

Objective-C 学习之路 使用 Xcode

整理：CoderDream

日期：2011-08-10

Xcode 功能很多，以下介绍常用的功能，帮助 Objective-C 开发人员提高编码和调试效率。

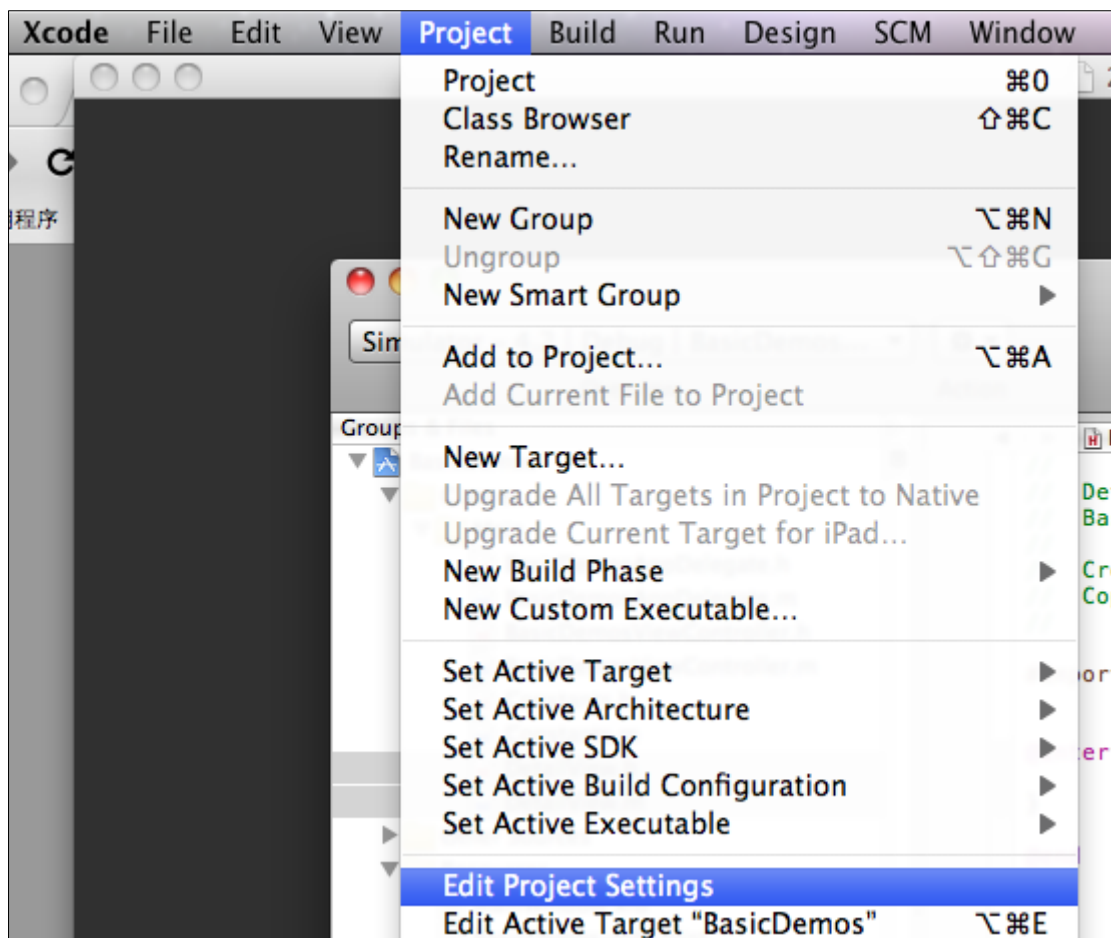
1. 改变公司名称

通过 Xcode 编写代码，代码的头部会有类似下面的内容：

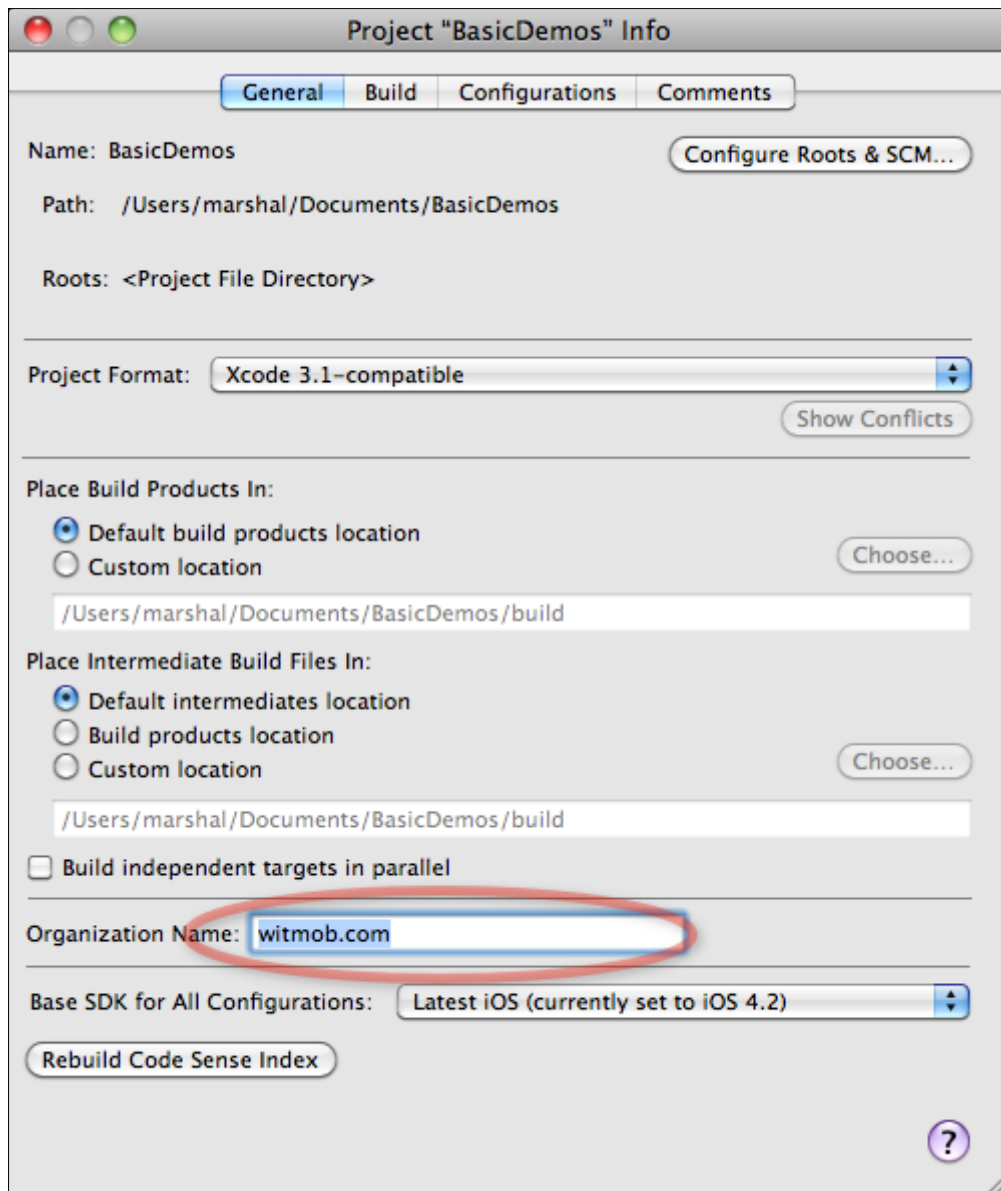
```
//  
// BasicDemosViewController.m  
// BasicDemos  
//  
// Created by Marshal Wu on 11-4-28.  
// Copyright 2011 __MyCompanyName__. All rights reserved.  
//  
#import "BasicDemosViewController.h"
```

应该将这个内容改为公司或者项目的名称。

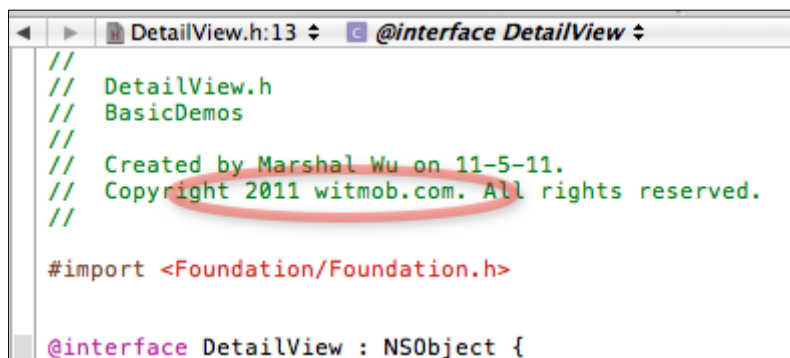
在 Xcode 3.2.x 之前，需要命令行设置变量。比如《Objective-c 基础教程》第七章中提到的方式。之后，可以通过 Xcode 的配置项操作了。操作步骤见下面图示：



