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

Lập trình iOS

Bài 6. *Singleton & Category*

Ngành Mạng & Thiết bị di động

2015

Nội dung

1. **Singleton**
   - Giới thiệu về Singleton
   - Sử dụng Singleton trong ứng Objective-C
2. **Category**
   - Mục đích sử dụng trong ứng dụng
   - Xây dựng Category trong ứng dụng iOS

1.1 Giới thiệu về Singleton

- Là một trong những kỹ thuật phổ biến nhất của kỹ thuật thiết kế mẫu (Design Pattern) trong công nghệ phát triển phần mềm.
- Là cách thức được dùng chủ yếu để chia sẻ dữ liệu và tài nguyên giữa các thành phần, các lớp một cách hiệu quả.
- Trong ứng dụng iOS, Singleton được sử dụng để chia sẻ các thể hiện, dữ liệu của lớp này cho lớp kia.
- Ví dụ: lớp *UIApplication* có một phương thức gọi là *sharedApplication* có thể gọi ở bất cứ nơi nào, nó sẽ trả về một thể hiện (instance) liên quan đến ứng dụng đang chạy.

1.1 Giới thiệu về Singleton

- Trong Objective-C, thường các phương thức dạng này không cố định tên nhưng nó nên bắt đầu với "shared", theo sau đó là lớp đối tượng. Ví dụ như đối với *UIApplication* là *sharedApplication*, đối với lớp *MyManager* là *sharedManager*.

- Tuy nhiên vẫn có một số ngoại lệ như lớp *NSFileManager* sử dụng phương thức *defaultManager* như là một singleton để chia sẻ tài nguyên.

1.2 Sử dụng Singleton trong ứng dụng iOS

- Giả sử ta có lớp MyManager sử dụng singleton để chia sẻ dữ liệu. Trong tập tin MyManager.h ta khai báo như sau:

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

@interface MyManager : NSObject {
    NSString *someProperty;
}
@property (nonatomic, retain) NSString *someProperty;
+ (id)sharedManager;
@end
```

1.2 Sử dụng Singleton trong ứng dụng iOS

- Trong tập tin MyManager.h ta thực hiện như sau:

```
#import "MyManager.h"

@implementation MyManager
@synthesize someProperty;
#pragma mark Singleton Methods
+ (id)sharedManager {
    //Khoi tao doi tuong sharedMyManager thong qua ham init cua lop.
    static MyManager *sharedMyManager = nil;
    if (sharedMyManager == nil)
        sharedMyManager = [[self alloc] init];

    return sharedMyManager;
}
//Khoi tao gia tri ban dau cho thuoc tinh someProperty
- (id)init {
    if (self = [super init]) {
        someProperty = @"Default Property Value";
    }
    return self;
}
- (void)dealloc {
    // Should never be called, but just here for clarity really.
}
@end
```

1.2 Sử dụng Singleton trong ứng dụng iOS

- Để sử dụng Singleton ở nơi nào khác trong ứng dụng, ta chỉ cần gọi hàm như sau:

