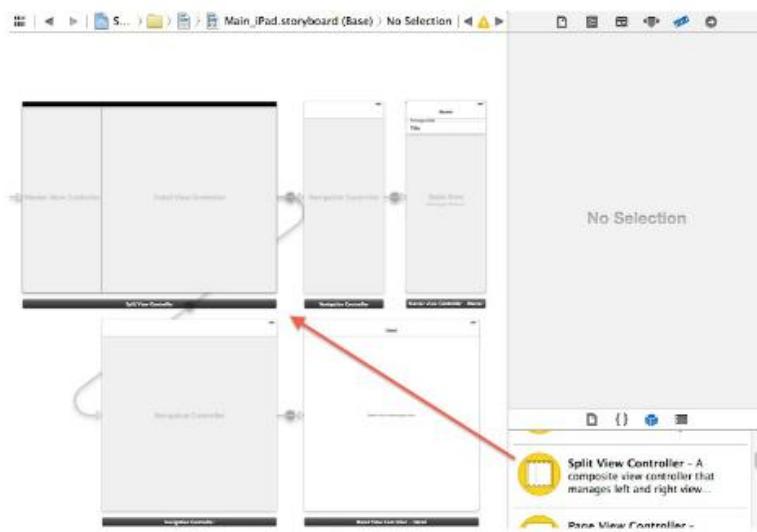


1.3 Xây dựng Split View Controller



- Xây dựng Split View Controller từ Interface Builder:

- Thực hiện kéo đối tượng UISplitViewController từ Object Library vào xib.





1.3 Xây dựng Split View Controller

❑ Xây dựng Split View Controller từ mã nguồn:

- Tạo ra đối tượng View Controller.

```
// Khởi tạo panel left và paner right
```

```
MyFirstViewController* firstVC = [[MyFirstViewController alloc] init];
```

```
MySecondViewController* secondVC = [[MySecondViewController alloc] init];
```



1.3 Xây dựng Split View Controller

- Tạo ra Split View Controller với View Controller vừa tạo.

```
// Khởi tạo splitview controller với 2 panel left và right
```

```
UISplitViewController* splitVC = [[UISplitViewController alloc] init];
```

```
splitVC.viewControllers = [NSArray arrayWithObjects:firstVC, secondVC,  
nil];
```

- Thiết lập Split View Controller làm root view.

```
window.rootViewController = splitVC;
```





1.4 Các tác vụ trong Split View Controller

❑ Hiển thị và ẩn View Controller

- [tableView:heightForRowAtIndexPath:](#)
- [splitViewController:shouldHideViewController:inOrientation:](#)
- [splitViewController:willHideViewController:withBarButtonItem:forPopoverController:](#)
- [splitViewController:willShowViewController:invalidatingBarButtonItem:](#)
- [splitViewController:popoverController:willPresentViewController:](#)



1.4 Các tác vụ trong Split View Controller

❑ Cấu hình các phương thức chuyển View

- [splitViewControllerSupportedInterfaceOrientations:](#)
- [splitViewControllerPreferredInterfaceOrientationForPresentation:](#)



