



LẬP TRÌNH iOS

Module 1

☞ Click vào phụ lục để chuyển tới bài cần đọc

Phụ lục

Bài 1 Giới thiệu Objective-C và lập trình ứng dụng iOS	2
Bài 2 Kiểu dữ liệu.....	10
Bài 3 Xây dựng ứng dụng iOS	30
Bài 4 Cấu trúc điều khiển.....	42
Bài 5 Hướng đối tượng trong Objective-C.....	53
Bài 6 Các điều khiển cơ bản	61
Bài 7 Lưu trữ dữ liệu dạng tập hợp	76
Bài 8 Đọc ghi tập tin	84
Bài 9 ScrollView & Page Control.....	96



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

[Go Screen Capture](#)

Lập trình iOS

Bài 1. *Giới thiệu ngôn ngữ Objective-C & lập trình ứng dụng iOS*

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Sơ lược về ngôn ngữ Objective-C
2. Objective-C và các ngôn ngữ khác
3. Môi trường thực thi Objective-C
4. Môi trường phát triển



1. Sơ lược về ngôn ngữ Objective-C



- ❑ Ngôn ngữ Objective C là sự kết hợp giữa SmallTalk và C.
- ❑ Được tạo ra bởi Brad Cox và Tom Love.
- ❑ Năm 1996, Apple ra mắt hệ điều hành MAC OS dựa trên nền tảng Objective-C.
- ❑ Năm 2007, phiên bản Objective-C 2.0 được Apple ra mắt bổ sung nhiều tính năng vượt trội.
- ❑ Hiện tại Objective C là ngôn ngữ chính trong lập trình ứng dụng trên hệ điều hành MAC và hệ điều hành iOS (iPhone, iPad, iPod) của Apple.





1. Sơ lược về ngôn ngữ Objective-C

- **Nền tảng phát triển Objective-C của Apple có thể gói gọn bao gồm các thành phần sau:**
 - Ngôn ngữ lập trình Objective-C
 - Môi trường thực thi
 - Bộ thư viện phát triển phần mềm
 - Bộ công cụ phát triển phần mềm



Nội dung

1. Sơ lược về ngôn ngữ Objective-C
2. Objective-C và các ngôn ngữ khác
3. Môi trường thực thi Objective-C
4. Môi trường phát triển





2. Objective-C và các ngôn ngữ khác

Một số điểm nổi bật của Objective-C so với các ngôn ngữ khác:

- Lập trình hướng đối tượng: mang đầy đủ các tính năng như tính kế thừa, tính đóng gói, tính đa hình, đệ quy mở...
- Đối tượng thông tin (Objective messaging): dạng gói tin được các đối tượng truyền thông tin với nhau.
- Bộ thực thi động: cho phép chuyển hướng tham chiếu đến các kiểu dữ liệu khác nhau thay cho việc tạo liên kết.
- Bộ quản lý bộ nhớ: cung cấp lựa chọn hoặc quản lý bộ nhớ thủ công hoặc tự động quản lý theo ARC.



2. Objective-C và các ngôn ngữ khác

Một số điểm nổi bật của Objective-C so với các ngôn ngữ khác:

- Giám sát đối tượng và phản xạ: cho phép truy vấn đối tượng và rút trích thông tin trong suốt quá trình thực thi của chương trình.
- Hỗ trợ ngôn ngữ C: cho phép truy xuất và sử dụng trực tiếp các thư viện C chuẩn.
- Công nghệ Apple: tận dụng tối đa các hạ tầng kết nối và tư duy dựa trên nền tảng phát triển ứng dụng chuẩn của Apple.



Nội dung



1. Sơ lược về ngôn ngữ Objective-C
2. Objective-C và các ngôn ngữ khác
3. Môi trường thực thi Objective-C
4. Môi trường phát triển



3. Môi trường thực thi Objective-C



- ❑ **Môi trường thực thi là nơi phân tích và thực thi các dòng mã lệnh cũng như kiểm soát ngoại lệ và lỗi chương trình.**
- ❑ **Môi trường thực thi Objective-C bao gồm hai phiên bản “modern” và “legacy”.**
 - Phiên bản “modern” được sử dụng cho các ứng dụng trên iOS và chương trình 64-bit trên MAC 10.5 trở lên, sử dụng Objective-C 2.0.
- ❑ **Có thể tương tác với hệ thống thực thi thông qua mã lệnh Objective-C, phương thức trong lớp NSObject hoặc gọi trực tiếp hàm trong hệ thống.**



Nội dung



1. Sơ lược về ngôn ngữ Objective-C
2. Objective-C và các ngôn ngữ khác
3. Môi trường thực thi Objective-C
4. Môi trường phát triển
 - Bộ công cụ phát triển phần mềm iOS SDK
 - Công cụ phát triển
 - Cài đặt Xcode



4. Môi trường phát triển



- Nền tảng phát triển ứng dụng dùng Objective-C của Apple bao gồm các thành phần sau:**
- Ngôn ngữ lập trình Objective-C
 - Môi trường thực thi Objective-C
 - Bộ thư viện phát triển phần mềm (SDK)
 - Bộ công cụ phát triển phần mềm





4.1 Bộ công cụ phát triển phần mềm iOS SDK

- ❑ **Bộ thư viện phát triển phần mềm SDK (Software Development Kit) bao gồm:**
- Các tập hàm API hay còn gọi là giao diện lập trình ứng dụng (Application Programming Interface)
 - Các thư viện hỗ trợ các hàm chức năng
 - Các tính năng tiện ích tùy thuộc theo từng phiên bản phát triển cho hệ điều hành.



4.2 Công cụ phát triển

- ❑ **Công cụ phát triển**
- Xcode là bộ IDE (Integrated Development Environment) duy nhất được Apple cung cấp.
 - Hỗ trợ đầy đủ các tính năng biên tập chỉnh sửa mã lệnh, xây dựng giao diện, tích hợp máy ảo cho phép biên dịch chạy ứng dụng trực tiếp.
 - Cho phép kiểm lỗi và kiểm thử phần mềm.
 - Xcode phiên bản 4.x dành cho hệ điều hành MAC từ phiên bản 10.8 trở về trước.
 - Xcode phiên bản 5.x dành cho hệ điều hành MAC từ phiên bản 10.8 trở lên và tích hợp SDK cho iOS 7.



4.3 Cài đặt XCode



□ Cài đặt Xcode

- Tương thích với tất cả các dòng máy tính của Apple (Macbook Pro, Macbook Air, Mac mini, iMac)
- Các dòng máy tính khác yêu cầu cài đặt Hackintosh.



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 2. *Kiểu dữ liệu*

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Hệ thống lưu trữ dữ liệu
 - Vấn đề về lưu trữ dữ liệu trong máy tính
 - Vấn đề về sử dụng dữ liệu
 - Vấn đề về đa dạng dữ liệu
2. Các kiểu dữ liệu cơ sở
3. Phép tính toán trên các dữ liệu cơ sở
4. Foundation Framework & Các kiểu dữ liệu trong Objective-C



1.1 Vấn đề về lưu trữ dữ liệu



- **Ứng dụng được xây dựng nhu cầu thực tế và cần lưu trữ thông tin để có thể truy xuất khi cần.**
 - Ví dụ: xây dựng ứng dụng quản lý học viên, cần lưu trữ thông tin học viên như họ tên, ngày tháng năm sinh...
- **Máy tính sử dụng đơn vị cơ bản nhất gọi là Bit, bao gồm hai giá trị cơ bản 0 và 1 tương ứng với tín hiệu điện tắt và bật.**
 - Ví dụ: khi thực thi ứng dụng, máy tính sẽ nhận được một chuỗi dữ liệu như sau: 0001101010100011101001....
- **Đơn vị cao hơn Bit được gọi là Byte, với 1byte = 8bit.**
 - Giá trị thấp nhất của Byte: 00000000 = 0
 - Giá trị cao nhất của Byte: 11111111 = 255
 - => mỗi byte có thể lưu trữ 256 giá trị khác nhau.



1.2 Vấn đề về sử dụng dữ liệu



- Ngôn ngữ máy nhị phân quá phức tạp cho lập trình viên khi sử dụng:
 - Ví dụ để lưu kí tự 'A' ta cần truyền mã nhị phân: **01000001**
- Kiểu dữ liệu cơ sở được sinh ra nhằm đơn giản hóa cách khai báo và sử dụng dữ liệu:
 - Ví dụ để lưu số nguyên có giá trị bằng 1. Ta khai báo:
 - `int i = 1;`
 - **Ý nghĩa:**
 - `int`: kiểu dữ liệu cơ sở số nguyên.
 - `i`: được gọi là tên **Biến**, dùng để truy xuất giá trị khi cần.
 - `=`: phép tính thực hiện gán giá trị cho một **Biến**.
 - `1`: giá trị được lưu trữ bên dưới bộ nhớ cho biến `i`.
 - **Cách hiểu:** khai báo biến `i` với giá trị được lưu trữ bằng 1.



1.3 Vấn đề về đa dạng dữ liệu



- Nhu cầu thực tế cần lưu trữ rất nhiều dạng dữ liệu khác nhau như: chuỗi, số nguyên, số thập phân... với kích thước khác nhau.
 - Ví dụ để lưu trữ thông tin về đặt vé xem phim trong ứng dụng xem phim:
 - Tên phim: Hobbit – Desolation of Smaug (kiểu chuỗi)
 - Ngày chiếu: 03/01/2014 (kiểu ngày tháng)
 - Tiền vé: 135.000 (kiểu số)
 - ...
- Hệ thống các kiểu dữ liệu cơ sở được tạo ra và cấp phát không gian lưu trữ phù hợp cho từng kiểu.
 - Kiểu dữ liệu số: lưu trữ các giá trị kiểu số.
 - Ví dụ: 123, 67.89...
 - Kiểu dữ liệu kí tự: lưu trữ giá trị một kí tự bất kì.
 - Ví dụ: 'A', '1', 'π', '@'...
 - Kiểu dữ liệu luận lý: lưu trữ một trong hai giá trị 0 hoặc 1.





Nội dung

1. Hệ thống lưu trữ dữ liệu
2. Các kiểu dữ liệu cơ sở
 - Kiểu dữ liệu số
 - Kiểu dữ liệu số nguyên
 - Kiểu dữ liệu số thực
 - Kiểu dữ liệu kí tự
 - Kiểu dữ liệu luận lý
3. Phép tính toán trên các dữ liệu cơ sở
4. Các kiểu dữ liệu cơ sở trong Objective-C



2.1 Kiểu dữ liệu số

- **Kiểu dữ liệu số dùng để lưu trữ các dữ liệu số phục vụ tính toán từ các nhu cầu phát sinh thực tế.**
 - Ví dụ: Tính toán số tiền cần thanh toán khi tiến hành đặt 4 vé phim, với mỗi vé giá 135.000. Khi đó: tổng tiền = 4×135.000 .
- **Dữ liệu số cần lưu trữ có thể là một con số có độ lớn bất kỳ, do đó cần nắm rõ bộ nhớ cấp phát cho từng kiểu.**

Kiểu dữ liệu	Không gian lưu trữ
short	2 byte
int	4 byte
float	4 byte
double	8 byte
long	8 byte
long long	16 byte
long double	16 byte





2.1.1 Kiểu dữ liệu số nguyên

- **Kiểu dữ liệu số nguyên được gán giá trị bằng một con số bao hàm cả có dấu và không dấu.**
- **Để có lưu trữ một dữ liệu số nguyên bất kỳ thay vì cấp phát chính xác số lượng byte tương ứng ta sẽ dựa trên mối tương quan của các kiểu dữ liệu.**
 - short <= int <= long <= long long



2.1.1 Kiểu dữ liệu số nguyên

- **Trong một ít trường hợp liên quan đến thuật toán cần sự chính xác tuyệt đối khi đó cần sử dụng dạng dữ liệu đặc biệt của kiểu số nguyên.**
 - (u)int<n>_t: định nghĩa số nguyên có dấu hoặc không dấu có độ dài n tương ứng 1, 2, 4 hoặc 8 byte. (ký tự “u” có nghĩa là số không dấu (unsigned)).
 - int8_t aOneByteInt = 127;
 - uint8_t aOneByteUInt = 255;
 - int16_t aTwoByteInt = 32767;
 - uint16_t aTwoByteUInt = 65535;
 - int32_t aFourByteInt = 2147483647;
 - uint32_t aFourByteUInt = 4294967295;
 - int64_t anEightByteInt = 9223372036854775807;
 - uint64_t anEightByteUInt = 18446744073709551615;





2.1.1 Kiểu dữ liệu số nguyên

- (u)int_least<n>_t: định nghĩa số nguyên qui định kích thước nhỏ nhất
 - int_least8_t aTinyInt = 127;
 - uint_least8_t aTinyUInt = 255;
 - int_least16_t aMediumInt = 32767;
 - uint_least16_t aMediumUInt = 65535;
 - int_least32_t aNormalInt = 2147483647;
 - uint_least32_t aNormalUInt = 4294967295;
 - int_least64_t aBigInt = 9223372036854775807;
 - uint_least64_t aBigUInt = 18446744073709551615;
- (u)intmax_t: định nghĩa số nguyên qui định kích thước lớn nhất hệ thống có thể xử lý.
 - intmax_t the BiggestInt = 9223372036854775807;
 - uintmax_t the BiggestUInt = 18446744073709551615;



2.1.1 Kiểu dữ liệu số nguyên

- Để truy xuất hiển thị dữ liệu số nguyên ta dùng kí hiệu %d cho số nguyên có dấu và %u cho số nguyên không dấu.
- Truy xuất có số nguyên có dấu:

```
// Khai báo kiểu dữ liệu số nguyên giá trị khởi tạo bằng 0
int soNguyen = 0;

// Xuất dòng chữ thông báo nhập số nguyên
NSLog(@"Nhập một số nguyên: ");

// Ghi nhận giá trị vào biến soNguyen
scanf("%d", &soNguyen);

// Xuất kết quả của số nguyên
NSLog(@"Số nguyên ghi nhận: %d", soNguyen);
```





2.1.1 Kiểu dữ liệu số nguyên

- Truy xuất số nguyên không dấu:

```
// Khai báo kiểu dữ liệu số nguyên giá trị khởi tạo bằng 0
unsigned int soNguyen = 0;

// Xuất dòng chữ thông báo nhập số nguyên
NSLog(@"Nhập một số nguyên: ");

// Ghi nhận giá trị vào biến soNguyen
scanf("%u", &soNguyen);

// Xuất kết quả của số nguyên
NSLog(@"Số nguyên ghi nhận: %u", soNguyen);
```



2.1.1 Kiểu dữ liệu số nguyên

- Bảng định dạng truy xuất:

Kiểu dữ liệu	Định dạng truy xuất có dấu	Định dạng truy xuất không dấu
short	hd	hu
int	d	u
long	ld	lu
long long	lld	llu





2.1.2 Kiểu dữ liệu số thực

- Giống với kiểu dữ liệu số nguyên tuy nhiên có thể lưu trữ được các số thập phân không phân biệt có dấu và không dấu bao gồm ba định dạng:
 - float <= double <= long double.