```
MyManager *sharedManager = [MyManager sharedManager];
```

Nội dung

1. **Singleton**
   - Giới thiệu về Singleton
   - Sử dụng Singleton trong ứng Objective-C

2. **Category**
   - Mục đích sử dụng trong ứng dụng
   - Xây dựng Category trong ứng dụng iOS

2.1 Mục đích sử dụng Category trong ứng dụng

- Category cung cấp một khả năng thêm các chức năng vào cho một đối tượng mà không cần tạo các lớp con hay thay đổi đối tượng đó.

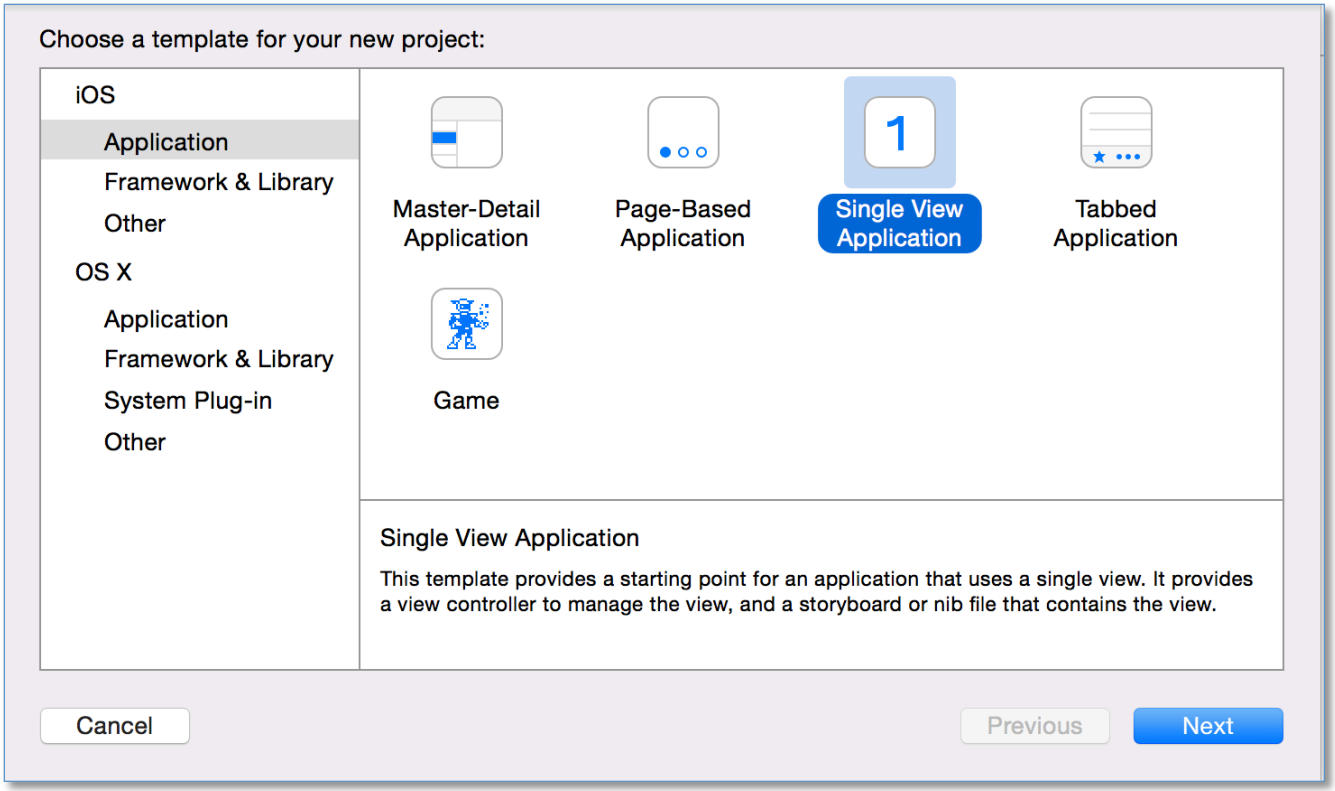
- Category thường được dùng để thêm vào một phương thức cho một lớp có sẵn ví dụ như lớp NSString hoặc lớp tuỳ chọn của mình.

