Lập trình ứng dụng trên iOS

Objective-C

Nội dung

- Giới thiệu về Objective-C
- Quy tắc đặt tên lớp, tên phương thức, tên biến
- Class trong Objective-C
- Phạm vi truy xuất các biến thể hiện
- Property trong Objective-C
- Phương thức trong Objective-C
- Category trong Objective-C
- Protocol trong Objective-C
- Kế thừa
- Xử lý ngoại lệ
- Một số cấu trúc điều khiển trong Objective-C
- Quản lý bộ nhớ
- Selector trong Objective-C

Giới thiệu về Objective-C

- Ngôn ngữ lập trình hướng đối tượng
- ANSI C + Smalltalk = Objective-C
- Objective-C là ngôn ngữ lập trình chính được Apple chọn để viết các ứng dụng cho hệ điều hành Mac OS, iOS

--> Dễ dàng nắm bắt được ngôn ngữ Objective-C nếu có "kinh nghiệm" về C

Quy tắc đặt tên

- Class
  - Bắt đầu bằng ký tự hoa
- Method
  - Bắt đầu bằng ký tự thường
- Variable
  - Giống method

```
@interface MyClass : NSObject {
}
@end
```

```
- (void) findMe;
```

```
NSString *stringURL;
```

MyClass.h

```
@interface MyClass : NSObject{
    NSString *stringURL;
}
- (void) findMe;
@end
```

Class trong Objective-C

- @interface: Khai báo class
- @implementation: Định nghĩa class
- @end: Kết thúc khai báo class hoặc kết thúc định nghĩa class

```
// .h
@interface MyClass : NSObject
{
    // ...
}
@end
```

```
// .m
@implementation MyClass
    // ...
@end
```

Phạm vi truy xuất các biến

- @private: Giới hạn phạm vi trong lớp mà biến thể hiện được khai báo
- @protected (default): Giới hạn phạm vi trong lớp và lớp con kế thừa mà biến thể hiện được khai báo
- @public: Không giới hạn phạm vi truy xuất

Phạm vi truy xuất các biến (tt)

- Ví dụ:

```
11 @interface MyClass : NSObject {
12 // Private
13 @private
14     NSString *str1;
15
16 // Protected (default)
17     NSString *str2;
18
19 // Public
20 @public
21     NSString *str3;
22 }
23 @end
```

Property trong Objective-C

- Property cho phép định nghĩa các bộ truy xuất (setter/getter)

  --> Thuận lợi cho việc truy xuất đến các biến thể hiện
- Định nghĩa trong file .h

  @property (<attributes>) type propertyName;
- Thực thi trong file .m

  @synthesize propertyName;

Property trong Objective-C (tt)

- Các thuộc tính (attributes)
  - readwrite (default): đọc và ghi (get/set)
  - read-only: chỉ có thể đọc (get)
  - assign (default): Thường dùng với các kiểu vô hướng như NSInteger, CGFloat, CGrect,...
  - retain: Thường được sử dụng cho đối tượng
  - copy: Tạo một bản sao của đối tượng sẽ được sử dụng cho đối tượng chỉ định
  - atomic (default): Thực hiện đồng bộ hoá
  - nonatomic: Ngược với atomic

Property trong Objective-C (tt)

- Các thuộc tính (attributes)
  - atomic

```
// @property (retain) UITextField *username;
// Generates roughly
- (UITextField *) username {
    UITextField *retval = nil;
    @synchronized(self) {
        retval = [[username retain] autorelease];
    }
    return retval;
}
- (void) setUsername : (UITextField *) _username {
    @synchronized(self) {
        [_username retain];
        [username release];
        username = _username;
    }
}
```

Property trong Objective-C (tt)

- Các thuộc tính (attributes)
  - nonatomic

```
// @property (retain, nonatomic) UITextField *username;
// Generates roughly
- (UITextField *) username {
    return username;
}
- (void) setUsername : (UITextField *) _username {
    [_username retain];
    [username release];
    username = _username;
}
```