然后：

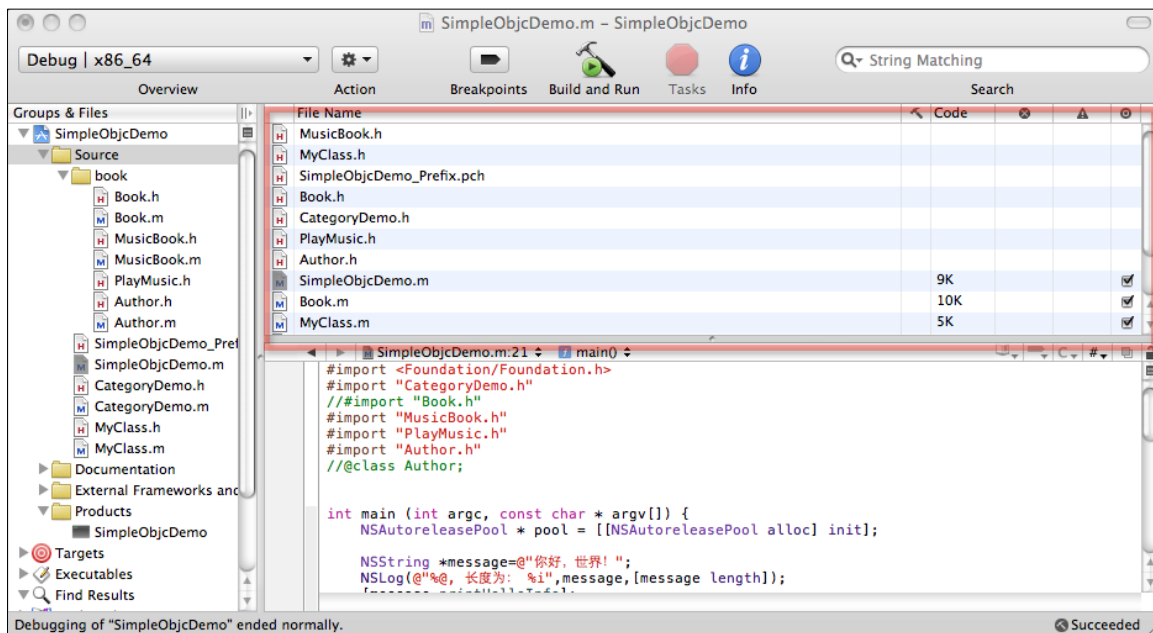


这样，再创建文件，就有类似这样的效果了：



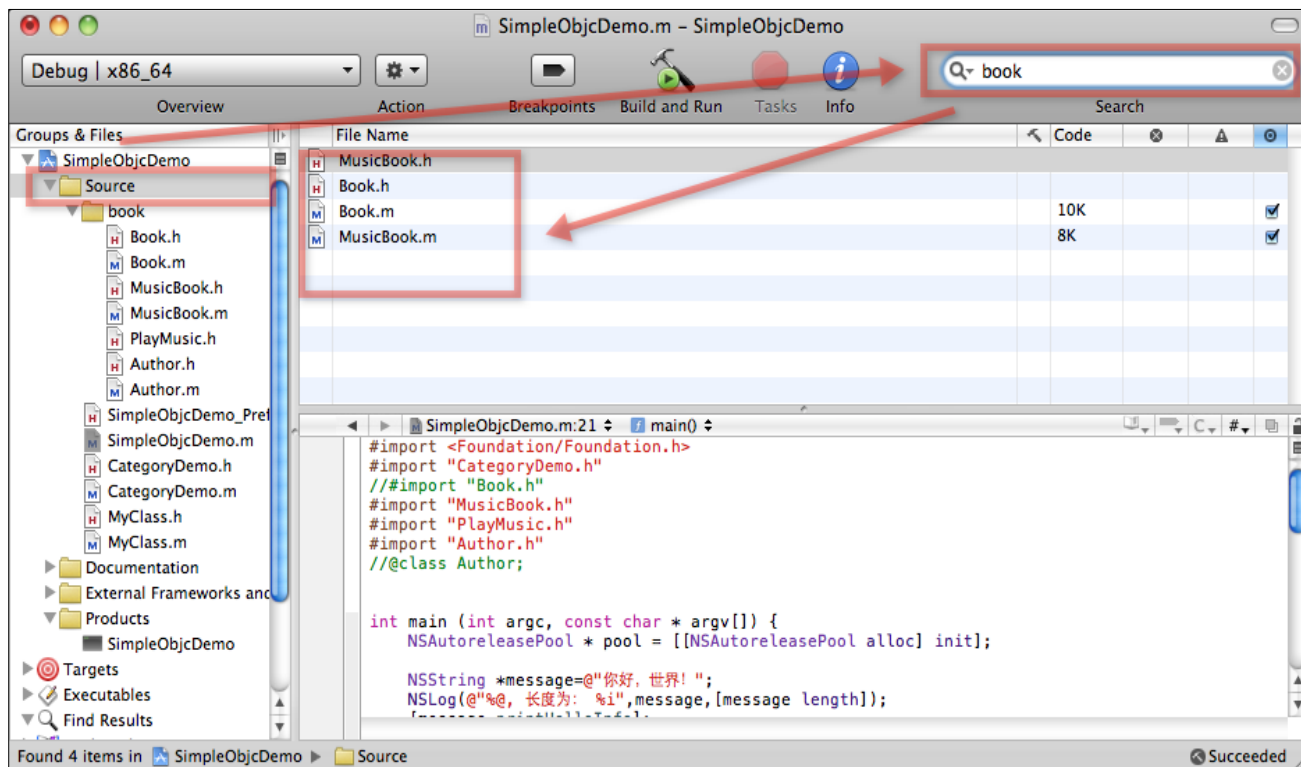
2. 通过搜索框缩小文件范围

当项目开发到一段时间后，源代码文件会越来越多。再从 Groups & Files 的界面去点选，效率比较差。可以借助 Xcode 的浏览器窗口，即：



这里插一句，如果讨厌显示这个窗口，也可以通过快捷键：shift+command+e 来切换是否显示。

还是继续说搜索框缩小文件范围。上图的搜索框，可以输入关键字，这样浏览器窗口里只显示带关键字的文件了。比如我只想看 Book 相关的类。



3. 如何格式化代码

比如下面这段代码：

```
SimpleObjcDemo.m:21  f main()
//#import "Book.h"
#import "MusicBook.h"
#import "PlayMusic.h"
#import "Author.h"
//@class Author;

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

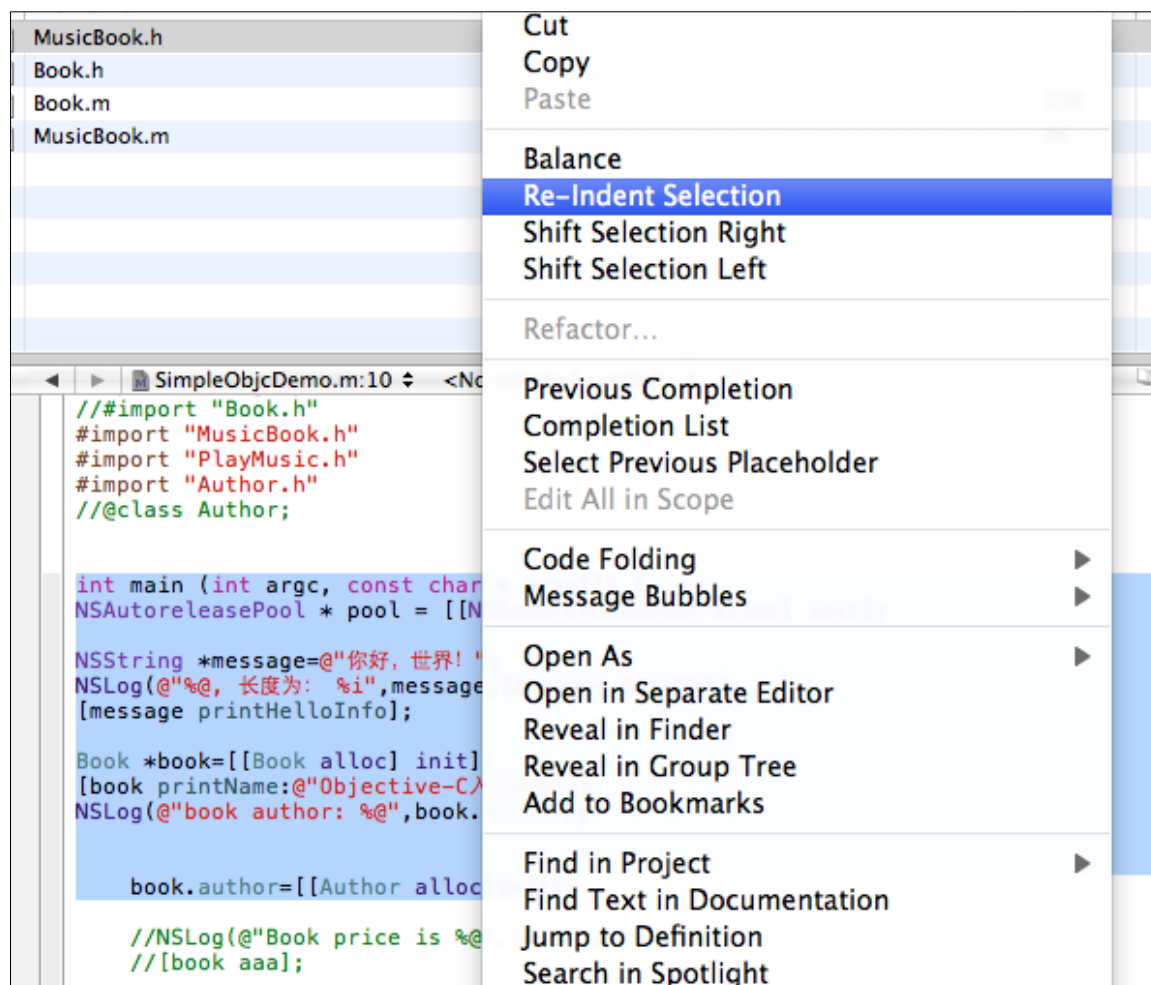
    NSString *message=@"你好, 世界! ";
    NSLog(@"%@, 长度为: %i",message,[message length]);
    [message printHelloInfo];

    Book *book=[[Book alloc] init];
    [book printName:@"Objective-C入门手册"];
    NSLog(@"book author: %@",book.author);

    book.author=[[Author alloc]init];

    //NSLog(@"Book price is %@", [book costPrice]);
}
```