1.5 Demo



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 9. Storyboard

Ngành Mạng & Thiết bị di động



2014

2014



Nội dung



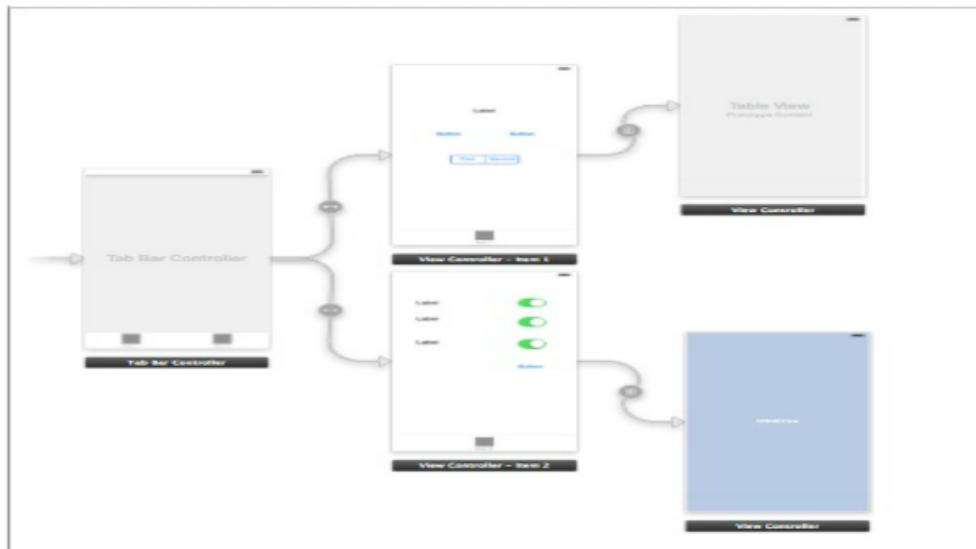
1. Giới thiệu
 - Đặt vấn đề
 - Tổng quan
2. Storyboard
3. Ưu điểm và khuyết điểm
4. Demo
5. Q&A



1.2 Tổng quan



- ❑ Storyboard được apple giới thiệu ra mắt khi IOS 5 ra đời . Và nó ra đời để giải quyết vấn đề trên.





1.2 Tổng quan

- **Storyboard** là một cách thức tổ chức, quản lý các file giao diện của ứng dụng.
- **Storyboard** bao gồm một chuỗi trình tự các view. Mỗi view có một controller điều khiển hoặc chung đại diện cho mình được kết nối với nhau thông qua đối tượng segue là đối tượng dùng để chuyển tiếp giữa hai view trên một storyboard.
- Xcode cung cấp một trình giao diện dùng để sử dụng storyboard một cách trực quan nhất. Bạn có thể thiết kế giao diện người dùng thêm những view tùy ý. Storyboard giúp bạn hình dung được luồng sơ đồ các màn hình một cách trực quan và cụ thể nhất.



Nội dung

1. Giới thiệu
2. **Storyboard**
 - Hàm cơ sở của Storyboard
 - Giới thiệu về Segues trong storyboard
 - Cách tạo một segue đơn giản
 - Giới thiệu về UIStoryboardSegue
 - Storyboard và UIViewController
3. Ưu điểm và khuyết điểm
4. Demo
5. Q&A





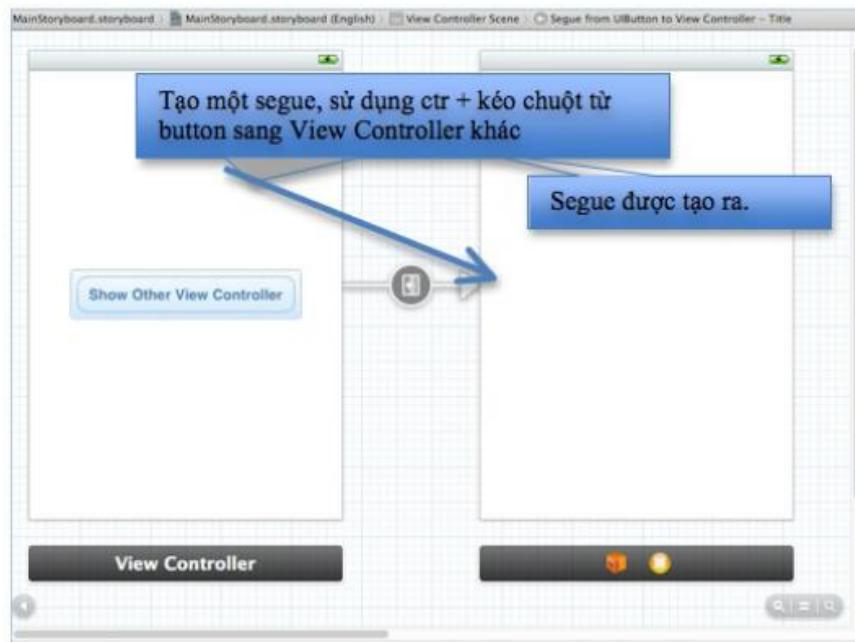
2.1 Hàm cơ sở của Storyboard

- Nhận một đối tượng Storyboard.
 - [storyboardWithName:bundle:](#)
- Tạo Storyboard View Controller.
 - [instantiateViewControllerInitialViewController](#)
 - [instantiateViewControllerWithIdentifier:](#)



2.2 Giới thiệu về Segues trong storyboard

- Tạo một segue đơn giản .