Kiểu dữ liệu	Không gian lưu trữ	Định dạng khai báo	Định dạng truy xuất
float	4 byte	f	f
double	8 byte	N/A	f
long double	16 byte	L	Lf



2.1.2 Kiểu dữ liệu số thực

- Để truy xuất số lượng chữ số trước và sau dấu chấm thập phân sử dụng định dạng <n>.<n>
 - Ví dụ:

```
float aFloat = -21.096758;
 NSLog(@"%@", aFloat);
```

- Kết quả in ra màn hình:

- -21.09





2.2 Kiểu ký tự

- **Kiểu dữ liệu kí tự được sử dụng để lưu các giá trị bao gồm: chữ số, kí hiệu, chữ cái và một số biểu tượng khác.**
- **Kiểu dữ liệu kí tự được cấp phát duy nhất một byte để lưu trữ dữ liệu (tương đương 256 giá trị) và do đó chỉ được phép lưu một kí tự duy nhất.**
 - Ví dụ khai báo dữ liệu kiểu kí tự:
 - `char c = 'A';` // Đúng
 - `char c = 'ABC';` // Sai
- **Kiểu dữ liệu ký tự có thể khai báo có dấu và không dấu.**
 - Ví dụ:
 - `char a = 'A'`
 - `unsigned char a = 98`



2.2 Kiểu ký tự

- **Truy xuất giá trị của kiểu kí tự:**
 - `%c` : truy xuất giá trị có dấu
 - `%hu` : truy xuất giá trị không dấu
 - `hhd` : truy xuất thứ tự trong bảng mã ASCII
 - `%s` : truy xuất giá trị con trả kiểu kí tự

```
char aChar = 'A';
unsigned char anUChar = 255;

 NSLog(@"%@", aChar);      // A
 NSLog(@"%@", aChar);      // 65
 NSLog(@"%@", anUChar);    // 255
```





2.2 Kiểu ký tự

- ❑ **Vấn đề:** cần lưu trữ thông tin là một chuỗi nhiều kí tự.
 - Ví dụ: cần lưu trữ tên phim: "Hobbit 2 – Desolation of Smaug"
- ❑ **Thực hiện khai báo kiểu kí tự ở dạng “con trỏ”, chuỗi lưu trữ sẽ được quản lý bởi “con trỏ” này.**
 - Ví dụ:

```
char* a = "Hobbit 2 - Desolation of Smaug";
NSLog(@"%@", a);
```



2.3 Kiểu luận lý

- ❑ **Kiểu luận lý (bool) được hiểu ở ngôn ngữ máy với hai giá trị 0 hoặc 1, ở môi trường lập trình bao gồm hai giá trị true và false.**
- ❑ **Các biến bool thường được dùng để kiểm tra giá trị của một đối tượng hoặc điều hướng trong các câu điều kiện.**
 - Ví dụ về cách khai báo và truy xuất:

```
bool isBool = true;
NSLog(@"%@", isBool);
```

- Kết quả in ra màn hình:

- 1



Nội dung



1. Hệ thống lưu trữ dữ liệu
2. Các kiểu dữ liệu cơ sở
3. Phép tính toán trên các dữ liệu cơ sở
 - Toán tử tính toán
 - Toán tử so sánh
 - Toán tử Logic
4. Foundation Framework & Các kiểu dữ liệu cơ sở trong Objective-C



3.1 Toán tử tính toán



- Toán tử tính toán được sử dụng trong các quá trình xử lý dữ liệu dựa các kết quả tính toán số học theo ý nghĩa thực thi.
- Phép tính toán cơ bản nhất bao gồm: 2 mệnh đề dữ liệu, một phép gán và một toán tử.
 - Ví dụ: `int i = 1 + 2;`
 - Ý nghĩa:
 - Mệnh đề dữ liệu 1: `int i`
 - Mệnh đề dữ liệu 2: `1 + 2`
 - Toán tử sử dụng: `+`
 - Phép gán: `=`
 - Kết quả biến `i` khi in ra màn hình:
 - `3`





3.1 Toán tử tính toán

Các toán tử tính toán

Toán tử tính toán	Ý nghĩa
+	Phép cộng
-	Phép trừ
*	Phép nhân
/	Phép chia
%	Phép chia lấy phần dư

Ví dụ:

```
NSLog(@"%@", 6 + 2 = %d, 6 + 2); // 8
NSLog(@"%@", 6 - 2 = %d, 6 - 2); // 4
NSLog(@"%@", 6 * 2 = %d, 6 * 2); // 12
NSLog(@"%@", 6 / 2 = %d, 6 / 2); // 3
NSLog(@"%@", 6 %% 2 = %d, 6 % 2); // 0
```



3.1 Toán tử tính toán

Các phép tính toán được thực hiện theo độ ưu tiên phép tính toán số học và cơ chế từ trái qua phải.

- Ví dụ: int i = 3 * 4 - 6 / 2 % 3;
- Kết quả: i = 12

Để ưu tiên các phép tính toán ta thực hiện “đóng ngoặc” các phép tính toán đó.

- Ví dụ: int i = 3 * 4 - 6 / (2 % 3);
- Kết quả: i = 9





3.1 Toán tử tính toán

- ❑ Đối với phép tính cộng và trừ được bổ sung hai toán tử ++ và -- (chỉ sử dụng cho kiểu dữ liệu số)
 - ++ : cho phép tăng một đơn vị
 - -- : cho phép giảm một đơn vị
- ❑ Ví dụ:

```
int soNguyen = 7;
NSLog(@"soNguyen + 1 = %d", ++soNguyen);      // 8
```


`float soThuc = 11.35;
NSLog(@"soNguyen - 1 = %.2f", --soThuc); // 10.35`
- ❑ Lưu ý: vị trí đặt các toán tử trước và sau biến dữ liệu có ý nghĩa khác nhau, ví dụ:
 - ++soNguyen: tăng giá trị số nguyên sau đó ghi kết quả (ví dụ trên)
 - soNguyen++ : ghi số nguyên sau đó tăng giá trị



3.1 Toán tử tính toán

- ❑ Có thể thực hiện tính toán và gán giá trị trực tiếp thông qua các toán tử rút gọn.

Toán tử tính toán	Ví dụ	Ý nghĩa
+	x += y	$x = x + y$
-	x -= y	$x = x - y$
*	x *= y	$x = x * y$
/	x /= y	$x = x / y$
%	x %= y	$x = x \% y$

- ❑ Phép rút gọn có thể hiểu là biến ở mệnh đề dữ liệu 1 sẽ tăng hoặc giảm một lượng bằng với giá trị của biến ở mệnh đề dữ liệu 2.





3.2 Toán tử so sánh

- **Toán tử so sánh** được sử dụng trong các trường hợp kiểm tra giá trị giữa 2 biến cùng kiểu dữ liệu. Kết quả trả về được đại diện bằng một biến bool (true hoặc false).

Toán tử so sánh	Ý nghĩa
==	Bằng (giá trị)
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
!=	Không bằng (giá trị)



3.3 Toán tử Logic

- **Toán tử Logic** được dùng kết hợp với các cấu trúc điều khiển để kiểm tra giá trị biến hoặc tập hợp mệnh đề.

Toán tử so sánh	Ví dụ	Ý nghĩa	Kết quả
!	!a	Phủ định giá trị biến	Trả về true nếu giá trị khác a và ngược lại
&&	<mệnh đề 1> && <mệnh đề 2>	Kiểm tra giá trị mệnh đề 1 kết hợp giá trị mệnh đề 2	Trả về true nếu cả hai mệnh đề có giá trị là true
	<mệnh đề 1> <mệnh đề 2>	Kiểm tra giá trị mệnh đề 1 kết hợp giá trị mệnh đề 2	Trả về false nếu một trong hai mệnh đề so sánh trả về false



Nội dung



1. Hệ thống lưu trữ dữ liệu
2. Các kiểu dữ liệu cơ sở
3. Phép tính toán trên các dữ liệu cơ sở
4. Foundation Framework & Các kiểu dữ liệu cơ sở Objective-C
 - Foundation Framework
 - Các kiểu dữ liệu cơ sở Objective-C
 - NSInteger
 - NSString



4.1 Foundation FrameWork



- Bộ Foundation Framework định nghĩa một tập cơ sở các lớp trong Objective-C bổ sung các đối tượng dữ liệu tiện ích bên cạnh các kiểu dữ liệu cơ sở.
- Mục đích sử dụng chính:
 - Cung cấp các lớp chức năng cơ bản (chuyển đổi, lưu đổi tượng...)
 - Xây dựng ứng dụng dễ hơn trong cấp phát và thu hồi bộ nhớ
 - Hỗ trợ các định dạng chuỗi, duy trì và phân chia đối tượng
 - Nâng cao tính linh hoạt trong xây dựng tầng hệ thống độc lập
 - True: Đối tượng được hiển thị
 - False: Đối tượng bị ẩn
- Lớp NSObject là làm gốc trong bộ Foundation Framework





4.1 Foundation FrameWork

- Bộ Foundation Framework được chia thành các lớp riêng rẽ, mỗi lớp được gom nhóm dựa trên chức năng, bao gồm:

- | | |
|---|--|
| <ul style="list-style-type: none">• Values Objects• XML• Strings• Collections• Predicate• Operating-System Service• File System• URL | <ul style="list-style-type: none">• Interprocess Communication• Locking/Threading• Notification• Archiving and Serialization• Objective C Language Service |
|---|--|



4.2.1 NSInteger

- Kiểu dữ liệu số nguyên trong Objective-C, cho phép lưu trữ các giá trị số nguyên (không dấu - NSUInteger).
- Được khai báo kiểu struct, không có hàm cấp phát.
- Được sử dụng để tăng tính hiệu quả của ứng dụng trên các kiến trúc khác nhau.
 - int: phân biệt về cấp phát khi chạy trên nền 32bit – 64bit
 - NSInteger: không phân biệt cấp phát trên các nền khác nhau.
- Được sử dụng rộng rãi trong các phương thức trả về giá trị số nguyên trong bộ Cocoa & CocoTouch (UITableView, PageControl).
- Dễ dàng chuyển đổi sang các kiểu dữ liệu trong Objective-C (NSNumber, NSValue, NSString...)



4.2.2 NSString



❑ Các cách tạo một chuỗi kiểu NSString.

- Tạo bằng cách gán với chuỗi.
- Tạo đối tượng theo định dạng

```
NSString *module= @"bài 2";
NSString *say=
[NSString stringWithFormat:@"Xin chào đây là %@", module];
```



4.2.2 NSString



❑ So sánh chuỗi dùng isEqualToString để so sánh 2 chuỗi bằng nhau thay vì ==.

- So sánh 2 chuỗi bằng nhau.
- Dùng hasPrefix so sánh với cụm đầu của chuỗi.
- Dùng hasSuffix so sánh với cụm cuối của chuỗi.

```
NSString *car = @"Porsche Boxster";
if ([car isEqualToString:@"Porsche Boxster"]) {
    NSLog(@"Đây là xe Porsche Boxster");
    // kết quả: đây là xe Porsche Boxster
} if ([car hasPrefix:@"Porsche"]) {
    NSLog(@"Đây là xe Porsche");
    // kết quả: Đây là xe Porsche
} if ([car hasSuffix:@"Carrera"]) {
    NSLog(@"Đây là xe Carrera");
    // không hiển thị vì Carrera không là cụm từ cuối cùng của biến car.
}
```





4.2.2 NSString

- So sánh hai chuỗi khác nhau dùng compare.

- So sánh chuỗi bằng NSComparisonResult.
- Kết quả trả về sẽ là: NSOrderedSame, NSOrderedAscending, NSOrderedDescending

```
NSString *otherCar = @"Ferrari";
NSComparisonResult result = [car
compare:otherCar];
if (result == NSOrderedAscending) {
    NSLog(@"Ký tự 'P' đứng trước ký tự 'F'");
} else if (result == NSOrderedSame) {
    NSLog(@"2 chuỗi này giống nhau.");
} else if (result == NSOrderedDescending) {
    NSLog(@"Ký tự 'P' đứng sau ký tự 'F'");
} // kết quả: Ký tự 'P' đứng sau ký tự 'F'
```



4.2.2 NSString

- Ta có thể nối chuỗi bằng các phương thức sau:

- Dùng stringByAppendingString để nối 2 chuỗi với nhau và trả về 1 chuỗi.
- Dùng stringByAppendingFormat để tạo 1 định dạng nối các chuỗi.

```
NSString *make = @"Ferrari";
NSString *model = @"458 Spider";
NSString *car = [make
stringByAppendingString:model];
NSLog(@"%@", car);
// kết quả: Ferrari458 Spider
car = [make stringByAppendingFormat:@" %@", model];
NSLog(@"%@", car);
// kết quả: Ferrari 458 Spider
```





4.2.2 NSString

□ Ta có thể tìm chuỗi trong chuỗi bằng NSRange

- Dùng NSRange để tìm chuỗi. Nếu kết quả trả về là NSNotFound thì không tìm thấy.

```
NSString *car = @"Maserati GranCabrio";
NSRange searchResult = [car
rangeOfString:@"Cabrio"];
if (searchResult.location == NSNotFound) {
    NSLog(@"Không tìm thấy");
} else {
    NSLog(@"%@", @"'Cabrio' bắt đầu tại %lu và có %lu ký tự
phù hợp",
(unsigned long)searchResult.location, // 13
(unsigned long)searchResult.length); // 6
}
```



4.2.2 NSString

□ Trong nhiều trường hợp ta cần chia chuỗi thành nhiều chuỗi nhỏ hơn.