很多行都顶格了。需要进行格式化。可以选中需要格式化的代码，然后在上下文菜单中找：



这是比较规矩的办法。Xcode 没有提供快捷键，当然自己可以设置。我又比较喜欢用快捷键。我的做法是：ctrl+a（全选文字），ctrl+x（剪切文字），ctrl+v（粘贴文字）。Xcode 会对粘贴的文字格式化。

4. 如何缩进代码

代码有的时候要缩进，有的时候又要做相反的操作。单行缩进和其他编辑器类似，tab 键即可。如果选中多行呢？需要快捷键了。command+]表示缩进，command+[表示反向缩进。

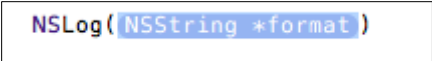
5. 代码的自动完成

使用 IDE 工具的一大好处是，工具能够帮助我们自动完成比如冗长的类型名称。Xcode 提供了这方面的功能。

比如上面提到的输出日志：

```
01. NSLog(@"book author: %@",book.author);
```

如果都自己敲，很麻烦的。可以先敲 ns，然后快捷键：ctrl+.，会自动出现：

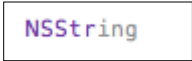


NSLog(NSString *format)

然后填写参数就行了。ctrl+.快捷键的功能是，自动给出第一个匹配 ns 关键字的函数或类型。刚好 NSLog 是第一个。如果继续 ctrl+.，则会出现比如 NSString。以此类推，会显示所有 ns 开头的类型或函数，循环往复。

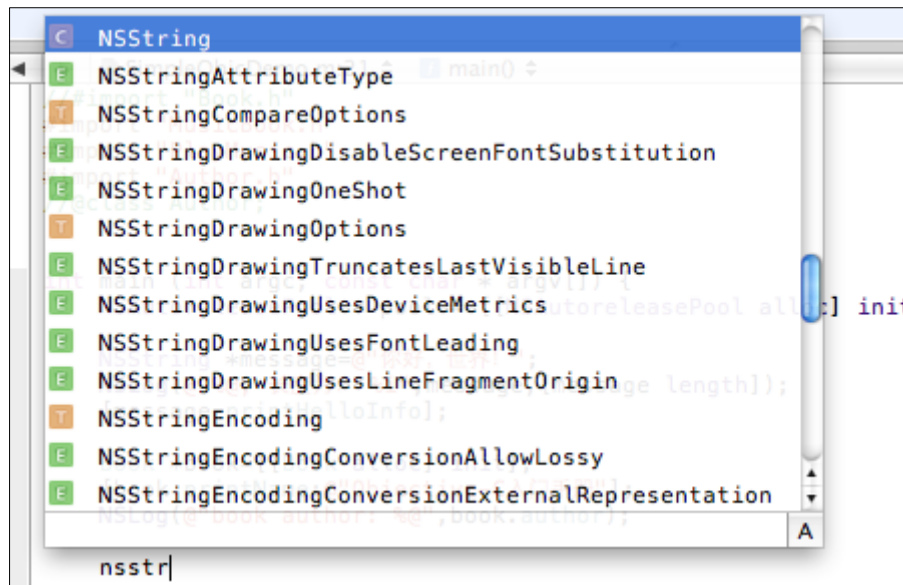
或者，也可以用 ctrl+,快捷键，比如还是 ns，那么会显示全部 ns 开头的类型、函数、常量等的列表。可以在这里选择。

其实，Xcode 也可以在你敲代码的过程中自动给出建议。比如咱们要敲 NSString。当敲到 NSStr 的时候：



NSString

后面的ing就自动冒出来了。如果和你预想的一样，直接按 tab 键确认即可。也许你想输入的是 NSStream，那么可以继续敲。另外，也可敲 esc 键，这时就会出现结果列表供选择了。



如果是正在输入方法，那么会自动完成比如下面的样子：

```
Book *book=[[Book alloc] init];  
[book printName:(NSString *)bookName authorName:(NSString *)name]
```

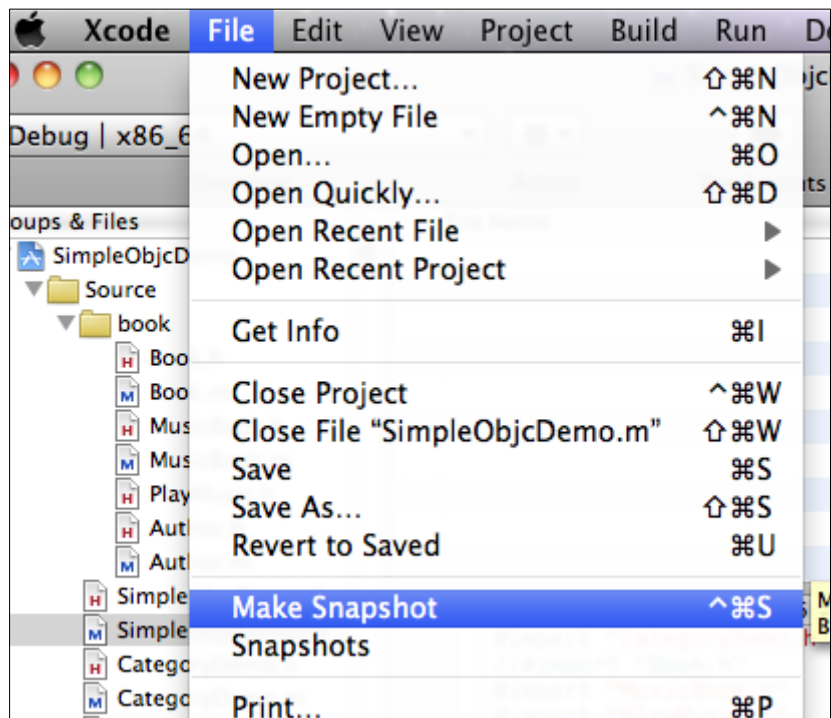
可以 tab 键确认方法中的内容，继续。或者可通过快捷键 ctrl+/, 在方法中的参数来回切换。

6. 设置项目快照以及恢复到快照

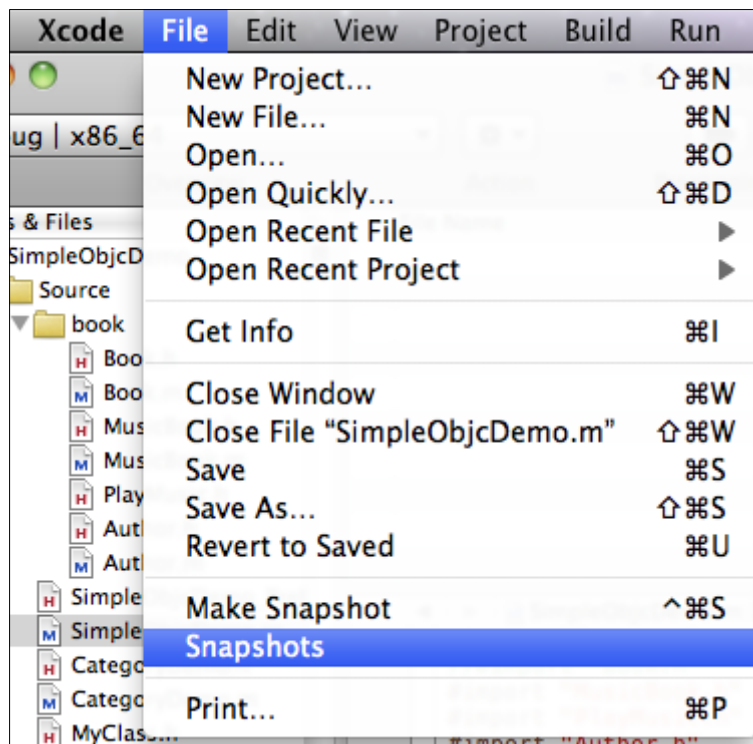
其实在不用 Xcode 之前，我根本没有使用这种东西的需求。如果使用 Eclipse，我习惯把代码提交到 SVN 上，并借助 SVN 的 copy 功能实现服务器端的快照。