2.2 Giới thiệu về Segues trong storyboard



❑ Tạo một segue đơn giản .

The screenshot shows the Xcode Storyboard Editor. A blue callout box points to a segue arrow originating from a button labeled "Show Other View Controller". Another blue callout box points to the same segue arrow. Both callout boxes contain the text: "Có thể thay đổi loại segue bằng cách thay đổi trong property. Và định nghĩa segue bằng identifier". The storyboard scene contains a "View Controller" and a "Segue from UIButton to View Controller - Title". The storyboard sidebar shows objects like "Table View Controller" and "Navigation Controller". The bottom status bar indicates "Lập trình iOS (2014) – Bài 9, Storyboard".

8

2.2 Giới thiệu về Segues trong storyboard



❑ Loại segue được tạo ra .

- Push : Được sử dụng trong trường hợp có NavigationController
- Modal : Gọi giữa các ViewController với nhau.
- Custom : Các loại segue tùy chỉnh .

❑ Cách tạo một segue bằng code : Sử dụng hàm của ViewController là.

- - `(void)performSegueWithIdentifier:(NSString *)identifier sender:(id)sender`





2.2 Giới thiệu về Segues trong storyboard

- **Giới thiệu về UIStoryboardSegue :** Là đối tượng chịu trách nhiệm cho quá trình thay đổi View mới trong storyboard. Đồng thời tạo ra các luồng dữ liệu giữa các View.
- **Các phương thức và thuộc tính của UIStoryboardSegue.**
 - [initWithIdentifier:source:destination:](#)
 - [sourceViewController property](#)
 - [destinationViewController property](#)
 - [identifier property](#)
 - [Perform](#)
 - [segueWithIdentifier:source:destination:performHandler:](#)



2.3 Storyboard và UIViewController

- **Một UIViewController có các phương thức đặc biệt để sử dụng hiệu quả trong storyboard như :**
 - [shouldPerformSegueWithIdentifier:sender:](#)
 - [performSegueWithIdentifier:sender:](#)
 - [prepareForSegue:sender:](#)
 - [canPerformUnwindSegueAction:fromViewController:withSender:](#)



Nội dung



1. Giới thiệu
2. Storyboard
3. **Ưu điểm và khuyết điểm**
 - Ưu điểm
 - Khuyết điểm
4. Demo
5. Q&A



3.1 Storyboard và UIViewController



- Ưu điểm:**
- Storyboard giúp cho việc hình dung, quan sát luồng UI một cách dễ dàng..
 - Storyboard giúp bạn giảm số lượng code mà bạn sinh ra như trong trường hợp gọi một View khác bạn phải khởi tạo và code chuyển màn hình.



3.2 Storyboard và UIViewController



❑ Khuyết điểm:

- Tuy nhiên ưu điểm bên trên lại đem lại bất lợi khi project của bạn phức tạp hơn về số lượng màn hình. Khi đó nhìn vào sơ đồ Storyboard bạn sẽ cảm thấy nó rối hơn.
- Xcode hỗ trợ bạn xây dựng ứng dụng với nhiều Storyboard khác nhau. Nhưng dù sao nó cũng sẽ rất rối mắt khi flow của bạn có nhiều màn hình.
- Storyboard mang lại sự bất tiện khi làm việc với nhóm.
- Với những projetc nhiều màn hình, storyboard sẽ gây khó khăn cho bạn trong việc tìm kiếm chủ đề nào đó trong storyboard.
- Không tương thích với iOS 4 trở xuống.



Nội dung



1. Giới thiệu
2. Storyboard
3. Ưu điểm và khuyết điểm
4. Demo
5. Q&A



4 Demo



Thảo luận