2.2 Xây dựng Category trong ứng dụng iOS

- Ta sẽ minh hoạ việc xây dựng bổ sung một phương thức dành cho lớp đã được định nghĩa sẵn là NSString. Phương thức này thực hiện việc tách các ký tự dạng số khỏi chuỗi được truyền vào.

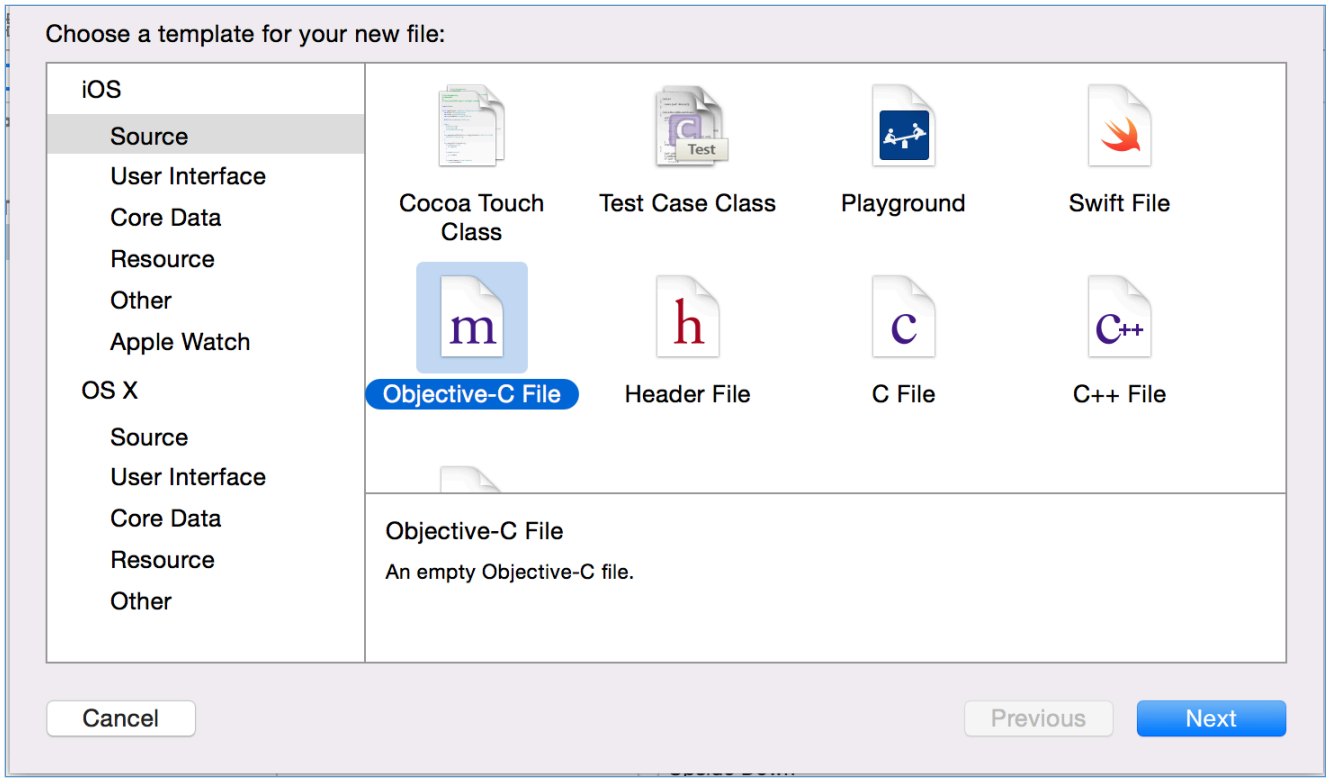
2.2 Xây dựng Category trong ứng dụng iOS

- **Bước 1: Tạo mới Project dạng Single view Application**



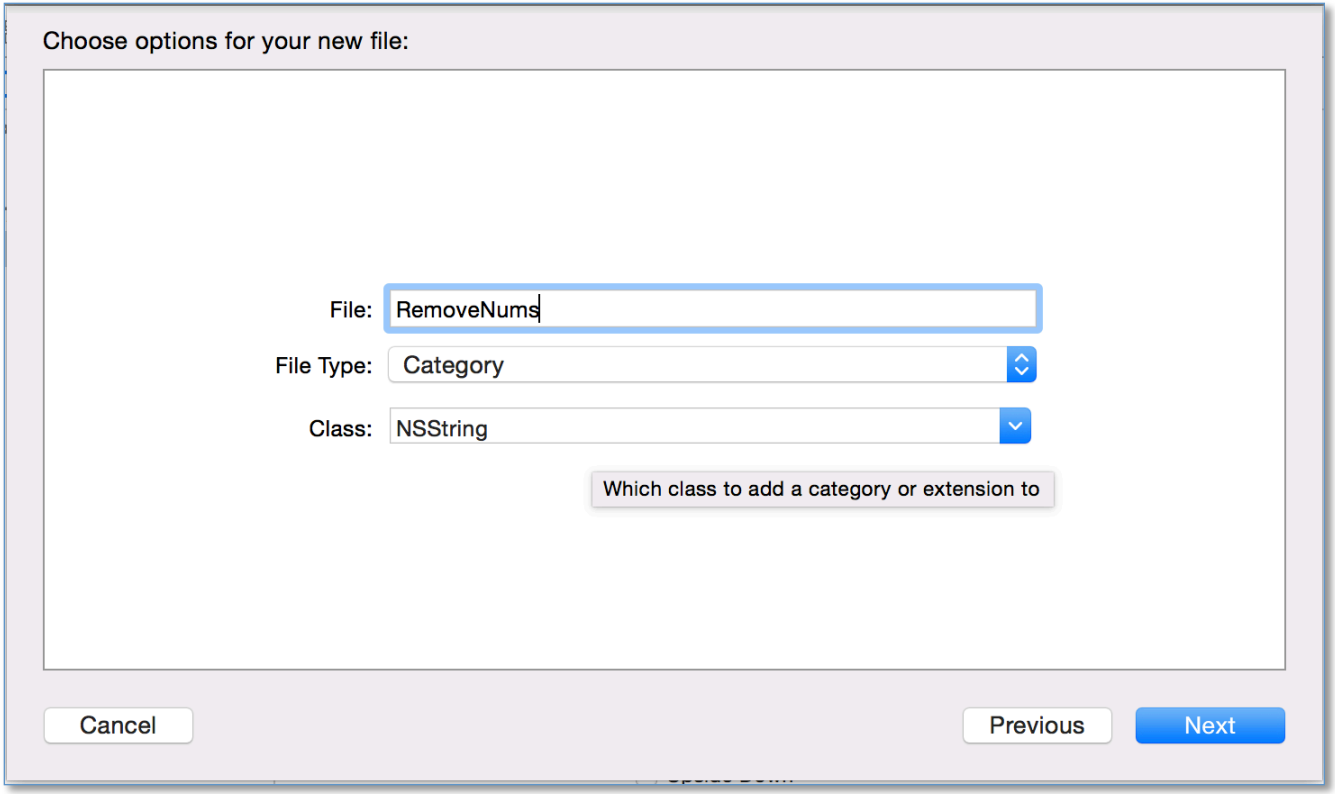
2.2 Xây dựng Category trong ứng dụng iOS

- Bước 2: Tạo Category. Thêm mới file vào project, chọn iOS-> Source-> Objective-C File



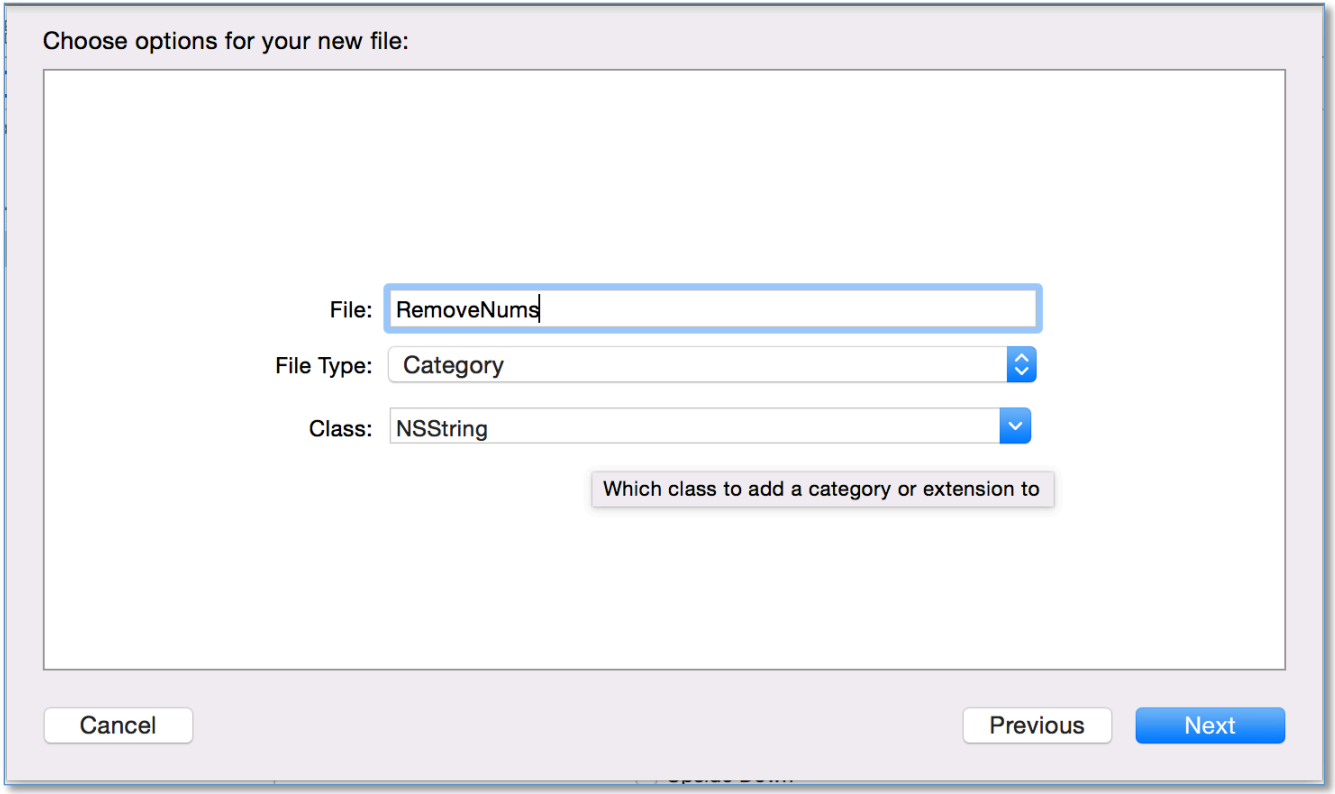
2.2 Xây dựng Category trong ứng dụng iOS

- Bước 3: Đặt tên file là RemoveNums và chọn File Type là *Category*, Class tương ứng là *NSString*-> Nhấn Next



2.2 Xây dựng Category trong ứng dụng iOS

- Bước 3: Đặt tên file là RemoveNums và chọn File Type là *Category*, Class tương ứng là *NSString*-> Nhấn Next



2.2 Xây dựng Category trong ứng dụng iOS

- Bước 4: Khai báo và sử dụng phương thức tách số khỏi chuỗi. Trong file NSString+RemoveNums.h, ta khai báo phương thức để tách số khỏi chuỗi.

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

@interface NSString (RemoveNums)
-(NSString *)removeNumberFromString:(NSString *)string;
@end
```