- Dùng substringToIndex để lấy từ đầu chuỗi đến vị trí mong muốn.
- Dùng substringWithRange để lấy chuỗi trong khoảng.
- componentsSeparatedByString để cắt chuỗi tại những ký tự được chọn.

```
NSString *carTemp = @"Maserati GranTurismo";
NSLog(@"%@", [carTemp substringToIndex:8]);
// Maserati
NSRange range = NSMakeRange(9, 4);
NSLog(@"%@", [carTemp
substringWithRange:range]);
// Gran
NSString *models = @"Porsche,Ferrari,Maserati";
NSArray *modelsAsArray = [models
componentsSeparatedByString:@",,"];
NSLog(@"%@", [modelsAsArray objectAtIndex:1]); // Ferrari
```





4.2.2 NSString

- ❑ NSString hỗ trợ 3 phương thức thay đổi hoa/thường cho chuỗi.

- Dùng lowercaseString để chuyển chuỗi thành ký tự thường.
- Dùng uppercaseString để chuyển chuỗi thành ký tự in hoa.
- Dùng capitalizedString để in hoa ký tự đầu tiên của mỗi từ.

```
NSString *tempStr = @"đÂY là MỘT chuỗi";
 NSLog(@"%@", [tempStr lowercaseString]);
//đây là một chuỗi
 NSLog(@"%@", [tempStr uppercaseString]);
//ĐÂY LÀ MỘT CHUỖI
 NSLog(@"%@", [tempStr capitalizedString]);
//Đây Là Một Chuỗi
```



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 3. Xây dựng ứng dụng iOS

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Kiến trúc ứng dụng iOS

- Cocoa Touch
- Media
- Core Service
- Core OS

2. Qui trình xây dựng ứng dụng



1. Kiến trúc ứng dụng iOS



❑ **Ứng dụng thực thi trên môi trường iOS giao tiếp với thiết bị phần cứng thông qua tập các lớp giao diện từ ngôn ngữ cấp thấp đến ngôn ngữ cấp cao.**

❑ **Kiến trúc ứng dụng iOS bao gồm:**

- Cocoa Touch
- Media
- Core Service
- Core OS





1.1 Cocoa Touch

- ❑ Cocoa Touch là bộ phát triển ứng dụng iOS dựa trên nền tảng Cocoa tập trung vào các xử lý chạm màn hình.
- ❑ Các bộ nền tảng trong Cocoa Touch cung cấp bao gồm:
 - Address Book UI
 - Event Kit UI
 - Game Kit
 - iAd
 - Map Kit
 - Message UI
 - Twitter
 - UIKit



1.1 Cocoa Touch

- ❑ Sử dụng các tính năng nổi bật của Cocoa Touch trong xây dựng ứng dụng:
 - AirDrop
 - TextKit
 - UIView
 - MultiTasking
 - Auto Layout
 - StoryBoard
 - UI State
 - Apple Push Notification Service
 - Local Notifications
 - Gesture Recognizers
 - Standard System View Controllers





1.2 Media

- ❑ Media là lớp cơ sở cho phép lập trình viên có thể thực hiện các xây dựng các ứng dụng đáp ứng hầu hết các tính năng đa truyền thông về âm thanh, hình ảnh, chuyển động...
- ❑ Các bộ nền tảng được Media cung cấp bao gồm:
 - Asset Library
 - AV Foundation
 - Core Audio
 - Core Graphic
 - Core Image
 - Core Text
 - Core Video
 - Game Controller
 - GLKit
 - Image I/O
 - Media Accessibility
 - MediaPlayer
 - OpenAL
 - OpenGL ES
 - Quartz Core
 - Sprite Kit



1.2 Media

- ❑ Sử dụng các tính năng nổi bật của Media trong xây dựng ứng dụng:
 - Graphics: TextKit, Core Text...
 - Audio: AAC, Apple Lossless, A-Law, IMA/ADPCM...
 - Video: H.264, MPEG-4...
 - AirPlay





1.3 Core Service

- ❑ Core Service cung cấp hầu hết các dịch vụ cơ bản được sử dụng trong ứng dụng như định vị, mạng kết nối, điện toán đám mây...
- ❑ Các bộ nền tảng được Core Service cung cấp bao gồm:
 - Account
 - Address Book
 - Ad support
 - CF Network
 - Core Data
 - Core Foundation
 - Core Location
 - Core Media
 - Core Motion
 - Core Telephony
 - Event Kit
 - Foundation
 - JavaScript Core
 - Mobile Core Service
 - Multipeer
 - Newsstand Kit
 - Pass Kit
 - Quick Look
 - Safari
 - Store
 - System Configuration



1.3 Core Service

- ❑ Core Service cung cấp các tính năng cho phép xây dựng các ứng dụng có khả năng kết nối mạnh mẽ đến các dịch vụ Apple và cả dịch vụ được cung cấp từ bên thứ ba.
 - Kết nối Peer-to-Peer
 - Lưu trữ điện toán đám mây iCloud
 - Bộ quản lý bộ nhớ ARC (Automatic Reference Counting)
 - Khóa đối tượng
 - Bảo mật dữ liệu
 - Hỗ trợ chia sẻ tập tin
 - Xử lý dữ liệu tập trung GDC (Grand Central Dispatch)
 - Thanh toán trực tiếp từ ứng dụng (In-App purchase)
 - Lưu trữ cơ sở dữ liệu với SQLite
 - Hỗ trợ đọc, ghi XML





1.4 Core OS

- ❑ Core OS là lớp truy xuất cấp thấp đến phần cứng thiết bị, thường được sử dụng gián tiếp thông qua các lớp khác. Khi đó các khai báo về trao đổi thông tin và bảo mật là điều bắt buộc.
- ❑ Các bộ nền tảng Core OS cung cấp bao gồm:
 - Accelerate
 - Core Bluetooth
 - External Accessory
 - Generic Security Services
 - Security
 - System
 - 64-Bit Support



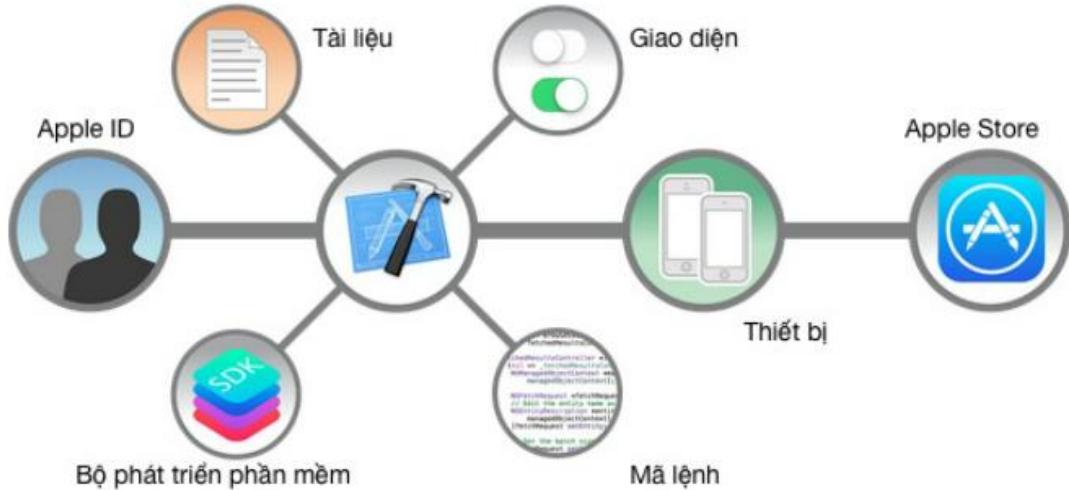
Nội dung

1. Kiến trúc iOS
2. Qui trình xây dựng ứng dụng
 - Apple Store & Apple ID
 - Ý tưởng và định hướng phát triển
 - Thiết kế giao diện
 - Xây dựng tương tác
 - Tổ chức dữ liệu





2. Qui trình xây dựng ứng dụng



2.1 Apple Store & Apple ID

- ❑ Apple Store cho phép các bên thứ ba cung cấp các ứng dụng dựa trên nền tảng phát triển công nghệ của Apple và các điều khoản, mỗi tổ chức hoặc cá nhân cần có Apple ID để có thể tải ứng dụng từ Apple Store hoặc đưa ứng dụng lên đó.
- ❑ Đăng ký Apple ID từ địa chỉ:
<https://developer.apple.com/register/index.action>

Create Apple ID

Please create a new Apple ID if you are enrolled in the iOS Developer Enterprise Program, have an iTunes Connect account, or prefer to have an Apple ID dedicated to your business transactions.

[Create Apple ID](#)



2.2 Ý tưởng và định hướng phát triển

Lập trình iOS (2014) – Bài 3. Xây dựng ứng dụng iOS

14

2.3 Thiết kế giao diện

- Trong thiết kế giao diện ứng dụng iOS, View được xem là thành phần quan trọng nhất. View cho phép cấu thành nên tất cả các thuộc tính một màn hình giao diện cần có.
- Khi thiết kế ứng dụng, cần lên trước kế hoạch các đối tượng View sẽ sử dụng, vì nó ảnh hưởng toàn bộ quá trình phát triển về sau.

Lập trình iOS (2014) – Bài 3. Xây dựng ứng dụng iOS

15



2.3 Thiết kế giao diện

❑ **Tùy loại đối tượng View có những thuộc tính khác nhau, tuy nhiên có thể gom thành 7 nhóm sau:**

- Nội dung
- Tập hợp
- Điều khiển
- Thanh điều khiển
- Nhập liệu
- Nhóm điều khiển
- Thông báo



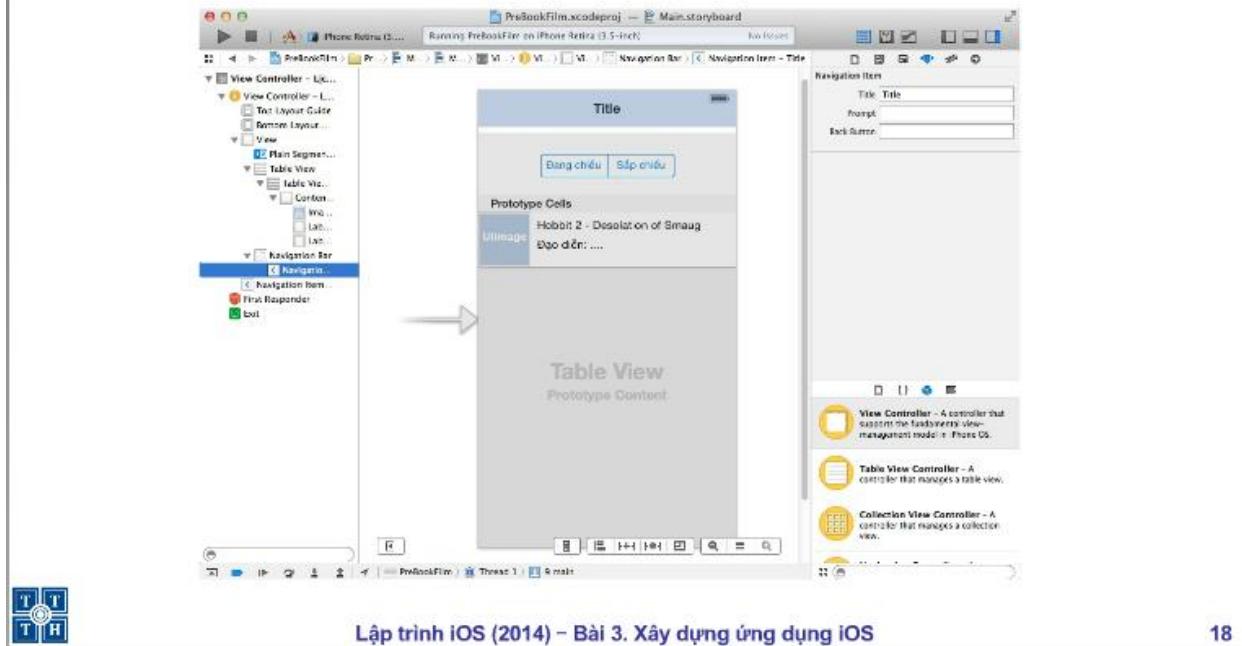
2.3 Thiết kế giao diện

Nhóm đối tượng	Chức năng	Ví dụ
Nội dung	Hiển thị một nội dung bất kì dựa trên thuộc tính đối tượng.	ImageView, Label
Tập hợp	Hiển thị tập hợp hoặc nhóm các đối tượng.	CollectionView, TableView
Điều khiển	Cho phép thực thi tương tác hoặc hiển thị thông tin.	Button, Slider, Switch
Thanh điều khiển	Gom nhóm các tương tác hoặc điều hướng.	Toolbar, Tabbar, Navigationbar
Nhập liệu	Nhận dữ liệu được nhập từ người dùng	Searchbar, TextView
Nhóm điều khiển	Chứa các điều khiển khác nhau	View, ScrollView
Thông báo	Bao gồm các điều khiển cho phép hiển thị thông báo trong ứng dụng	ActionSheet, AlertView



2.3 Thiết kế giao diện

- ❑ Xcode hỗ trợ Interface Builder cho phép lập trình viên xây dựng giao diện một cách tương minh.