Phương thức trong Objective-C

- Class method
  - Ký hiệu dấu +
  - Truy xuất thông qua tên lớp
  - Không thể truy cập vào các biến thể hiện (Instance variables)
- Instance method
  - Ký hiệu dấu -
  - Truy xuất thông qua đối tượng

Phương thức trong Objective-C (tt)

- Khai báo phương thức:

```
-/+ (return_type) method;
-/+ (return_type) methodPara1 : (type) para1;
-/+ (return_type) methodPara1 : (type) para1 andPara2: (type) para2;
```

- Gọi phương thức:

```
// Class method
MyClass *myClass = [MyClass method];

// Instance method
[object method];
[object methodPara1:pa1];
[object methodPara1:pa1 andPara2:pa2];
```

Category trong Objective-C

- Khi muốn thêm một số phương thức vào một lớp có sẵn
  - Mở rộng lớp đó bằng cách viết lại mã nguồn
  - Mở rộng lớp đó không cần viết lại mã nguồn
    --> Category

MyClass.h

```
@interface MyClass : NSObject{
}
- (void) showMe;
@end
```

MyClass.m

```
@implementation MyClass
- (void) showMe {
      // show me ...
}
@end
```

Category trong Objective-C (tt)

- Thêm một phương thức newMethod cho lớp MyClass thông qua Category

MyClass+MyCategory.h

```
@interface MyClass (MyCategory)
- (void) newMethod;
@end
```

MyClass+MyCategory.m

```
@implementation MyClass (MyCategory)
- (void) newMethod {
      // to do new method ...
}
@end
```

Chú ý: - Tên của Category là duy nhất (Không được trùng)
- Trong Category không cho phép thêm các biến thể hiện

Procol trong Objective-C

- Protocol là gì ?
- Ví dụ:

```
@protocol MyProtocol
// required (default)
- (void) requiredMethod;

@optional
- (void) anOptionalMethod;
- (void) anotherOptionalMethod;

@required
- (void) anotherRequiredMethod;

@end
```

Kế thừa

- Tương tự như các ngôn ngữ khác
- Trong Objective-C, root class của tất cả các class là NSObject

Person.h

```
@interface Person : NSObject {
    NSString *name;
    NSString *birthday;
}
// ...
@end
```

Teacher.h

```
@interface Teacher : Person{
    float fSalary;
}
// ...
@end
```

Xử lý ngoại lệ

```
@try {
    // trying ...
}
@catch (NSException *exception) {
    // handler ...
}
@finally {
    // final ...
}
```

```
@try {
    NSFileManager *fileManager = [NSFileManager defaultManager];
    if (![fileManager fileExistsAtPath:[[[NSBundle mainBundle]
        resourcePath] stringByAppendingPathComponent:@"abc.txt"]]) {
         @throw  [NSException exceptionWithName:@"Error!!!" reason:
             @"This file doesn't exist" userInfo:nil];
    }
   // Do something ...
}
@catch (NSException *exception) {
    UIAlertView *alert = [[[UIAlertView alloc] initWithTitle:[exception
        name] message:[exception reason] delegate:self
        cancelButtonTitle:nil otherButtonTitles:@"OK", nil] autorelease
        ];
    [alert show];
}
@finally {
    // Do something ...
}
```

Một số cấu trúc điều khiển

- If
- For
- Switch
- While

```
if (condition) {
    statements
}
```

```
if (condition) {
    statements-if-true
} else {
    statements-if-false
}
```

```
for (initialization; condition; increment) {
    statements
}
```

```
for (type *object in collection) {
    statements
}
```

```
do {
    statements
} while (condition);
```

```
while (condition) {
    statements
}
```

Quản lý bộ nhớ

- Các nguyên tắc quản lý bộ nhớ
- Vấn đề khi khởi tạo đối tượng
- Release
- Retain
- Dealloc
- Tham chiếu yếu