Xcode 上使用版本控制不是很方便。因此本地快照功能还是很值得使用的。

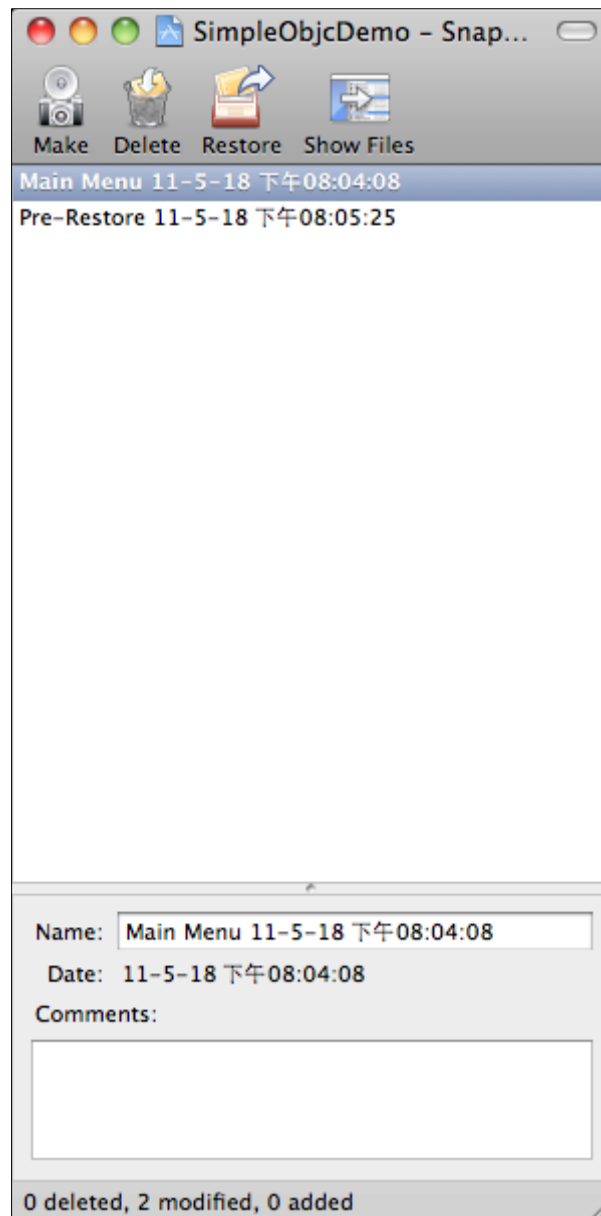
快照 (snapshot)，主要作用是，创建快照，好比，给你的项目拍了个照。然后你可以随便修改代码了，不必担心改乱了无法回退到之前的版本。如果确实改乱了，恢复到快照就可以了。恢复后，好像什么也没发生过。



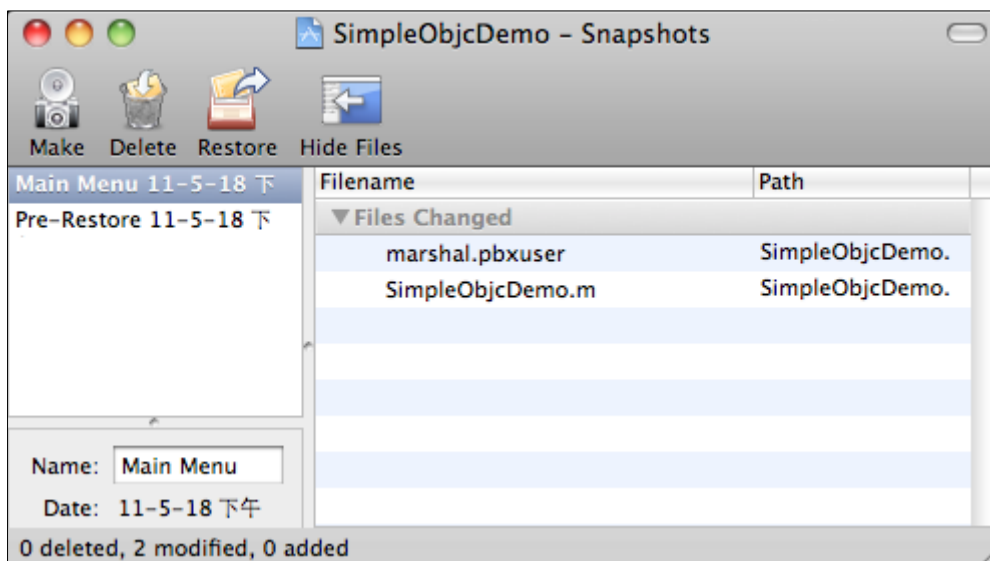
可以通过 make snapshot 创建快照，或者快捷键 `ctrl+command+s`。
想要恢复的时候：



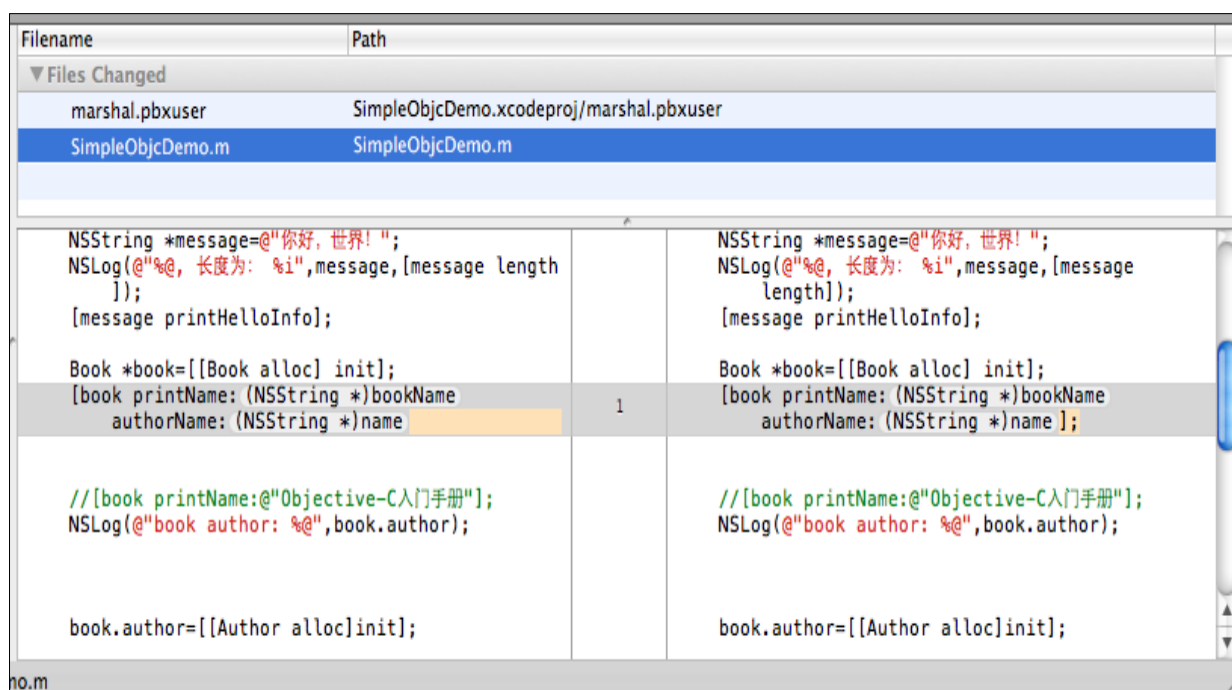
然后选中做快照的版本：



make 按钮可拍照当前项目，生成新的快照。可在 comments 中写下该快照的备注信息，便于以后恢复时辨别。delete 按钮可删除不必要的快照。restore，将用选中的快照覆盖当前项目。show files 可列出选中快照和当前项目文件的差异。