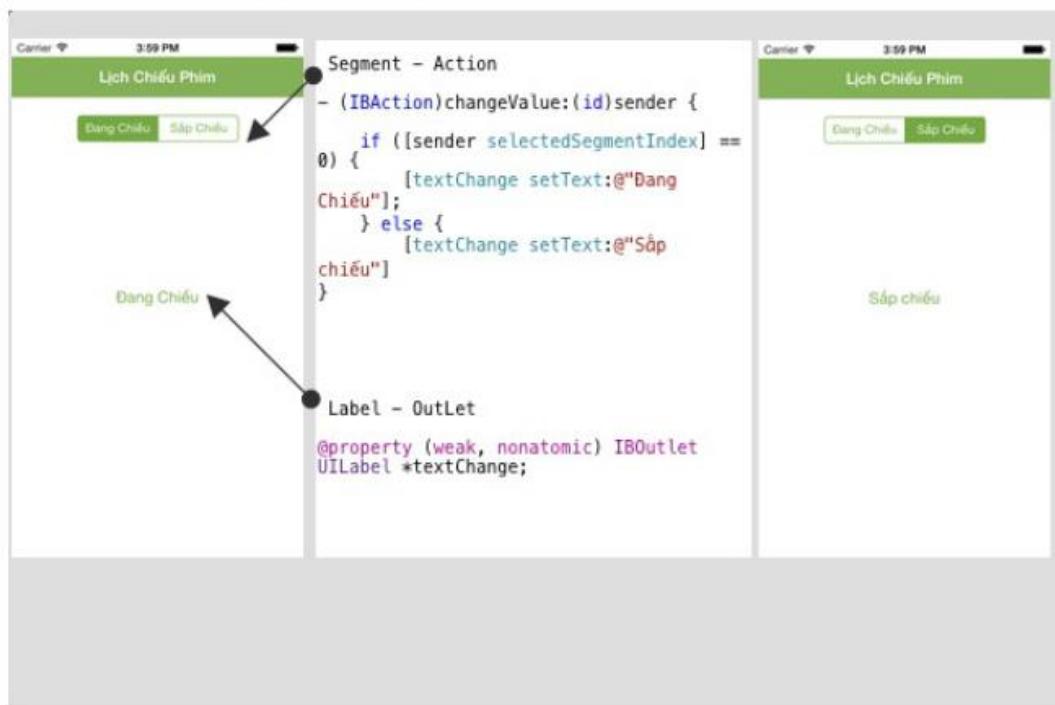


2.4 Xây dựng tương tác

- ❑ Dựa trên mô hình giao diện trên Interface Builder, cần xây dựng thuộc tính tương tác đặc trưng cho từng đối tượng View.
- ❑ **Tương tác trong các điều khiển bao gồm hai thuộc tính chung:**
 - Action: mô tả đoạn mã lệnh được thực hiện khi có sự kiện tương ứng với điều khiển được gọi.
 - Ví dụ: đoạn mã lệnh chuyển sang màn hình chi tiết phim khi có sự kiện nhấn lên một phim bất kỳ trong danh sách phim.
 - Outlet: mô tả cách thức phản hồi làm thay đổi nội dung của điều khiển trên giao diện khi có tương tác xảy ra.
 - Ví dụ: danh sách phim được thay đổi nội dung "Đang chiếu" sang "Sắp chiếu" và ngược lại, khi người dùng lựa chọn trên điều khiển Segment.



2.4 Xây dựng tương tác

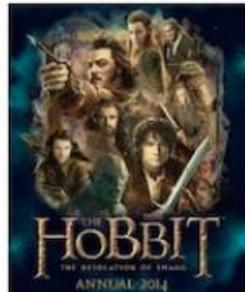


2.4 Tổ chức dữ liệu

- ❑ **Tổ chức dữ liệu cho phép ta mô tả các đối tượng ở thế giới thực vào mã nguồn.**

- Ví dụ: ứng dụng xem lịch chiếu phim cần mô tả đối tượng Phim bao gồm các thuộc tính:

- Tên phim
- Thể loại
- Đạo diễn
- Ngày chiếu
- Thời lượng
- Diễn viên
- Nội dung tóm tắt
- ...



Đặt vé Đánh giá

Đạo diễn: Peter Jackson
 Thời lượng: 2 giờ và 41 phút
 Thể loại: Phiêu Lưu, Tâm lý, KH Viễn Tưởng/
 Thần Thoại
 Khởi chiếu: 03/01/2014

Phần 2 tiếp tục kể về cuộc phiêu lưu của nhân vật chính Bilbo Baggins và cuộc hành trình với phù thuỷ Gandalf cùng mười ba người lùn dẫn đầu bởi Thorin Oakenshield, họ mang trên mình một nhiệm vụ sứ thi để đòi lại Lonely Mountain và Vương quốc người lùn đã mất của Erebor.



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 4. Cấu trúc điều khiển

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Cấu trúc điều khiển

- Cấu trúc điều kiện IF
- Cấu trúc điều kiện SWITCH...CASE
- Cấu trúc vòng lặp FOR
- Cấu trúc vòng lặp WHILE & DO...WHILE
- Break & Continue trong cấu trúc vòng lặp

2. Biến định nghĩa



1.1 Cấu trúc điều kiện IF



□ **Cấu trúc điều kiện IF được sử dụng phổ biến trong điều khiển thực thi ứng dụng thông qua việc kiểm tra giá trị của các biến. Khi đó giá trị trả về của câu điều kiện hoặc là true hoặc là false.**

- Cú pháp khai báo:

```
If ( <điều kiện> )  
    [đoạn mã thực thi];
```

- Ví dụ:

```
int i = 1;  
if ( i < 2 )  
    NSLog(@"Biến i có giá trị nhỏ hơn 2");
```

- Ý nghĩa:

▪ Thực hiện khai báo giá trị biến i có giá trị bằng 1, sau đó dùng câu điều kiện để kiểm tra giá trị có nhỏ hơn 2 hay không. Nếu có in ra màn hình câu thông báo, nếu không bỏ qua mã lệnh.





1.1 Cấu trúc điều kiện IF

- Cấu trúc điều kiện IF chỉ thực thi khi câu điều kiện true, do đó để giải quyết khi câu điều kiện false, dùng cú pháp sau:

- Cú pháp khai báo:

```
If ( <điều kiện> ) {  
    [đoạn mã thực thi khi câu điều kiện đúng];  
} else {  
    [đoạn mã thực thi khi câu điều kiện sai];  
}
```

- Ví dụ:

```
int i = 1;  
if ( i < 2 ) {  
    NSLog(@"Biến i có giá trị nhỏ hơn 2");  
} else {  
    NSLog(@"Biến i có giá trị lớn hơn hoặc bằng 2");  
}
```



1.1 Cấu trúc điều kiện IF

- Cấu trúc điều kiện IF có thể sử dụng ở dạng rút gọn như sau:

- Cú pháp khai báo:

```
<điều kiện> ? <mệnh đề true> : <mệnh đề false>
```

- Ý nghĩa: nếu <điều kiện> đúng thực hiện <mệnh đề true>, ngược lại thực hiện <mệnh đề false>.

- Ví dụ:

```
int i = 1;  
NSLog( i < 2 ? @"Biến i có giá trị nhỏ hơn 2" :  
        @"Biến i có giá trị lớn hơn hoặc bằng 2");
```



1.2 Cấu trúc điều kiện SWITCH...CASE



- Cấu trúc điều kiện SWITCH...CASE là dạng so sánh điều kiện mở rộng cho phép dễ dàng kiểm tra giá trị trong nhiều trường hợp hơn.

Lưu ý: SWITCH...CASE chỉ sử dụng cho giá trị số nguyên.

- Cú pháp khai báo:

```
switch (<điều kiện> ) {
    case <giá trị 1>:
        [đoạn mã thực thi];
        break;

    ....
    case <giá trị n>:
        [đoạn mã thực thi];
        break;
    default:
}
```

- Ý nghĩa: kiểm tra <điều kiện> và thực thi đoạn mã thực thi theo <giá trị n>, giá trị default dùng cho các trường hợp ngoại lệ.



1.2 Cấu trúc điều kiện SWITCH...CASE



- Ví dụ về cách sử dụng SWITCH...CASE.

- Giả sử ở đây ta có loại vé bao gồm:

- vé thường = 1; giá 135000/vé
- vé VIP = 2; giá 155000/vé
- Thực hiện tính tổng tiền dựa trên loại vé và số lượng vé.

```
float tongTien = 0;
switch (loaiVe) {
    case 1:
        tongTien = 135000*soLuongVe;
        break;
    case 2:
        tongTien = 155000*soLuongVe;
        break;
    default:
        NSLog(@"Loại vé không hợp lệ");
        break;
}
```





1.3 Cấu trúc vòng lặp FOR

- ❑ Cấu trúc vòng lặp FOR được sử dụng để thực hiện lặp một đoạn mã lệnh thao tác giống nhau.

- Cú pháp khai báo:

```
for ([khởi tạo điều kiện lặp] ; [điều kiện dừng] ; [điều kiện lặp kế tiếp])  
{  
    [đoạn mã thực thi];  
}
```

- Ý nghĩa:

- Khởi tạo điều kiện lặp: cho phép khởi giá trị ban đầu bắt đầu lặp
 - Điều kiện dừng: thực hiện dừng vòng lặp dựa trên điều kiện đã thiết lập
 - Điều kiện lặp kế tiếp: cho phép thực hiện tiếp vòng lặp ở giá trị kế tiếp



1.3 Cấu trúc vòng lặp FOR

- ❑ Ví dụ về vòng lặp FOR

- Cho phép người dùng nhập thông tin phim dựa theo số lượng phim cho trước

```
int soLuongPhim = 4;  
for (int i = 0 ; i < soLuongPhim; i++)  
{  
    printf("Nhập tên phim: ");  
    char tenPhim[40];  
    scanf("%s", tenPhim);  
}
```

- Ý nghĩa:

- int i = 0: khởi tạo giá trị số nguyên ban đầu i = 0
 - i < soLuongPhim: khi i tăng kiểm tra giá trị nhỏ hơn giá trị của soLuongPhim hay không
 - i++: tăng i lên một giá trị





1.4 Cấu trúc vòng lặp WHILE

- Cấu trúc vòng lặp WHILE có chức năng tương tự FOR, tuy nhiên WHILE thường dùng trong trường hợp không rõ điều kiện dừng.

- Cú pháp khai báo:

```
while (<điều kiện lặp>)
{
    [đoạn mã thực thi];
}
```

- Ví dụ: nhập thông tin phim cho đến khi người dùng thoát chương trình.

```
int i = 1;
while (i == 1)
{
    printf("Nhập tên phim: ");
    char tenPhim[40];
    scanf("%s", tenPhim);

    printf("1. Nhập tiếp ? \n2. Thoát chương trình? : ");
    scanf("%d", &i);
}
```



1.4 Cấu trúc vòng lặp DO...WHILE

- Cấu trúc vòng lặp DO...WHILE là dạng mở rộng của WHILE, cho phép thực hiện thực thi mã lệnh trước sau đó mới kiểm tra giá trị để dừng.

- Cú pháp khai báo:

```
do {
    [đoạn mã thực thi];
} while (<điều kiện lặp>);
```

- Ví dụ: nhập thông tin phim cho đến khi quá số lượng phim cho phép.

```
int soLuongPhim = 0;
do {
    printf("Nhập tên phim: ");
    char tenPhim[40];
    scanf("%s", tenPhim);
    soLuongPhim++;
} while (soLuongPhim < 4);
```



1.5 Break & Continue trong cấu trúc vòng lặp



- Break và Continue là hai từ khóa thường được sử dụng kết hợp với cấu trúc điều kiện cho phép điều khiển cấu trúc vòng lặp.

- Break: dừng và lập tức thoát vòng lặp
- Continue: bỏ qua điều kiện lặp hiện tại
- Ví dụ: nhập thông phim cho đến khi người thoát chương trình

```
int i = 1; char tenPhim[40];  
while (true)  
{  
    printf("Nhập tên phim: ");  
    scanf("%s", tenPhim);  
    printf("1. Nhập tiếp ? \n2. Thoát chương trình? : ");  
    scanf("%d", &i);  
    if (i == 1)  
        break;  
}
```



Nội dung



1. Cấu trúc điều khiển

2. Biến định nghĩa

- Macro
- Typedef
- Struct
- Enum





2.1 Macro

- ❑ Macro là dạng biến khai báo cho phép định nghĩa biến hằng sử dụng chung cho toàn bộ dự án. Biến Macro nên được khai báo ngoài lớp hoặc Interface.

- Cú pháp khai báo:

```
#define <Tên biến> <Giá trị biến>
```

- Ví dụ: khai báo 2 loại vé và phụ thu thêm cho ghế ở rạp ATMOS

```
#define VE_THUONG 125000
```

```
#define VE_VIP 155000
```

```
#define ATMOS(giaVe) (giaVe + 15000)
```

- Sử dụng: tính tổng tiền vé trong trường hợp đặt vé VIP ở rạp ATMOS

```
int giaVe = VE_VIP;
```

```
int soLuongVe = 4;
```

```
NSLog(@"Tổng tiền vé: %d", soluongVe*ATMOS(giaVe) )
```



2.2 Typedef

- ❑ Typedef được dùng định nghĩa kiểu dữ liệu mới hoặc tái định nghĩa kiểu dữ liệu có sẵn giúp mã nguồn mang tính tường minh hơn.

- Cú pháp khai báo:

```
typedef <Kiểu dữ liệu> <Tên kiểu dữ liệu>
```

- Ví dụ: ta có thể định nghĩa kiểu dữ liệu giá vé cho từng phim

```
typedef unsigned GiaVe;
```

- Sử dụng: khai báo giá vé cho các phim khác nhau

```
GiaVe hobbit = 215000;
```

```
GiaVe ronin = 135000;
```

```
GiaVe frozen = 155000;
```

- ❑ Typedef thường được sử dụng để khai báo struct hoặc enum.





2.3 Struct

- Struct là kiểu dữ liệu cấu trúc, cho phép cấu tạo nhiều kiểu dữ liệu vào một kiểu dữ liệu đại diện duy nhất.

- Cú pháp khai báo:

```
typedef struct {  
    <Kiểu dữ liệu 1>;
```

```
    ...
```

```
    <Kiểu dữ liệu n>
```

```
} <Tên kiểu dữ liệu>;
```

- Ví dụ: ta có thể định nghĩa kiểu dữ liệu Phim với các trường dữ liệu sau

```
typedef struct {  
    char* tenPhim;  
    char* daoDien;  
    int thoiLuong;  
} Phim;
```



2.3 Struct

- Sử dụng kiểu dữ liệu Struct đã khai báo.

- Ví dụ khai báo: khai báo kiểu dữ liệu Phim đã định nghĩa:

```
Phim phim = { "Hobbit", "Peter", 161};
```

Lưu ý: các trường dữ liệu phải truyền đúng thứ tự khai báo và kiểu dữ liệu tương ứng. Hoặc có thể dùng cách khai báo tường minh hơn:

```
Phim phim;
```

```
phim.tenPhim = "Hobbit";
```

```
phim.daoDien = "Peter";
```

```
phim.thoiLuong = 161;
```

- Ví dụ truy xuất thông tin: truy xuất thông qua kiểu dữ liệu Phim đã khai báo.

```
printf("Tên phim: %s; Đạo diễn: %s; Thời lượng: %d phút",  
    phim.tenPhim, phim.daoDien, phim.thoiLuong);
```





2.4 Enum

- Enum được sử dụng như một kiểu dữ liệu liệt kê bao gồm nhiều biến có kiểu dữ liệu số nguyên, mỗi biến được đánh số từ 0 đến n.