Quản lý bộ nhớ

- Các nguyên tắc quản lý bộ nhớ
  - Nguyên tắc căn bản
    - Khi bạn nắm quyền sở hữu một đối tượng, khởi tạo đối tượng bằng các phương thức mà trong tên bắt đầu với với alloc hoặc new hoặc copy (ví dụ, alloc, newObject hoặc mutableCopy…) hoặc gửi một thông điệp retain, bạn phải có trách nhiệm giải phóng quyền sở hữu đối tượng đó bằng cách sử dụng release hoặc autorelease. Bất kỳ khi nào bạn nhận được một đối tượng (không phải tự mình khởi tạo), bạn không được release nó.
  - Nguyên tắc khác
    - Khi bạn cần lưu trữ một đối tượng được nhận như một property trong một biến thể hiện, bạn phải retain hoặc copy nó. (Điều này không đúng cho cho tham khảo yếu, nhưng đây là điển hình hiếm).
    - Một đối tượng được nhận thường đảm bảo vẫn có hiệu lực trong phương thức mà nó đã được nhận (ngoại trừ trong các ứng đa luồng và vài trường hợp Distributes Objects). Phương thức đó có thể trả về an toàn đối tượng mà nó được triệu gọi. Sử dụng retain trong việc kết hợp với release hoặc autorelease khi cần thiết để bảo vệ một đối tượng khỏi hiệu lực của các thông điệp không hợp lệ bên ngoài.
    - autorelease có nghĩa là "gửi một thông điệp release sau đó

Quản lý bộ nhớ

- Vấn đề khi khởi tạo đối tượng

```
MyClass *myClass = [MyClass alloc];
[myClass init];
if(myClass) {
    //...
}
```

Quản lý bộ nhớ

- Vấn đề khi khởi tạo đối tượng (tt)

```
MyClass *myClass = [[MyClass alloc] init];
if(myClass) {
    //...
}
```

Chú ý: - Luôn luôn trả về nil trong phương thức init nếu có lỗi xảy ra
- Phải kết hợp 02 lời gọi phương thức alloc và init

Quản lý bộ nhớ

- Release
  - Chỉ được release khi khởi tạo đối tượng một cách thủ công alloc
  - KHÔNG release thủ công một đối tượng autorelease --> CRASH

```
// Must release string1 when done
NSString *string1 = [[NSString alloc] init];
// ...
[string1 release];
```

Autorelease

release

```
// string2 will be released automatically
NSString *string2 = [NSString string];
```

Quản lý bộ nhớ

- Retain
  - Mỗi đối tượng có một bộ đếm được sử dụng để kiểm soát tất cả các tham chiếu bởi đối tượng hoặc nó có
  - Phương thức alloc, new, copy và retain đều làm tăng bộ đếm lên 1
  - Phương thức release giảm bộ đếm đi 1
  - Khi bộ đếm có giá trị bằng 0 --> Phương thức dealloc của đối tượng sẽ được gọi
  - Để xác định giá trị của bộ đếm: [object retainCount]

Quản lý bộ nhớ

- Dealloc
  - Được gọi khi đối tượng đang được remove khỏi bộ nhớ
  - Nếu một lớp có các biến thể hiện (instance variable) là các đối tượng thì trong phương thức dealloc của lớp phải thực hiện giảii phóng các biến thể hiện này

```
- (void) dealloc {
      [childVar1 release];
      [childVar2 release];
      //...
      [super dealloc];
}
```

Quản lý bộ nhớ

- Tham chiếu yếu
  - Tạo tham chiếu đến đối tượng mà không cản trở đối tượng tự giải phóng chính nó --> Thiết lập tham chiếu "yếu" đến đối tượng

Selector trong Objective-C

- Selector trong Objective-C có 02 ý nghĩa:
  - Chỉ đến tên của một phương thức khi nó được sử dụng trong mã nguồn một thông điệp gởi đến một đối tượng
  - Chỉ đến một định danh duy nhất mà thay thế cho một tên khi mã nguồn được biên dịch