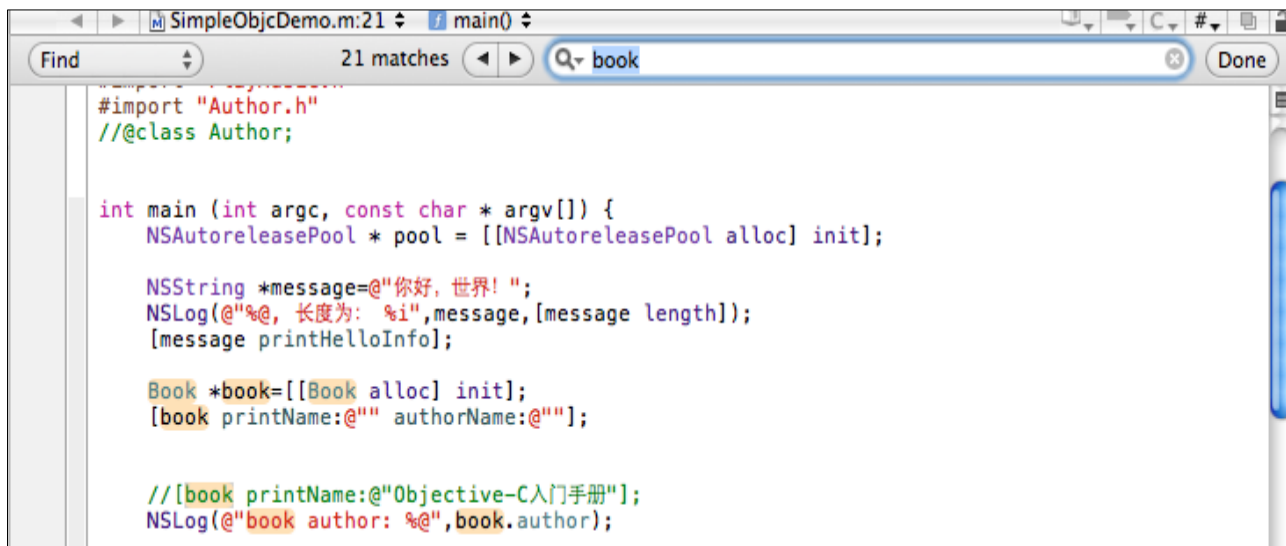
上图列出了有两个文件不同。再选中文件：



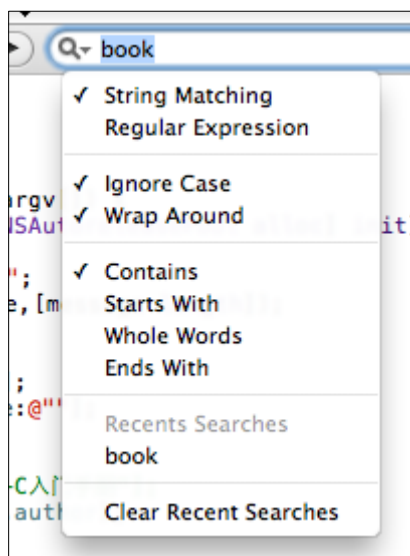
可以看到不同的地方给出了标注。

7. 文件内查找和替代

代码中经常会做查找和替代的操作。如果只是查找。直接按 `command+f` , 代码的右上角会出现对话框：

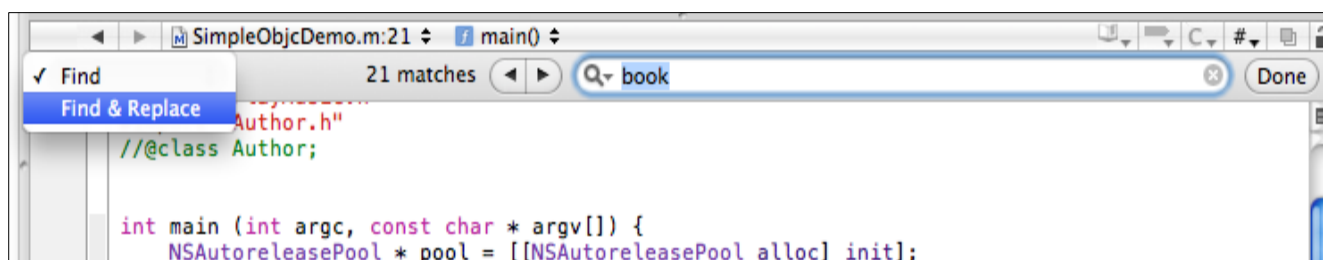


在里面输入关键字，不论大小写，代码中所有命中的文字都高亮显示。

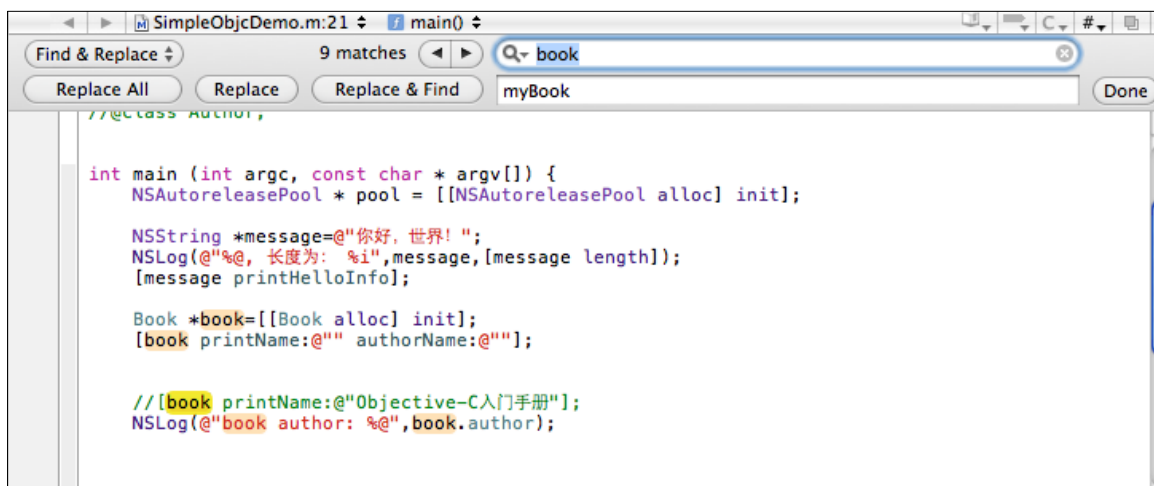


可以做更复杂的查找，比如是否大小写敏感，是否使用正则表达式等等。

可以切换到替代界面：



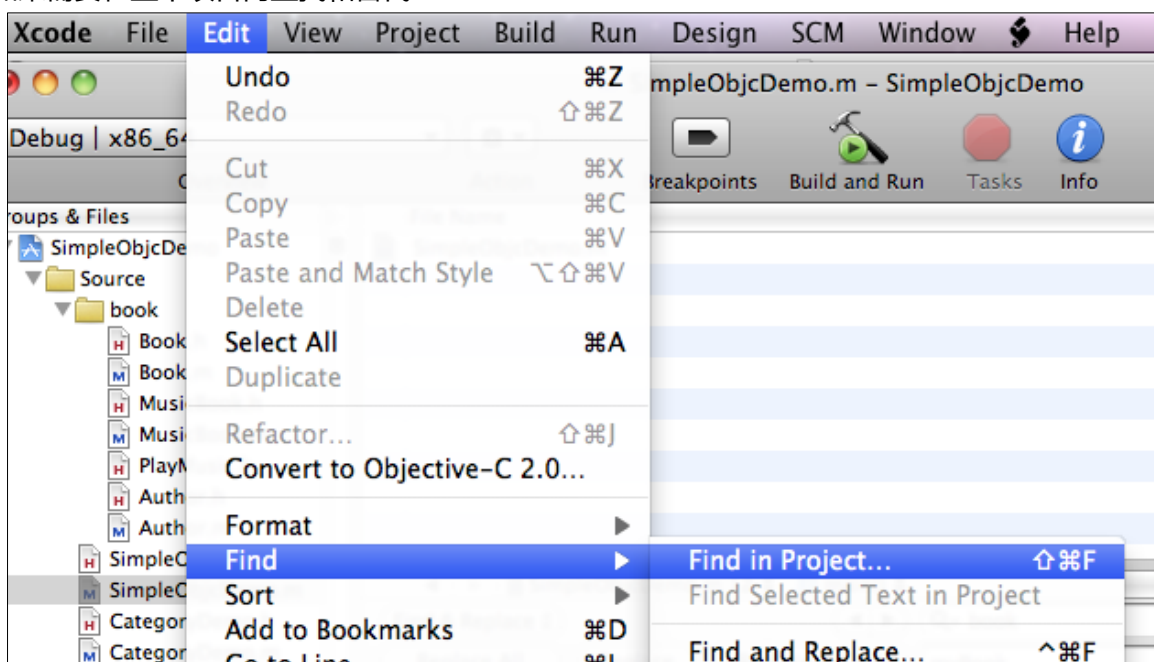
下面的界面，我是将查找设置为大小写敏感，然后替代为 myBook：



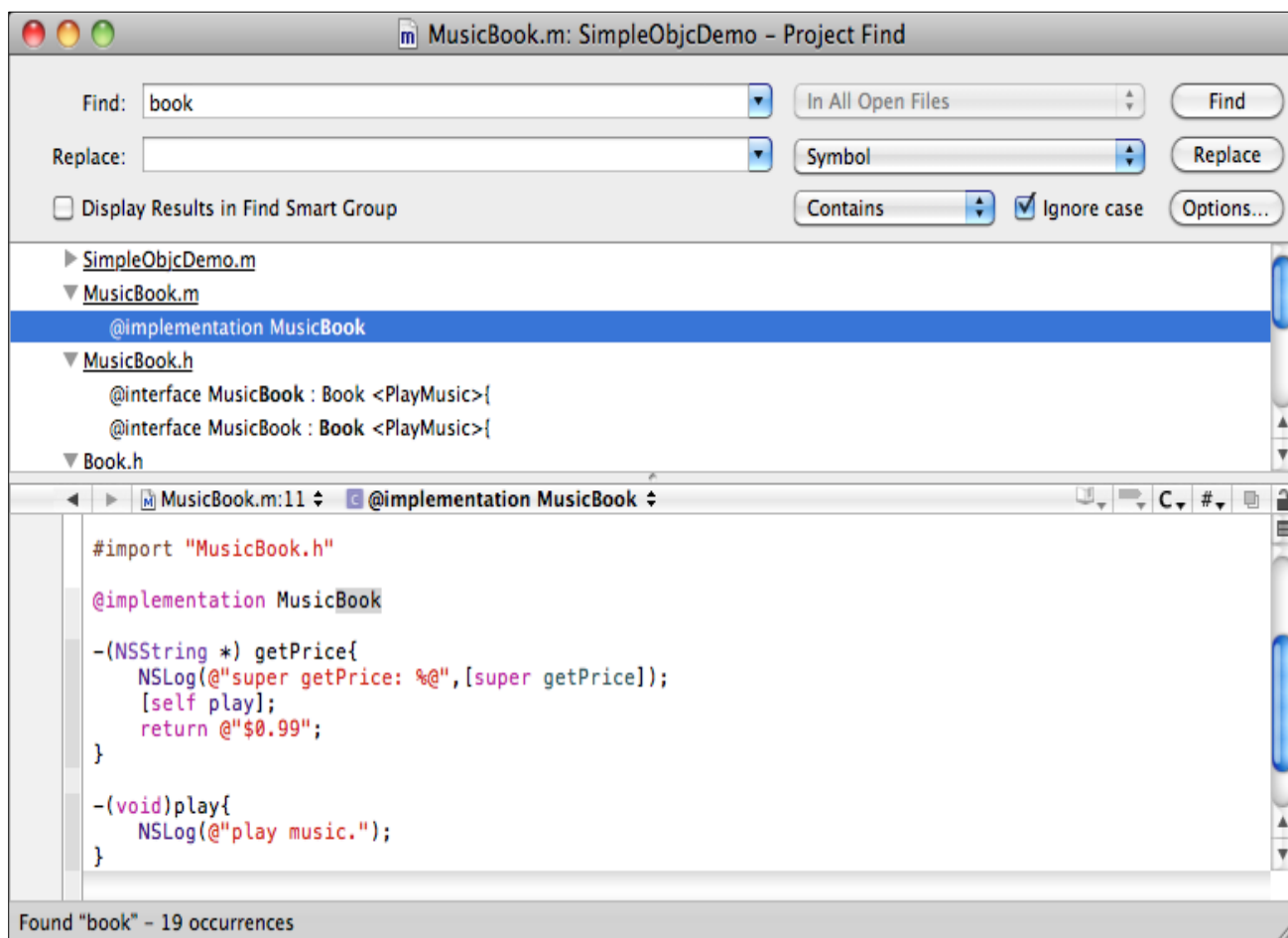
可点击按钮是否全部替代，还是查找一个替代一个等。

8. 项目内查找和替代

如果需要在整个项目内查找和替代：



还是找关键字 book：



替代就不说了，如何操作一看便知。

9. 作用域内编辑

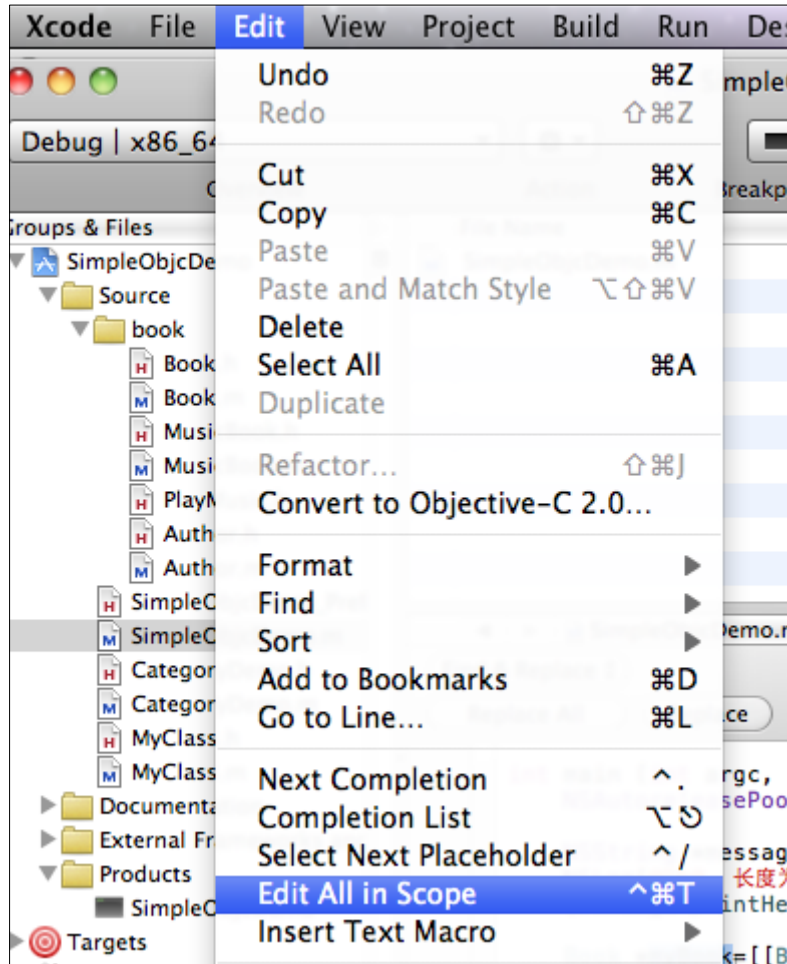
比如：



我想把局部变量 myBook 改回成 book。那么可以用到这个功能。首先要鼠标选中变量：

```
Book *myBook=[[Book alloc] init];
[myBook printName:@" " authorName:@" "];
```

然后：



这时会看到：

```
int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

    NSString *message=@"你好, 世界! ";
    NSLog(@"%@, 长度为: %i",message,[message length]);
    [message printHelloInfo];

    Book *myBook=[[Book alloc] init];
    [myBook printName:@" " authorName:@" "];