- Cú pháp khai báo:

```
typedef enum {  
    <Biến 0>,  
    ...
```

```
    <Biến n>  
} <Tên kiểu enum>;
```

- Ví dụ khai báo: khai báo Enum chứa thông tin về định dạng phim.

```
typedef enum {  
    TwoD,  
    TwoDDigital,  
    ThreeD  
} DinhDangPhim;
```



2.4 Enum

- Truy xuất giá trị Enum đã khai báo.

- Ví dụ truy xuất: thực hiện dùng cấu trúc điều khiển SWITCH...CASE để thiết lập giá vé theo định dạng phim đã khai báo trong Enum DinhDangPhim.

```
int giaVe = 0;  
switch (dinhDangPhim) {  
    case TwoD:  
        giaVe = 135000;  
        break;  
    case TwoDDigital:  
        giaVe = 145000;  
        break;  
    case ThreeD:  
        giaVe = 155000;  
        break;  
}
```



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 5. *Hướng đối tượng trong Objective-C*

Ngành Mạng và Thiết bị di động



2014

5014



Nội dung



1. Lớp và đối tượng
 - Khái niệm lớp trong Objective-C
 - Thành phần cấu tạo lớp
 - Đối tượng và sử dụng lớp
2. Thuộc tính
3. Phương thức



1.1 Khái niệm lớp trong Objective-C



- Objective-C định nghĩa lớp bao gồm những thành phần khai báo quan trọng trong tập tin .h, sau đó thực thi các xử lý trong tập tin .m.



```
@interface Phim : NSObject {
    // Khai báo các trường dữ liệu lớp.
}

/*
Khai báo các thuộc tính và phương thức xử lý.
*/

@end
```

```
#import "Phim.h"

@implementation Phim {
    // Khai báo các biến thực thể
}

/*
Thực thi các phương thức xử lý đã định nghĩa
*/

@end
```





1.2 Thành phần cấu tạo lớp

❑ Lớp là khái niệm dùng để mô tả một thực thể ở thế giới thực thành đối tượng ở trong ứng dụng. Lớp bao gồm ba thành phần:

- Biến thực thể:
 - Bao gồm các biến dữ liệu được cấp phát, truy xuất sử dụng bên trong lớp.
 - Ví dụ: lớp Phim được khai báo các biến để lưu trữ tên phim, tên đạo diễn, thời lượng...
- Phương thức
 - Mô tả cách thức đối tượng hoạt động
 - Ví dụ: thao tác trên đối tượng phim: cập nhật thông tin phim, xem giới thiệu phim...
- Thuộc tính
 - Mô tả tính chất của một đối tượng.
 - Ví dụ: đối tượng Phim có thể có những thuộc tính: tên phim, tên đạo diễn, hãng sản xuất...



1.3 Đối tượng và sử dụng lớp

❑ Biến thực thể được sử dụng để cấu tạo một đối tượng thuộc lớp, đối tượng có thuộc tính và cách thức hoạt động đã khai báo trong lớp. Hoặc có thể hiểu đối tượng có kiểu dữ liệu là lớp đã khai báo.

- Cú pháp khởi tạo:
`<Tên Lớp> *<Tên đối tượng> = [[<Tên Lớp> alloc] init];`
- Ví dụ:
`Phim *phim = [[Phim alloc] init];`
- Ý nghĩa:
 - Đối tượng phim được cấp phát bộ nhớ lưu trữ bằng phương thức **alloc**, sau đó khởi tạo giá trị mặc định thông qua phương thức **init**.





1.3 Đối tượng và sử dụng lớp

- **Đối tượng có thể truy xuất các biến thực thể của lớp, tuy nhiên tùy thuộc vào khai báo khả năng truy xuất của từng biến. Khả năng truy xuất thuộc tính bao gồm:**
 - Private
 - Chỉ có đối tượng thuộc lớp hoặc các đối tượng thuộc lớp kế thừa mới có thể truy xuất vào thuộc tính của lớp.
 - Protected
 - Từ khóa mặc định cho các thuộc tính không khai báo từ khóa, chỉ có đối tượng thuộc lớp và các lớp con mới có thể truy xuất vào thuộc tính lớp
 - Public
 - Các thuộc tính lớp có thể truy xuất ở bất cứ lớp nào.
 - Package
 - Các thuộc tính lớp có thể truy xuất ở bất cứ lớp nào, tuy nhiên phải thuộc cùng gói ứng dụng hoặc thư viện.



1.3 Đối tượng và sử dụng lớp

- **Ví dụ về khai báo biến và sử dụng**





```
@interface Phim : NSObject {
    // Khai báo các trường dữ liệu lớp.
    @protected
    char* tenPhim;
    char* daoDien;
    int thoiLuong;
}

/*
    Khai báo các thuộc tính và phương thức xử lý.
*/
- (void) khoiTaoPhim;

@end
```

```
- (void) khoiTaoPhim {
    tenPhim = "Hobbit";
    daoDien = "Peter Jackson";
    thoiLuong = 161;

    printf("Tên Phim: %s \nĐạo diễn: %s \nThời lượng: %d phút ",
        tenPhim, daoDien, thoiLuong);
}
```



Nội dung



1. Lớp và đối tượng
2. Thuộc tính
 - Khai báo thuộc tính
 - Tập từ khóa thuộc tính
3. Phương thức



2.1 Khai báo thuộc tính



- Thuộc tính được sử dụng để có thể truy xuất dữ liệu thông qua đối tượng khởi tạo, tránh truy xuất trực tiếp các biến thực thể.
 - Cú pháp khai báo:
`@property <Tập thuộc tính> <Kiểu dữ liệu> <Tên thuộc tính>`
 - Ví dụ:
`@property (readwrite) char* tenPhim;`
 - Ý nghĩa:
Khai báo thuộc tính **tenPhim** có dạng (readwrite: mặc định có thể truy xuất đọc ghi) có kiểu dữ liệu chuỗi kí tự.





2.2 Tập từ khóa thuộc tính

□ Thuộc tính khi được khai báo sẽ có nhiều dạng định nghĩa để có thể quản lý bộ nhớ cũng như quản lý truy xuất. Bao gồm các dạng sau:

- Atomicity:

- nonatomic: chỉ định thuộc tính có thể trả về các giá trị khác nhau ở thời điểm khác nhau trong các tiến trình khác nhau.

- Setter Semantic:

- retain: giá trị thuộc tính nhận vào được lưu lại, giá trị lưu trữ trước đó được giải phóng khỏi vùng nhớ.
 - copy: sao chép giá trị thuộc tính mới nhận vào, giá trị lưu trữ trước đó được giải phóng khỏi vùng nhớ.
 - assign: thiết lập giá trị cho thuộc tính không sử dụng retain và copy.
 - strong: cùng ý nghĩa với retain nhưng sử dụng trong hệ thống ARC.
 - weak: cùng ý nghĩa với assign nhưng sử dụng trong hệ thống ARC.



2.2 Tập từ khóa thuộc tính

- Read/write:

- readwrite: chỉ định thuộc tính có thể đọc và ghi dữ liệu, thuộc tính cần có phương thức đóng gói (getter & setter).
 - read-only: chỉ định thuộc tính chỉ có thể đọc, thuộc tính cần có phương thức đóng gói (getter).

- Methods Name:

- getter=<getterName>: thay đổi tên phương thức getter thành <getterName>.
 - setter=<setterName>: thay đổi tên phương thức setter thành <setterName>.



Nội dung



1. Lớp và đối tượng
2. Thuộc tính
3. Phương thức
 - Định nghĩa và khai báo phương thức
 - Thực thi gọi phương thức



3.1 Định nghĩa và khai báo phương thức



- Phương thức được dùng để mô tả các thao tác của đối tượng, hoặc một thể hiện của lớp trong quá trình thực thi.
 - Cú pháp khai báo
`<Kiểu phương thức> <Kiểu dữ liệu trả về> <Tên phương thức>;
(<Kiểu dữ liệu 1>) <Tên tham số 1>
...
<Tên diễn giải tham số n>: (<Kiểu dữ liệu n>) <Tên tham số n>;`
 - Ví dụ
 - (Phim) khoiTaoPhim: (char*) tenPhim sanXuat: (char*) nhaSanXuat;





3.2 Thực thi phương thức

□ Phương thức bao gồm 2 dạng:

- Phương thức truy xuất thông qua đối tượng được kí hiệu “-” ở đầu phương thức.

Ví dụ: - (Phim) khaiTaoPhim: (char*) tenPhim sanXuat: (char*) nhaSanXuat;

- Phương thức truy xuất thông qua lớp được kí hiệu “+” ở đầu phương thức.

Ví dụ: + (Phim) khaiTaoPhim: (char*) tenPhim sanXuat: (char*) nhaSanXuat;



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 6. Các điều khiển cơ bản

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Điều khiển hiển thị nội dung văn bản

- Label
- TextField
- TextView

2. Điều khiển tương tác



1.1 Label



❑ **UILabel:** đối tượng hiển thị văn bản tĩnh có thuộc tính chỉ đọc.

- Mục đích sử dụng chính
 - Diễn giải chức năng của các điều khiển khác trong ứng dụng
 - Diễn giải ngữ cảnh hoặc chỉ dẫn trong ứng dụng
- Ví dụ

Thông Tin Phim



Hobbit 2 - The Desolation of Smaug

Điểm đánh giá: 8.1



Label



1.1 Label



□ Tùy chỉnh hiển thị đối tượng UILabel trong Interface Builder



Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

4

1.1 Label



□ Tùy chỉnh hiển thị đối tượng UILabel thông qua thuộc tính:

- text
- attributedText
- numberOfLine
- font
- textColor
- textAlignment



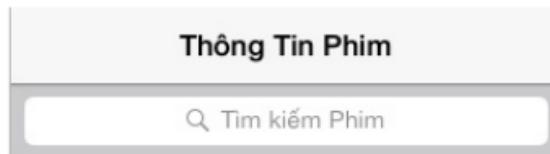
Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

5



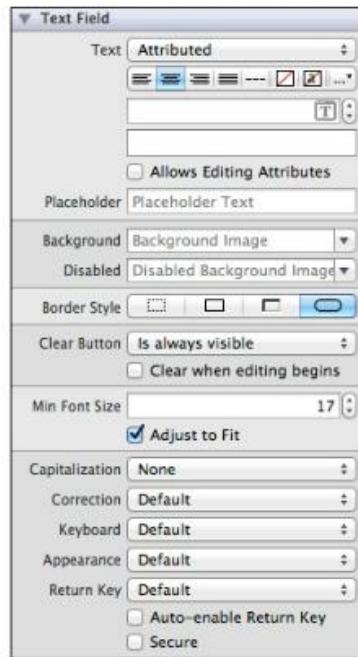
1.2 TextField

- ❑ **UITextField:** đối tượng cho phép nhập một dòng dữ liệu văn bản từ người dùng.
 - Mục đích sử dụng chính
 - Nhận dữ liệu nhập và xử lý khi có yêu cầu.
 - Ví dụ



1.2 TextField

- ❑ Tùy chỉnh hiển thị đối tượng UITextField trong Interface Builder.

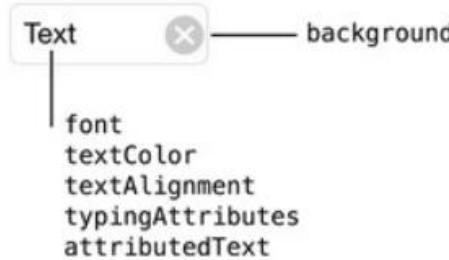


1.2 TextField



❑ Tùy chỉnh hiển thị đối tượng UITextField thông qua thuộc tính:

- text
- attributedText
- placeHolder
- attributePlaceHolder
- defaultTextAttributes
- font
- textColor
- textAlignment
- typingAttributes



1.3 TextView



❑ UITextView: đối tượng cho phép nhập và hiển thị dữ liệu văn bản nhiều dòng, hỗ trợ thao tác cuộn.

- Mục đích sử dụng chính

- Hiển thị và cho phép tùy chỉnh dữ liệu nhập và xử lý khi có yêu cầu.

- Ví dụ

		Hủy	Thêm Phim	Xong
Tên Phim	<input type="text" value="Hobbit 2"/>			
Thời lượng	161	phút		
Đạo diễn	<input type="text" value="Peter Jackson"/>			
Ngày chiếu	<input type="text" value="01.01.2014"/>			
Đánh giá	<input type="text" value="8.1"/>			
Giá vé	<input type="text" value="135000"/>			
Nội dung	<input type="checkbox"/> Phần 2 tiếp tục kể về cuộc phiêu lưu của nhân vật chính Bilbo Baggins và cuộc hành trình với phù thủy Gandalf cùng mười ba người lùn dẫn đầu bởi Thorin Oakenshield, họ mang trên mình một nhiệm vụ sứ thi để đòi lại Lonely Mountain và Vương quốc người lùn đã mất của Erebor.			

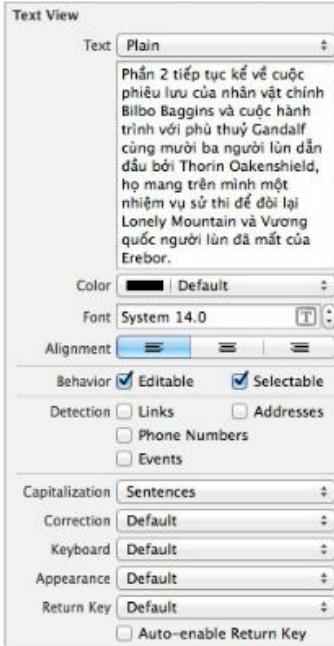
TextView



1.3 TextView



- **Tùy chỉnh hiển thị đối tượng UITextView trong Interface Builder.**

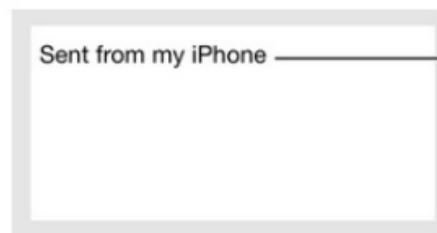


1.3 TextView



- **Tùy chỉnh hiển thị đối tượng UITextView thông qua thuộc tính:**

- text
- attributedText
- hasText
- font
- textColor
- editable
- allowsEditingTextAttributes
- dataDetectorTypes
- textAlignment
- typingAttributes
- linkTextAttributes
- textContainerInset



font
textColor
textAlignment
typingAttributes
attributedText



Nội dung



1. Điều khiển hiển thị nội dung văn bản

2. Điều khiển tương tác

- Button
- Switch
- Segment
- Slider
- Stepper



2.1 Button



□ **UIButton:** đối tượng nhận các tương tác chạm, nhấn trên màn hình thiết bị thuộc khu vực hiển thị và thực thi hành động đã thiết lập sẵn.