2.2 Xây dựng Category trong ứng dụng iOS

- Trong file NSString+RemoveNums.m, ta định nghĩa phương thức đã khai báo trong file .h.

```
#import "NSString+RemoveNums.h"

@implementation NSString (RemoveNums)
-(NSString *)removeNumberFromString:(NSString *)string{
    NSString *trimedString = nil;
    //Khoi ta bo cac ky tu can tach, o day ta can tach cac ky tu dang so
    NSCharacterSet *numbersSet = [NSCharacterSet characterSetWithCharactersInString:@"0123456789"];
    //Goi ham cat cac ky tu so khoi chuoi duoc truyen vao
    trimedString = [string stringByTrimmingCharactersInSet:numbersSet];

    return trimedString;
}
@end
```

2.2 Xây dựng Category trong ứng dụng iOS

- Trong file ViewController.m, import lớp NSString +RemoveNums.h

```
#import "NSString+RemoveNums.h"
```

- Trong hàm viewDidLoad, ta sử dụng như sau:

```
- (void)viewDidLoad {
    [super viewDidLoad];
    //Tao chuoi voi ky tu alphabet va so
    NSString *stringWithNums = @"ABC 123";
    NSLog( @"stringWithNums: %@",stringWithNums);
    //Goi ham tach so khoi chuoi, giong nhu goi mot Phuong thuc cua lop NSString
    stringWithNums = [stringWithNums removeNumberFromString:stringWithNums];
    //In ra console chuoi sau khi da tach so
    NSLog(@"string without nums: %@",stringWithNums);

}
```

Thảo luận