    //[myBook printName:@"Objective-C入门手册"];
    NSLog(@"myBook author: %@",myBook.author);

    myBook.author=[[Author alloc]init];
}
```

直接修改变量名，发现所有该变量名同时跟着改变了：

```

int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

    NSString *message=@"你好, 世界! ";
    NSLog(@"%@, 长度为: %i",message,[message length]);
    [message printHelloInfo];

    Book *myBook=[[Book alloc] init];
    [myBook printName:@" " authorName:@""];

    //[myBook printName:@"Objective-C入门手册"];
    NSLog(@"myBook author: %@",myBook.author);

    myBook.author=[[Author alloc]init];

```

10. 重构代码

重构 (refactor) 的概念这里不展开说了。读者可参考专门的论述。这里只举具体例子。比如修改类的名称，就是一种**重构**行为。Xcode 提供了这方面的支持。

比如想把 Book 类改为 GeneralBook 类。首先要把光标放在类的头文件或者 m 文件的标注部位：

```

//
//  Book.h
//  SimpleObjcProto
//
//  Created by Marshal Wu on 11-5-18.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

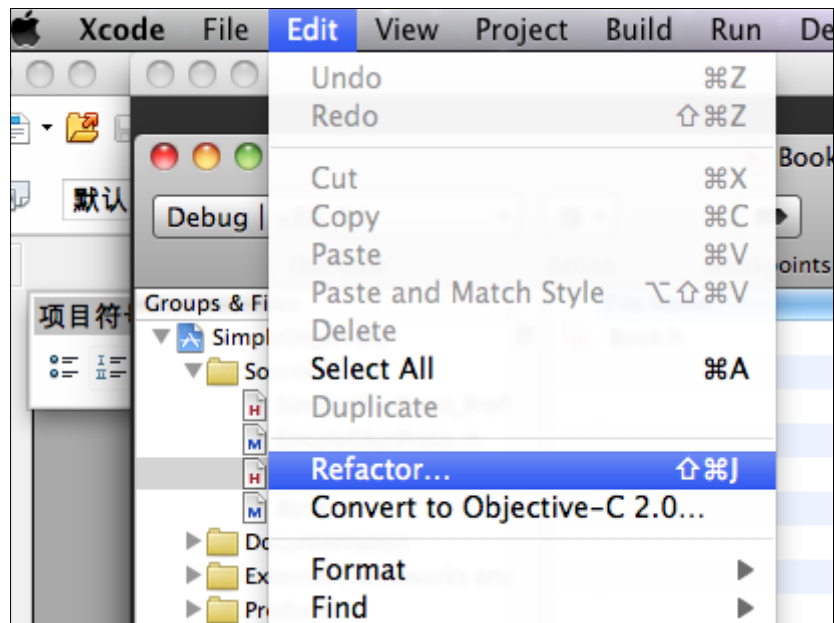
#import <Cocoa/Cocoa.h>

@interface Book : NSObject {
}

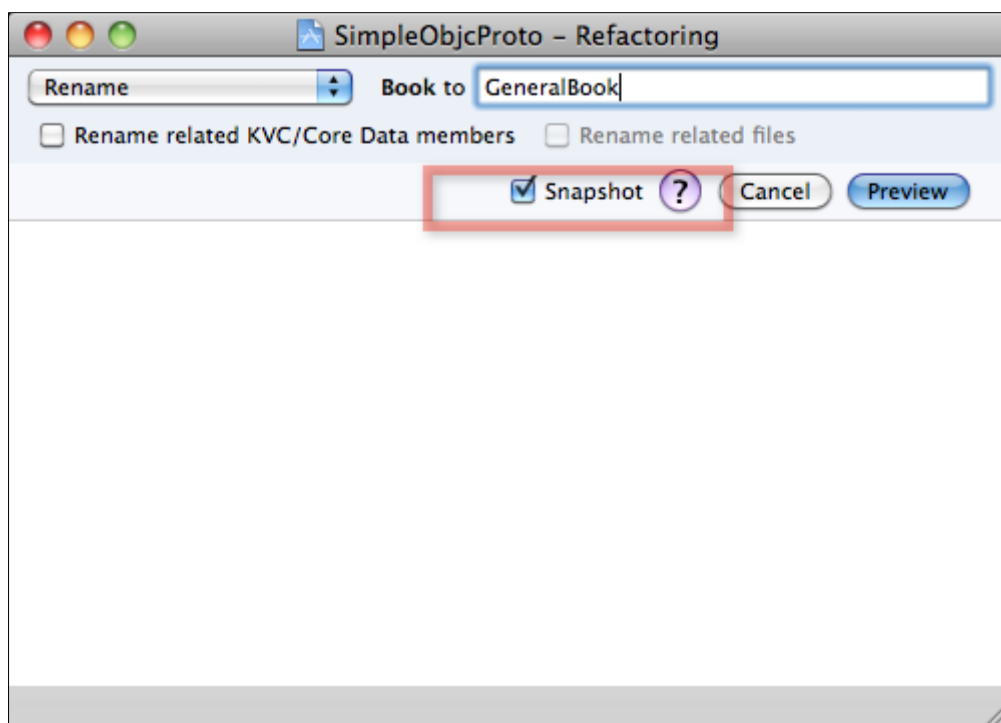
@end