- Mục đích sử dụng chính
 - Thực thi các xử lý được thiết lập dựa trên tương tác người dùng.
- Ví dụ

Button + ⓘ

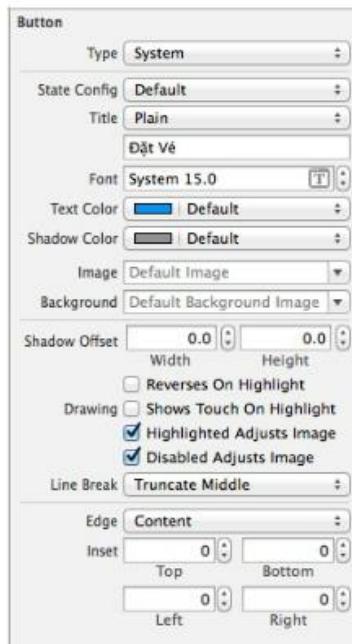
Đặt vé Đánh giá



2.1 Button



- ❑ **Tùy chỉnh hiển thị đối tượng UIButton trong Interface Builder.**



2.1 Button



- ❑ **Khởi tạo đối tượng UIButton thông qua phương thức:**
 - buttonWithType:
- ❑ **Truy xuất các thuộc tính liên quan đến tiêu đề hiển thị UIButton:**
 - titleLabel
 - titleForState:
 - setTitle:
 - attributedTitleForState:
 - setAttributedTitle: forState:
 - titleColorForState:
 - setTitleColor:
 - titleShadowColorForState:
 - setTitleColorShadowColor: forState
 - reversesTitleShadowWhenHighlighted





2.2 Switch

- ❑ **UISwitch:** đối tượng cho phép người dùng thực hiện chuyển đổi giữa 2 trạng thái.

- Mục đích sử dụng chính

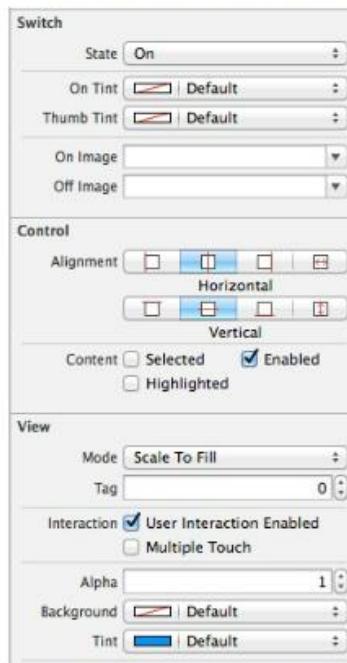
- Thực thi chuyển đổi giữa hai trạng thái và xử lý theo từng trạng thái riêng biệt.
- Bật hoặc tắt một tính năng nào đó.

- Ví dụ



2.2 Switch

- ❑ Tùy chỉnh hiển thị đối tượng UISwitch trong Interface Builder.





2.2 Switch

❑ Khởi tạo đối tượng UISwitch:

- initWithFrame:

❑ Truy xuất thuộc tính bật/tắt:

- on
- setOn:animated:

❑ Truy xuất các thuộc tính hiển thị n:

- onTintColor
- tintColor
- thumbTintColor
- onImage
- offImage



2.3 Segment

❑ UISegmentedControl: dạng điều khiển tổ hợp các nút theo chiều dọc, với mỗi nút được thực thi xử lý riêng biệt, tại một thời điểm chỉ một nút được thực thi.

- Mục đích sử dụng chính
 - Tổ hợp thực thi các nút điều khiển và xử lý theo từng nút riêng biệt khi có tương tác.
- Ví dụ



2.3 Segment



- **Tùy chỉnh hiển thị đối tượng UISegmentedControl trong Interface Builder.**



Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

20

2.3 Segment



- **Khởi tạo đối tượng UISegmentedControl:**

- initWithFrame:

- **Thiết lập nội dung hiển thị:**

- setImage: forSegmentAtIndex:
- imageForSegmentAtIndex:
- setTitle: forSegmentAtIndex:
- titleForSegmentAtIndex:

- **Quản lý thông tin Segment:**

- insertSegmentWithImage: atIndex: animated:
- insertSegmentWithTitle: atIndex: animated:
- numberOfSegments
- removeAllSegments
- removeSegmentAtIndex: animated:
- selectedSegmentIndex



Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

21



2.4 Slider

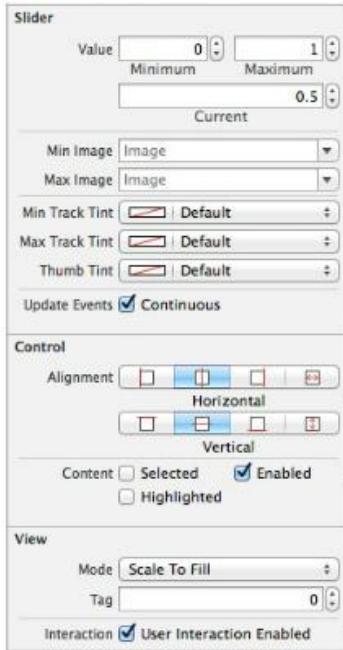
- ❑ UISlider: dạng điều khiển cho phép người dùng tùy chỉnh thông số dựa trên tương tác kéo trượt.

- Mục đích sử dụng chính
 - Thực hiện tăng giảm giá trị một cách tự động hoặc dựa trên tương tác.
 - Thiết lập giá trị cho điều khiển khác.
 - Thiết lập giá trị nhanh chóng bằng tương tác kéo trượt đơn giản.
- Ví dụ



2.4 Slider

- ❑ Tùy chỉnh hiển thị đối tượng UISlider trong Interface Builder.





2.4 Slider

❑ Khởi tạo giá trị cho đối tượng

UISlider:

- value
- setValue: animated:

❑ Truy xuất giá trị min/max:

- minimumValue:
- maximumValue:

❑ Thay đổi giá trị UISlider:

- minValueImage
- maxValueImage
- minimumTrackTintColor
- currentMinimumTrackImage
- minimumTrackImageForState:
- setMinimumTrackImage: forState:
- thumbTintColor
- currentThumbImage
- thumbImageForState:
- setThumbImage: forState:



2.5 Stepper

❑ UIStepper: dạng điều khiển cho phép tăng giảm với lượng giá trị nhất định.

- Mục đích sử dụng chính
 - Tăng giảm giá trị có thông số rời rạc.
 - Thiết lập giá trị cho điều khiển khác.
- Ví dụ

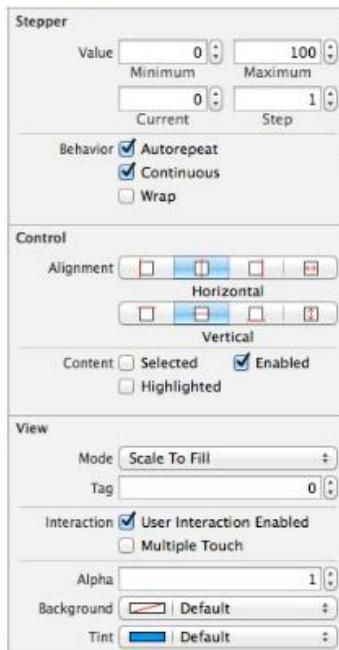
Số lượng vé: 4



2.5 Stepper



- **Tùy chỉnh hiển thị đối tượng UIStepper trong Interface Builder.**



Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

26

2.5 Stepper



- **Thiết lập thông số cho đối tượng UIStepper**

- continuous
- Autorepeat
- Wraps
- minimumValue
- maximumValue
- stepValue

- **Truy xuất giá trị:**

- value



Lập trình iOS (2014) - Bài 6. Các điều khiển cơ bản

27

2.5 Stepper



❑ Thiết lập hiển thị:

- tintColor
- backgroundImageForState:
- setBackgroundImage: forState:
- decrementImageForState:
- setDecrementImage: forState:
- dividerImageForLeftSegmentState: rightSegmentState:
- setDividerImage: forState: rightSegmentState:
- incrementImageForState:
- setIncrementImage: forState:



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 7. Lưu trữ dữ liệu dạng tập hợp

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



1. Khái niệm dữ liệu tập hợp
 - Chức năng & Phân nhóm
 - Dữ liệu tập hợp biến đổi và bất biến đổi.
2. NSArray



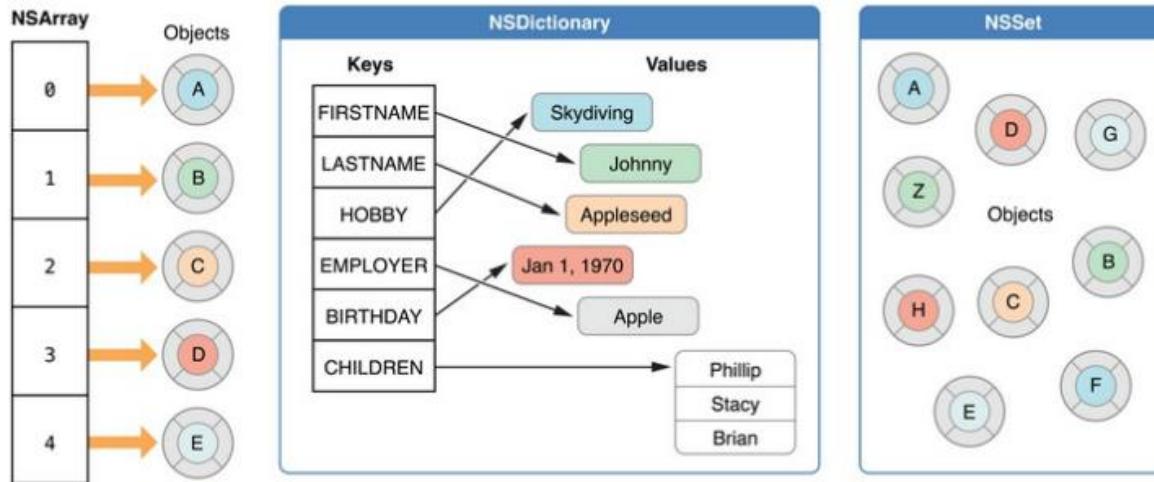
1.1 Chức năng và Phân nhóm



- **Chức năng:**
 - CocoaTouch cung cấp các lớp nền tảng cho phép lập trình viên sử dụng trong việc lưu trữ một nhóm các đối tượng.
 - Việc lưu trữ dạng tập hợp giúp dễ dàng lưu trữ và các thao tác truy xuất.
- **Phân nhóm: dữ liệu tập hợp được phân vào 3 nhóm chính**
 - NSArray
 - NSSet
 - NSDictionary



1.1 Chức năng và Phân nhóm



1.2 Dữ liệu tập hợp biến đổi và bất biến đổi



- Các loại dữ liệu tập hợp đều tổ chức quản lý các đối tượng ở hai dạng:
 - Bất biến đổi (Immutable)
 - Biến đổi (Mutable)

- Các chức năng quản lý của dữ liệu tập hợp đối với đối tượng:
 - Bất biến đổi :
 - Liệt kê dữ liệu đối tượng trong tập hợp
 - Định nghĩa đối tượng thuộc tập hợp
 - Truy xuất các phần tử trong tập hợp
 - Biến đổi:
 - Bao gồm các chức năng của dữ liệu tập hợp bất biến
 - Thêm đối tượng vào tập hợp
 - Xóa đối tượng khỏi tập hợp



Nội dung



1. Khái niệm dữ liệu tập hợp

2. NSArray

- Khởi tạo mảng
- Liệt kê phần tử mảng
- So sánh mảng
- Kiểm tra phần tử mảng
- Sắp xếp mảng



2. NSArray



❑ NSArray cho phép quản lý các đối tượng

- Bắt biến đổi (NSArray)
- Biến đổi (NSMutableArray)

❑ Các chức năng quản lý của dữ liệu tập hợp đối với đối tượng:

- Bắt biến đổi :
 - Liệt kê dữ liệu đối tượng trong tập hợp
 - Định nghĩa đối tượng thuộc tập hợp
 - Truy xuất các phần tử trong tập hợp
- Biến đổi:
 - Bao gồm các chức năng của dữ liệu tập hợp bắt biến
 - Thêm đối tượng vào tập hợp
 - Xóa đối tượng khỏi tập hợp





2.1 Khởi tạo mảng

❑ Khởi tạo mảng

- Cú pháp: `@[]`
- Hoặc sử dụng phương thức `arrayWithObjects`
- Ví dụ:

```
NSArray *tenPhim = @[@"Hobbit", @"Avengers", @"Frozen"];  
NSArray *tenPhim =  
[NSArray arrayWithObjects:@@"Hobbit", @@"Avengers", @@"Frozen", nil];
```



2.2 Liệt kê phần tử mảng

❑ Liệt kê phần tử mảng

- Truy xuất từng phần tử theo chỉ số trong mảng hoặc có thể sử dụng các vòng lặp để truy xuất tuần tự các phần tử trong mảng
- Ví dụ:

```
NSArray *tenPhim = @[@"Hobbit", @"Avengers", @"Frozen"];  
for (NSString *item in tenPhim) {  
    NSLog(@"%@", item);  
}  
Hoặc  
for (int i=0; i<[tenPhim count]; i++) {  
    NSLog(@"%@", i, germanMakes[i]);  
}
```