```

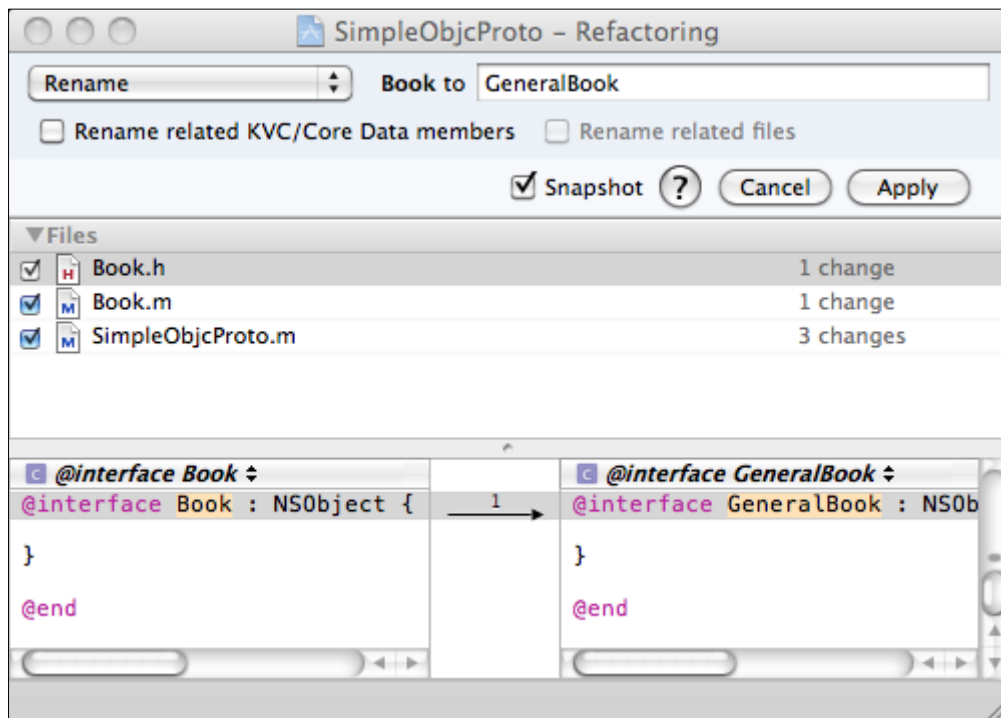
然后：



然后：



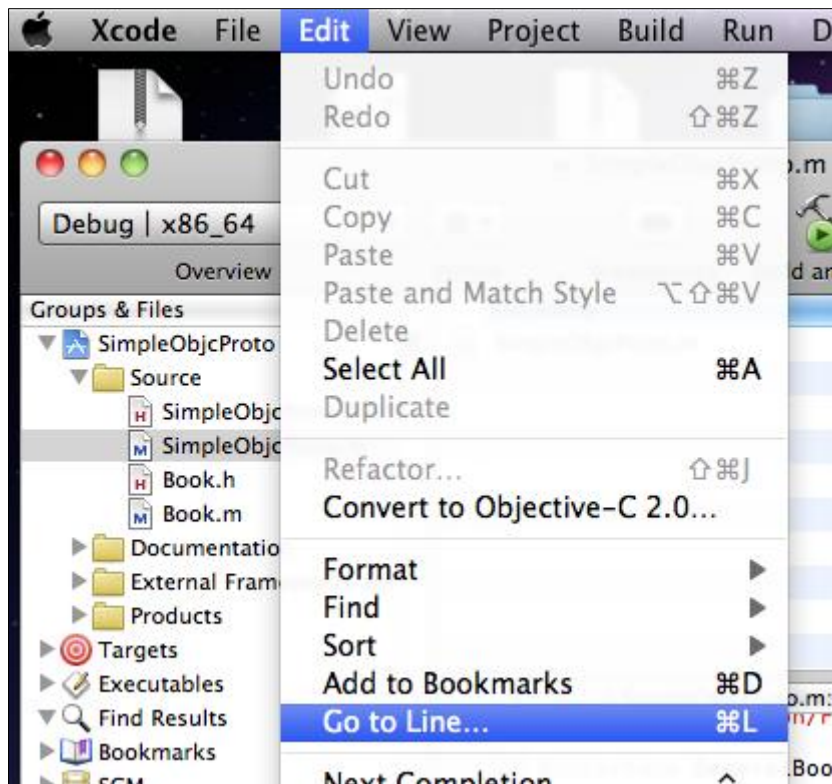
点击 preview，可以预览改动的内容：



一般要保持 snapshot 的勾选，这样重构操作会生成快照，便于重构错误后的恢复。
点击 apple，重构将执行。

11. 快速定位到代码行

可以：



定位光标到选中文件的行上。一般会用快捷键，command+L。使用菜单或者快捷键，都会出现下面的对话框，输入行号，回车，就会到该文件的指定行。

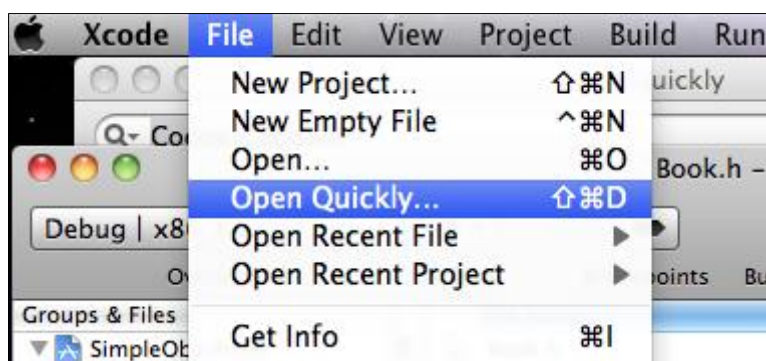


12. 快速打开文件

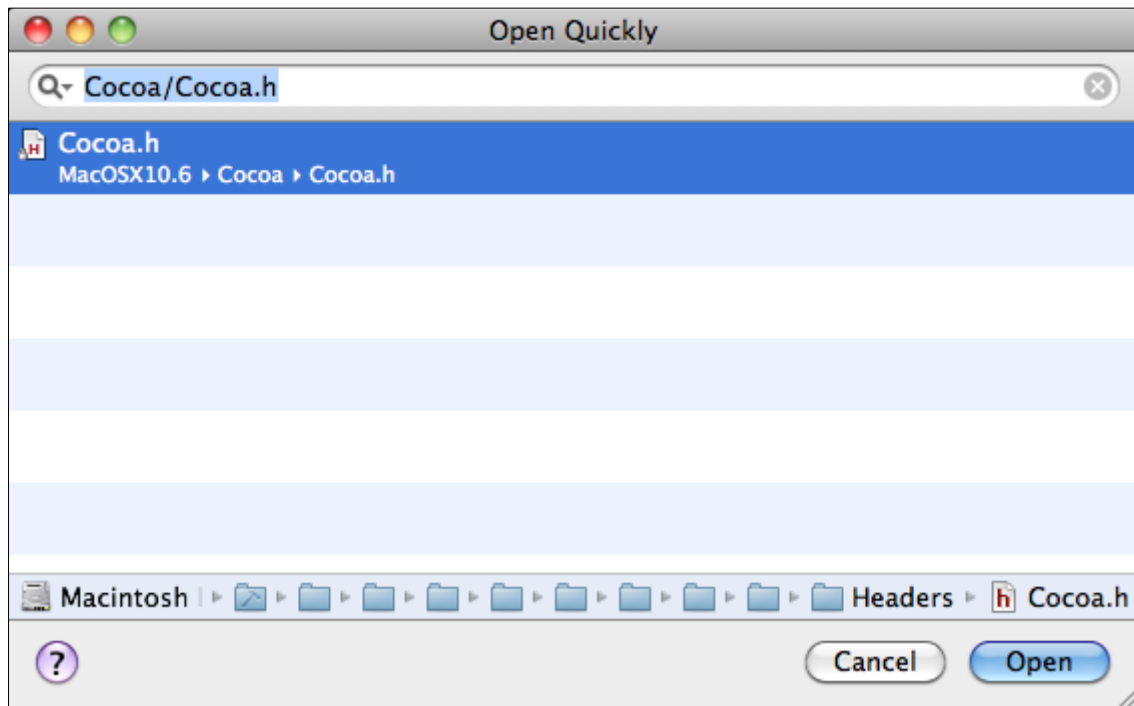
有时候，我们想快速打开头文件，比如：



这里的 Cocoa.h 到底是啥内容。可以鼠标选中 Cocoa.h，如上图。然后：



这时会弹出对话框：



双击 Cocoa.h 条目就可以看到了：



另外，还有个更方便的操作，针对头文件的，就是按住 command 键鼠标双击类型（函数、变量等）名称。会在源代码窗口中显示该类型的头文件。

13. 使用书签

以前在使用 Eclipse，我经常用到 TODO 标签功能，比如正在编写代码的时候需要做其他事情，或者提醒自己以后再实现的功能，就写个 TODO 注释，这样，可以在 Eclipse 的视图可以找到，方便以后找到这个代码并修改。