2.3 So sánh mảng



□ So sánh mảng

- Sử dụng phương thức **isEqualToString** để thực hiện kiểm tra số lượng và giá trị phần tử mảng.
- Kết quả trả về:
 - YES: hai mảng bằng nhau
 - NO: hai mảng không bằng nhau
- Ví dụ:

```
NSArray *phimDangChieu = @[@"Hobbit", @"Avengers", @"Frozen"];
NSArray *phimSapChieu = @[, @"47 Ronin", @"In the door"];
```

```
if ([phimDangChieu isEqualToString:phimSapChieu]) {
    NSLog(@"Hai mảng bằng nhau");
}
```



2.4 Kiểm tra phần tử mảng



□ Kiểm tra phần tử mảng

- Sử dụng phương thức **containsObject** để thực hiện kiểm tra phần tử tồn tại trong mảng, phương thức **indexForObject** cho biết vị trí đối tượng trong mảng.
- Ví dụ:

```
NSArray *phimDangChieu = @[@"Hobbit", @"Avengers", @"Frozen"];
```

```
if ([phimDangChieu containsObject:@"Hobbit"]) {
    NSLog(@"Hobbit – Đạo diễn: Peter Jackson");
}
NSUInteger index = [phimDangChieu indexForObject:@"Hobbit"];
if (index == NSNotFound) {
    NSLog(@"Không tìm thấy");
} else {
    NSLog(@"Hobbit – Đạo diễn: Peter Jackson – Vị trí phim %d", index);
}
```



2.5 Sắp xếp mảng



□ Sắp xếp mảng

- Sử dụng phương thức **sortedArrayUsingComparator** kết hợp sử dụng kĩ thuật block **^NSComparisonResult(id obj1, id obj2)** để tiến hành sắp xếp mảng.
- Kết quả khi sắp xếp:
 - **NSOrderedAscending**: obj1 trước obj2
 - **NSOrderedSame**: obj1 và obj2 không có quan hệ thứ tự
 - **NSOrderedDescending**: obj1 sau obj2



2.5 Sắp xếp mảng



- Ví dụ:

```
NSArray *danhSachPhim = @[@"Hobbit", @"Avengers", @"Frozen"];
```

```
NSArray *dsSapXep = [danhSachPhim sortedArrayUsingComparator:
```

```
    ^NSComparisonResult(id obj1, id obj2) {  
        if ([obj1 length] < [obj2 length]) {  
            return NSOrderedAscending;  
        } else if ([obj1 length] > [obj2 length]) {  
            return NSOrderedDescending;  
        } else {  
            return NSOrderedSame;  
        }  
    };  
    NSLog(@"%@", dsSapXep );
```



Thảo luận





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

[Go Screen Capture](#)

Lập trình iOS

Bài 8. *Đọc ghi tập tin*

Ngành Mạng và Thiết bị di động



2014

2014



Nội dung



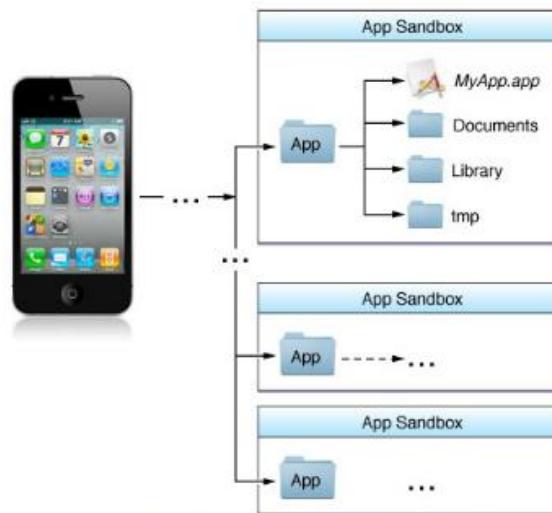
1. Giới thiệu bộ lưu trữ tập tin trong iOS
2. Thao tác với tập tin



1. Giới thiệu bộ lưu trữ tập tin trong iOS



- ❑ Tập tin lưu trữ trong iOS được quản lý chặt chẽ bởi hệ thống. Để đơn giản hóa, ứng dụng chỉ được phép truy xuất và tạo thư mục trong khu vực cấp phát ứng dụng đó (trường hợp ngoại lệ ứng dụng xây dựng Interface truy xuất danh bạ hoặc hình ảnh).





1. Giới thiệu bộ lưu trữ tập tin trong iOS

- ❑ **Ứng dụng iOS được cấp phát một thư mục <Application_Home> bao gồm các thư mục bên trong cho phép ghi các tập tin theo từng mục đích khác nhau. Bao gồm:**
 - **AppName.app** : thư mục bundle, chứa nội dung của ứng dụng đã được đánh dấu không được ghi khi tiến hành cài ứng dụng.
 - **Documents**: thư mục chứa các tài liệu người dùng và các tập tin ứng dụng quan trọng. Các tài liệu và tập tin trong thư mục này chỉ có thể tạo một lần và được **back-up** bởi **iTunes**.
 - **Documents/Inbox**: sử dụng thư mục cho phép truy xuất các tập tin được yêu cầu mở bởi các ứng dụng bên ngoài. Tập tin trong thư mục chỉ cho phép đọc hoặc xoá.
 - Ví dụ: ứng dụng Mail sẽ lưu trữ các tập tin đính kèm có liên quan đến ứng dụng trong thư mục Inbox của ứng dụng đó.



1. Giới thiệu bộ lưu trữ tập tin trong iOS

- **Library**: thư mục ở dạng cấp cao cho phép lưu trữ các tập tin đặc biệt (các tập tin không muốn cho người dùng truy xuất và sử dụng). Có thể tạo thư mục con trong thư mục này.
- **tmp**: sử dụng thư mục cho các mục đích tạo các tập tin tạm, khi ứng dụng không hoạt động, các tập tin trong thư mục này sẽ tự động được xoá bởi hệ thống.





1. Giới thiệu bộ lưu trữ tập tin trong iOS

❑ Nên lưu tập tin ở đâu?

- Việc lưu trữ tập tin ảnh hưởng trực tiếp đến quá trình backup thiết bị hoặc đồng bộ hóa với iCloud. Do đó khi lưu trữ nên tuân thủ một số nguyên tắc sau:
 - Các tập tin liên quan đến người dùng nên để vào thư mục Documents.
 - Các tập tin hỗ trợ, ứng dụng tải về, tự động tạo hoặc có thể tái tạo thì có 2 kiểu lưu trữ:
 - iOS 5.0 và phiên bản cũ hơn: chứa trong thư mục Library/Caches cho phép ngăn việc backup không cần thiết.
 - iOS 5.0.1 trở lên: sử dụng thư mục Library/Application Support, nên gán tên ứng dụng vào phần mở rộng tập tin để dễ quản lý và chống backup.
 - Ví dụ: film-data-extended.txt.com.t3h.Datvexemphim
 - Các tập tin Cache nên lưu trữ trong Library/Cache, ứng dụng nên kiểm soát các trường hợp hệ thống xoá những tập tin này.



1. Giới thiệu bộ lưu trữ tập tin trong iOS

❑ Nên lưu tập tin ở đâu?

- Thư mục tmp lưu trữ các tập tin phát sinh và nên xoá các tập tin này khi không còn cần thiết nữa, tránh tình trạng chiếm dụng bộ nhớ thiết bị.





1. Giới thiệu bộ lưu trữ tập tin trong iOS

□ Chọn cách để truy xuất tập tin

- Tất cả tập tin trong iOS có thể thực hiện truy xuất để đọc ghi dữ liệu theo kiểu **stream of bytes**, tuy nhiên để dễ dàng hơn trong việc truy xuất iOS cung cấp các phương thức phù hợp với từ loại nội dung (văn bản, hình ảnh, âm thanh, danh sách thuộc tính...) bao gồm:
 - Tập tin tài nguyên: nib, hình ảnh, âm thanh, chuỗi...
 - Tập tin văn bản: văn bản thường, văn định định dạng UTF-8, UTF-16
 - Tập tin có cấu trúc: XML, Property List, Preference
 - Tập tin đóng gói: các tập tin sử dụng đối tượng đóng gói mã hoá
 - Gói tập tin: bao gồm các tập tin có định dạng khác nhau
 - Bundle: lưu trữ các mã nguồn và các tập tin liên quan đến mã nguồn
 - Tập tin mã nguồn: chứa các tập tin plug-in hoặc mã nguồn thư viện
 - Tập tin Wrapper: chứa tất cả các tập tin khác trong một tập tin khác



Nội dung

1. Giới thiệu bộ lưu trữ tập tin trong iOS

2. Thao tác với tập tin văn bản

- Đối tượng NSFileManager
- Các thao tác trên tập tin văn bản



2.1. Đối tượng NSFileManager



❑ NSFileManager:

- Đối tượng NSFileManager cho phép tổ chức và thực thi thao tác tập tin hệ thống và cách ly ứng dụng khỏi các tập tin hệ thống.
- Có thể thực thi tạo đối tượng tham chiếu đến quản lý tập tin hoặc tạo đối tượng trực tiếp NSFileManager.
- Hỗ trợ lớp NSURL và NSString cho phép chỉ định đường dẫn đến thư mục hoặc tập tin cần truy xuất.
- Cho phép thực hiện tạo các hàm uỷ thác cho phép quản lý các hoạt động như di chuyển, sao chép, liên kết hoặc xoá bỏ các tập tin hoặc thư mục.
- Từ iOS 5 trở lên, bao hàm các phương thức cho phép thực hiện tương tác với các danh mục được lưu trữ trên iCloud.



2.1. Đối tượng NSFileManager



❑ Khởi tạo đối tượng:

- init
- defaultManager

Ví dụ:

```
NSFileManager *quanLyTapTin = [NSFileManager defaultManager];  
// hoặc  
NSFileManager *quanlyTapTin = [[[NSFileManager alloc] init];
```





2.1. Đối tượng NSFileManager

❑ Truy xuất nội dung thư mục

- [contentsOfDirectoryAtURL:includingPropertiesForKeys:options:error:](#)
- [contentsOfDirectoryAtPath:error:](#)
- [enumeratorAtURL:includingPropertiesForKeys:options:errorHandler:](#)
- [enumeratorAtPath:](#)
- [mountedVolumeURLsIncludingResourceValuesForKeys:options:](#)
- [subpathsOfDirectoryAtPath:error:](#)
- [subpathsAtPath:](#)



2.1. Đối tượng NSFileManager

❑ Khởi tạo và xoá thư mục/ tập tin

- [createDirectoryAtURL:withIntermediateDirectories:attributes:error:](#)
- [createDirectoryAtPath:withIntermediateDirectories:attributes:error:](#)
- [createFileAtPath:contents:attributes:](#)
- [removeItemAtURL:error:](#)
- [removeItemAtPath:error:](#)
- [replaceItemAtURL:withItemAtURL:backupItemName:options:resultingItemURL:error:](#)





2.1. Đối tượng NSFileManager

❑ Di chuyển và xoá thư mục/ tập tin

- [copyItemAtURL:toURL:error:](#)
- [copyItemAtPath:toPath:error:](#)
- [moveItemAtURL:toURL:error:](#)
- [moveItemAtPath:toPath:error:](#)



2.1. Đối tượng NSFileManager

❑ Kiểm tra quyền truy xuất tập tin

- [fileExistsAtPath:](#)
- [fileExistsAtPath:isDirectory:](#)
- [isReadableFileAtPath:](#)
- [isWritableFileAtPath:](#)
- [isExecutableFileAtPath:](#)
- [isDeletableFileAtPath:](#)





2.1. Đối tượng NSFileManager

- ❑ Truy xuất và so sánh nội dung tập tin
 - [contentsAtPath:](#)
 - [contentsEqualAtPath:andPath:](#)
- ❑ Chuyển đổi đường dẫn tập tin thành chuỗi
 - [fileSystemRepresentationWithPath:](#)
 - [stringWithFileSystemRepresentation:length:](#)
- ❑ Quản lý thư mục đang truy xuất
 - [changeCurrentDirectoryPath:](#)
 - [currentDirectoryPath](#)



2.2. Các thao tác với tập tin văn bản

- ❑ Tạo thư mục lưu trữ tập tin
 - ```
(NSString *)truyXuatThuMuc {
 quanLyTapTin = [NSFileManager defaultManager];

 tenThuMuc = [NSHomeDirectory()
 stringByAppendingPathComponent:@"LuuPhim"];

 return tenThuMuc;
}
```





## 2.2. Các thao tác với tập tin văn bản

### □ Ghi nội dung tập tin văn bản

```
duongDanTapTin = [[NSString alloc] init];

// Biến ghi nhận lỗi nếu có.
NSError *loiGhiTapTin;

duongDanTapTin = [self.truyXuatThuMuc
 stringByAppendingPathComponent:self.tenTapTin];

if (![quanLyTapTin fileExistsAtPath:duongDanTapTin]) {

 [noiDungCanGhi writeToFile:duongDanTapTin
 atomically:YES
 encoding:NSUTF8StringEncoding
 error:&loiGhiTapTin];

 if(loiGhiTapTin) {
 NSLog(@"Có lỗi trong quá trình ghi tập tin %@", loiGhiTapTin);
 }
}
}
```



Cơ sở dữ liệu (2014) - Bài 1. Mô hình dữ liệu quan hệ

18



## 2.2. Các thao tác với tập tin văn bản

### □ Đọc nội dung tập tin văn bản

```
duongDanTapTin = [[NSString alloc] init];

// Biến ghi nhận lỗi nếu có.
NSError *loiDocTapTin;

duongDanTapTin = [self.truyXuatThuMuc
 stringByAppendingPathComponent:self.tenTapTin];

NSString *noiDungTapTin = [[NSString alloc]
 initWithContentsOfFile:duongDanTapTin
 encoding:NSUTF8StringEncoding
 error:&loiDocTapTin];

if (loiDocTapTin) {
 NSLog(@"Có lỗi khi đọc tập tin: %@", loiDocTapTin);
}

return noiDungTapTin;
```



Cơ sở dữ liệu (2014) - Bài 1. Mô hình dữ liệu quan hệ

19



## 2.2. Các thao tác với tập tin văn bản

### □ Kiểm tra sự tồn tại của tập tin văn bản

```
- (BOOL)kiemTraTonTaiTapTin
{
 duongDanTapTin = [[NSString alloc] init];
 duongDanTapTin = [self.truyXuatThuMuc
 stringByAppendingPathComponent:self.tenTapTin];
 if (![quanLyTapTin fileExistsAtPath:duongDanTapTin]) {
 return YES;
 }
 return NO;
}
```



## 2.2. Các thao tác với tập tin văn bản

### □ Xoá tập tin văn bản

```
duongDanTapTin = [[NSString alloc] init];
duongDanTapTin = [self.truyXuatThuMuc
 stringByAppendingPathComponent:self.tenTapTin];

// Biến ghi nhận lỗi nếu có.
NSError *loiXoaTapTin;

BOOL kiemTra = [self.quanLyTapTin
 removeItemAtPath:duongDanTapTin
 error:&loiXoaTapTin];
if (kiemTra) {
 NSLog(@"Da Xoa Tap tin");
} else {
 NSLog(@"Có lỗi trong quá trình xoá tập tin: %@", loiXoaTapTin);
}
```



## Thảo luận





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

[Go Screen Capture](#)

## Lập trình iOS

### Bài 9. *ScrollView & Page Control*

Ngành Mạng & Thiết bị di động



2014

5014



## Nội dung



### 1. ScrollView

- Mục đích sử dụng trong ứng dụng
- Khảo sát lớp UIScrollView
- Xây dựng ScrollView
- Các tác vụ trong ScrollView

### 2. Page Control



### 1.1 Mục đích sử dụng trong ứng dụng

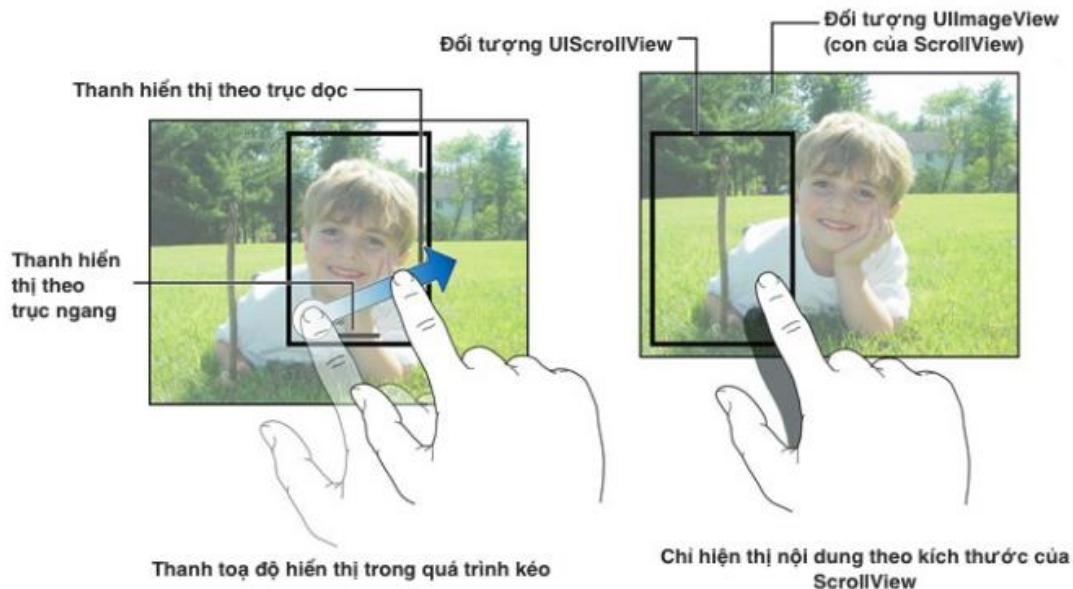


- ❑ **ScrollView được sử dụng nhằm hiển thị các nội dung có kích thước hiển thị lớn hơn kích thước màn hình thiết bị. Bao gồm hai thao tác chính:**
  - Cho phép người dùng sử dụng thao tác kéo thả nội dung muốn hiển thị.
  - Thực hiện phóng to, thu nhỏ nội dung hiển thị.
- ❑ **ScrollView còn cho phép xây dựng hiển thị ở chế độ từng màn hình riêng lẻ (chế độ phân trang).**





## 1.1 Mục đích sử dụng trong ứng dụng



## 1.2 Khảo sát lớp UIScrollView

### ❑ Tuỳ chỉnh hiển thị nội dung

- [setContentOffset:animated:](#)
- [contentOffset](#)
- [contentSize](#)
- [contentInset](#)





## 1.2 Khảo sát lớp UIScrollView

### ❑ Tuỳ chỉnh các thao tác cuộn trong ScrollView

- [directionalLockEnabled property](#)
- [scrollsToTop property](#)
- [scrollRectToVisible:animated:](#)
- [pagingEnabled property](#)
- [bounces property](#)
- [alwaysBounceVertical property](#)
- [alwaysBounceHorizontal property](#)
- [touchesShouldBegin:withEvent:inContentView:](#)
- [touchesShouldCancelInContentView:](#)
- [canCancelContentTouches property](#)
- [delaysContentTouches property](#)



## 1.2 Khảo sát lớp UIScrollView

### ❑ Tuỳ chỉnh các thông số thanh toạ độ trong ScrollView

- [indicatorStyle property](#)
- [scrollIndicatorInsets property](#)
- [showsHorizontalScrollIndicator property](#)
- [showsVerticalScrollIndicator property](#)
- [flashScrollIndicators](#)



## 1.2 Khảo sát lớp UIScrollView



### ❑ Tuỳ chỉnh các thao tác kéo thả trong ScrollView

- [panGestureRecognizer \*property\*](#)
- [pinchGestureRecognizer \*property\*](#)
- [zoomToRect:animated:](#)
- [zoomScale \*property\*](#)
- [setZoomScale:animated:](#)
- [maximumZoomScale \*property\*](#)
- [minimumZoomScale \*property\*](#)
- [zoomBouncing \*property\*](#)
- [zooming \*property\*](#)
- [bouncesZoom \*property\*](#)

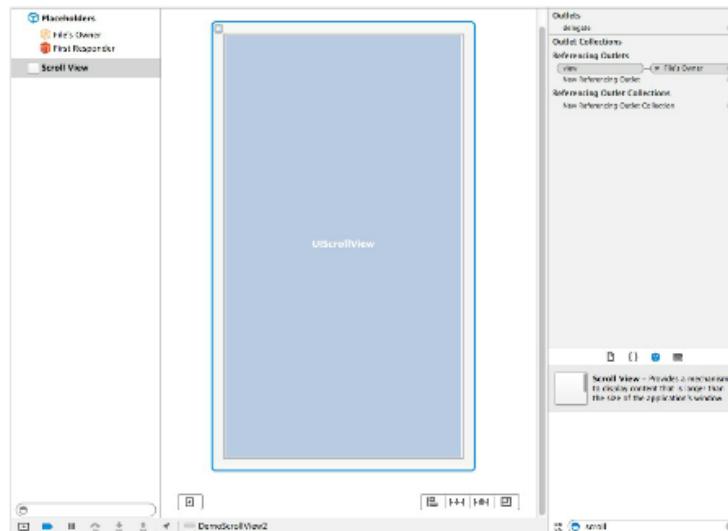


## 1.3 Xây dựng ScrollView



### ❑ Xây dựng ScrollView từ Interface Builder:

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

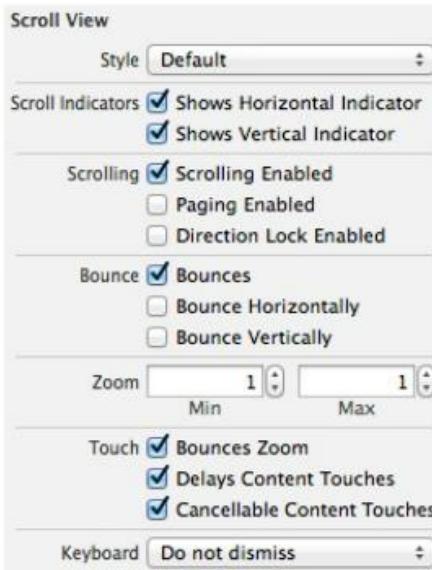


## 1.3 Xây dựng ScrollView



### ❑ Xây dựng ScrollView từ Interface Builder:

- Tuỳ chỉnh các thông số Scrollview trong thẻ Attributes.



## 1.3 Xây dựng ScrollView



### ❑ Xây dựng ScrollView từ bằng mã nguồn:

- Trong hàm loadView của UIViewController thực hiện khởi tạo ScrollView.

```
// Truy xuất thông tin kích thước màn hình ứng dụng
CGRect fullScreenRect = [UIScreen mainScreen] applicationFrame;

// Khởi tạo đối tượng UIScrollView
// trong khu vực kích thước màn hình ứng dụng
scrollView=[[UIScrollView alloc] initWithFrame:fullScreenRect];

// Tuỳ chỉnh lại kích thước
scrollView.contentSize=CGSizeMake(720,1280);

// Thiết lập View của UIViewController là ScrollView
self.view=scrollView;
```





## 1.3 Xây dựng ScrollView

### ❑ Xây dựng ScrollView từ bằng mã nguồn:

- Tuỳ chỉnh các thông số hiển thị ScrollView thông qua các thuộc tính trong lớp UIScrollView.

```
// Thiết lập kích thước
scrollView.contentSize = CGSizeMake(720, 1280);

// Thiết lập màu nền
scrollView.backgroundColor = [UIColor grayColor];

// Tắt hiệu ứng này khi cuộn
scrollView.bounces = NO;

// Thực hiện canh lề
scrollView.contentInset = UIEdgeInsetsMake(64.0, 0.0, 44.0, 0.0);
```



## 1.4 Các tác vụ trong ScrollView

### ❑ UIScrollViewDelegate

- Cho phép thực hiện các phương thức sự kiện được xây dựng để tương tác trực tiếp với lớp ScrollView.
- Tuỳ chỉnh các tương tác trên ScrollView bao gồm: thu phóng nội dung, tăng giảm tốc độ cuộn, các chuyển hoạt...





## 1.4 Các tác vụ trong ScrollView

### ❑ Thao tác cuộn và kéo:

- [scrollViewDidScroll:](#)
- [scrollViewWillBeginDragging:](#)
- [scrollViewWillEndDragging:withVelocity:targetContentOffset:](#)
- [scrollViewDidEndDragging:willDecelerate:](#)
- [scrollViewShouldScrollToTop:](#)
- [scrollViewDidScrollToTop:](#)
- [scrollViewWillBeginDecelerating:](#)
- [scrollViewDidEndDecelerating:](#)



## 1.4 Các tác vụ trong ScrollView

### ❑ Thao tác thu phóng nội dung:

- [viewForZoomingInScrollView:](#)
- [scrollViewWillBeginZooming:withView:](#)
- [scrollViewDidEndZooming:withView:atScale:](#)
- [scrollViewDidZoom:](#)





## 1.4 Các tác vụ trong ScrollView

### ❑ Thiết lập chuyển hoạt khi cuộn:

- [– scrollViewDidEndScrollingAnimation:](#)



## Nội dung

### 1. ScrollView

### 2. Page Control

- Mục đích sử dụng trong ứng dụng
- Khảo sát lớp UIPageControl
- Xây dựng Page Control





## 2.1 Mục đích sử dụng trong ứng dụng

- ❑ PageControl bao gồm một hàng các dấu chấm biểu tượng, với mỗi dấu sẽ đại diện cho một trang hoặc một màn hình trong ứng dụng có nhiều màn hình.



## 2.1 Mục đích sử dụng trong ứng dụng

- ❑ PageControl không bao gồm các nội dung hiển thị trong trang, đồng thời không quản lý các nội dung bên trong trang.
- ❑ Hiển thị cho biết trang hiện tại.
- ❑ Cho phép thực hiện chuyển qua các trang khác nhau thông qua tương tác người dùng.
- ❑ Kết hợp với các điều khiển khác như ScrollView – hiển thị chế độ phân trang.





## 2.2 Khảo sát lớp UIPageControl

- ❑ Lớp UIPageControl bao gồm các thuộc tính cho phép thiết lập các thuộc tính và hành động cho điều khiển Page Control.
- ❑ Quản lý điều khiển trang.
  - [currentPage property](#)
  - [numberOfPages property](#)
  - [hidesForSinglePage property](#)



## 2.2 Khảo sát lớp UIPageControl

- ❑ Cập nhật nội dung hiển thị trang
  - [pageIndicatorTintColor property](#)
  - [currentPageIndicatorTintColor property](#)
  - [defersCurrentPageDisplay property](#)
  - [\\_updateCurrentPageDisplay](#)
- ❑ Điều chỉnh số lượng trang
  - [\\_sizeForNumberOfPages:](#)



## 1.3 Xây dựng ScrollView



### ❑ Xây dựng Page Control từ Interface Builder:

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



Lập trình iOS (2014) – Bài 9. ScrollView & Page Control

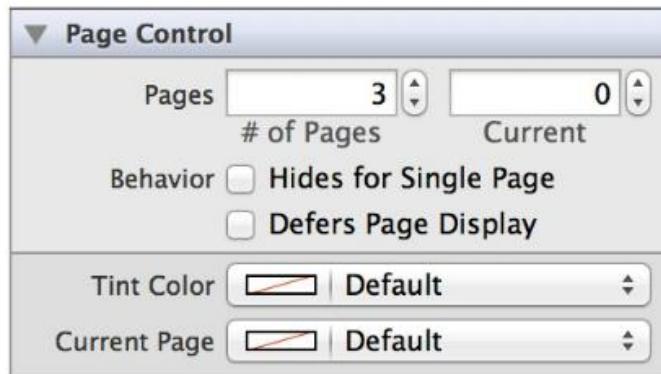
22

## 1.3 Xây dựng ScrollView



### ❑ Xây dựng Page Control từ Interface Builder:

- Tùy chỉnh các thuộc tính trong thẻ Attributes



Lập trình iOS (2014) – Bài 9. ScrollView & Page Control

23



## 2.2 Xây dựng Page Control

- Có thể bắt lại sự kiện chuyển trang thông qua hành động **valueChange**.

- Ví dụ:

```
- (IBAction)changePage:(id)sender {
 NSInteger num = [self.pageControl currentPage];
 NSString* numString = [[NSString
alloc] initWithFormat:@"%d", ++num];
 [self.numberOfPage setText:numString];
 [self.pageControl updateCurrentPageDisplay];
}
```



## Thảo luận