Xcode 中是否有相应的功能呢？我现在觉得书签功能可以做类似的事情。

比如我写了个代码：

```
#import "Book.h"

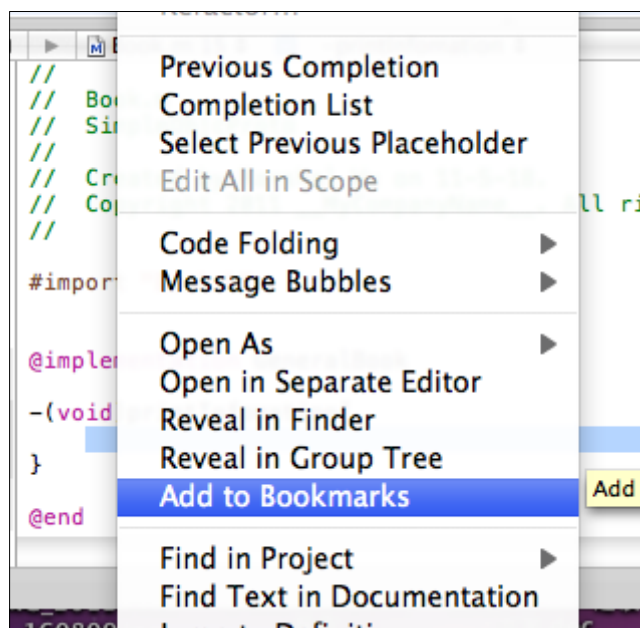
@implementation GeneralBook

-(void)printInfomation{

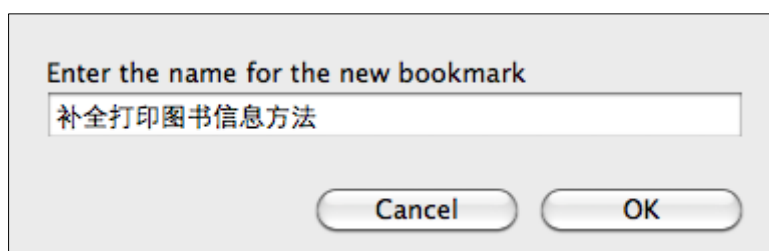
}

@end
```

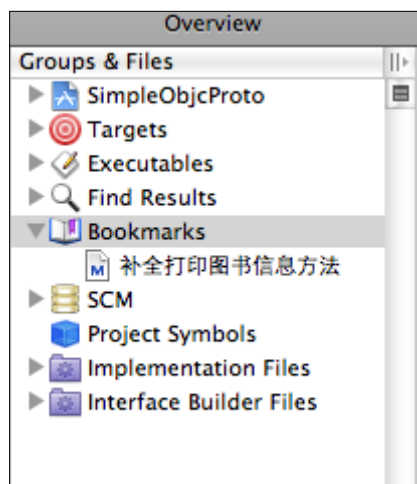
这个方法是空的，printInfomation。暂时不需要实现。但是要记下来，便于以后能找到并补充。那么让光标在方法内部。然后鼠标右键，Add to Bookmarks：



然后会弹出对话框，在里面填写标签的内容，比如：



这样，就可以在项目的书签节点找到这个条目了：



点击该条目，将回到刚才添加书签时光标的位置。

14. 自定义导航条

在代码窗口上边，有一个工具条。提供了很多方便的导航功能。比如：



也可以用来实现上面 TODO 的需求。这里有两种自定义导航条的写法。其中：

01. #pragma mark

是标准写法。而：

01. // TODO: xxx

02. // FIXME: xxx

是 Xcode 兼容的格式。

完整的代码：

```
#import "Book.h"

@implementation GeneralBook

-(id) init{
    return [super init];
}

#pragma mark 以下为必须实现的方法

// TODO: 在这里增加copy方法

-(void)printInfomation{
    // FIXME: bug #212
}

#pragma mark 以下为可选实现方法

-(NSString *)getName{
    return @"";
}

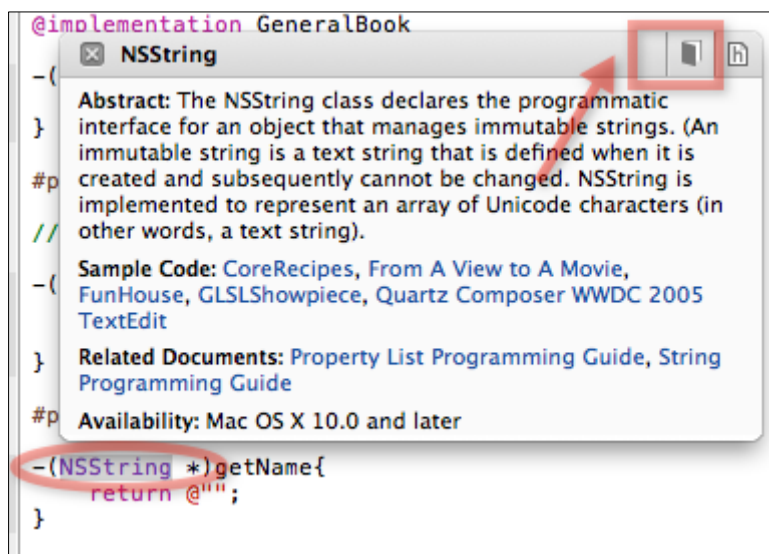
@end
```

产生了这样的导航条效果：

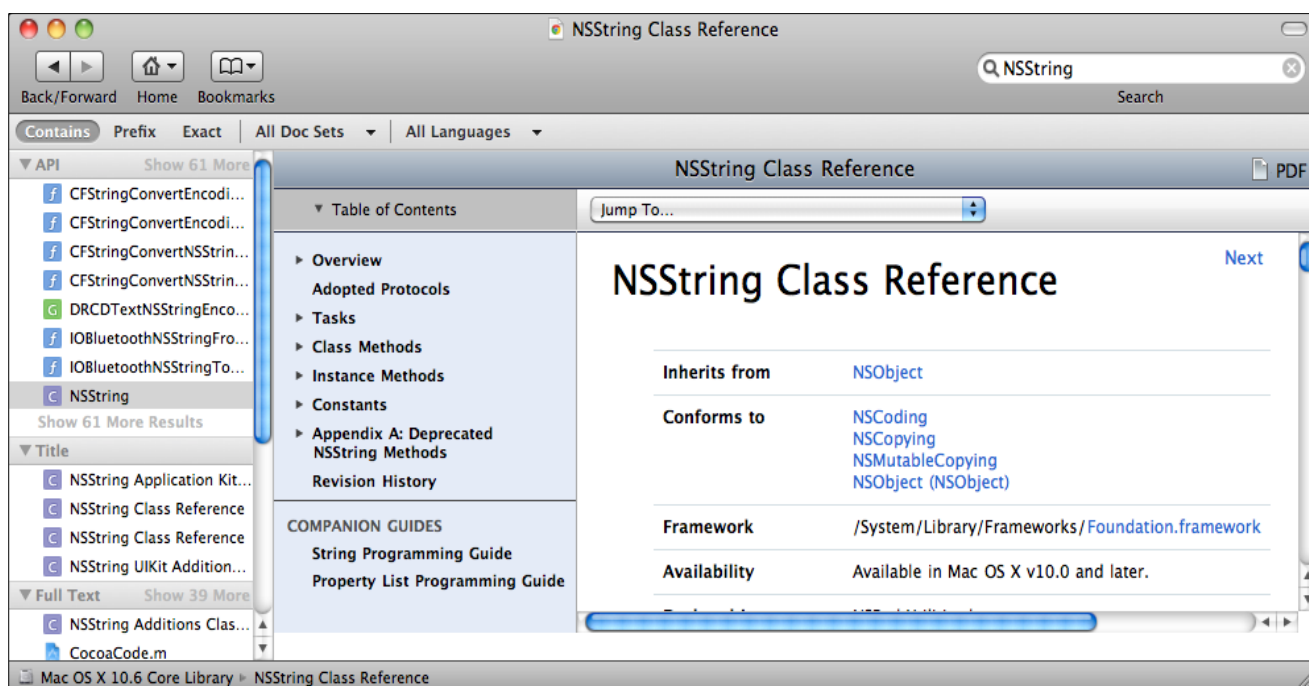


15.使用 Xcode 帮助

如果想快速的查看官方 API 文档，可在源代码中按下 option 键并鼠标双击该类型（函数、变量等），比如，下面是 NSString 的 API 文档对话框：



如果点击上面标识的按钮，则会弹出完整文档的窗口：

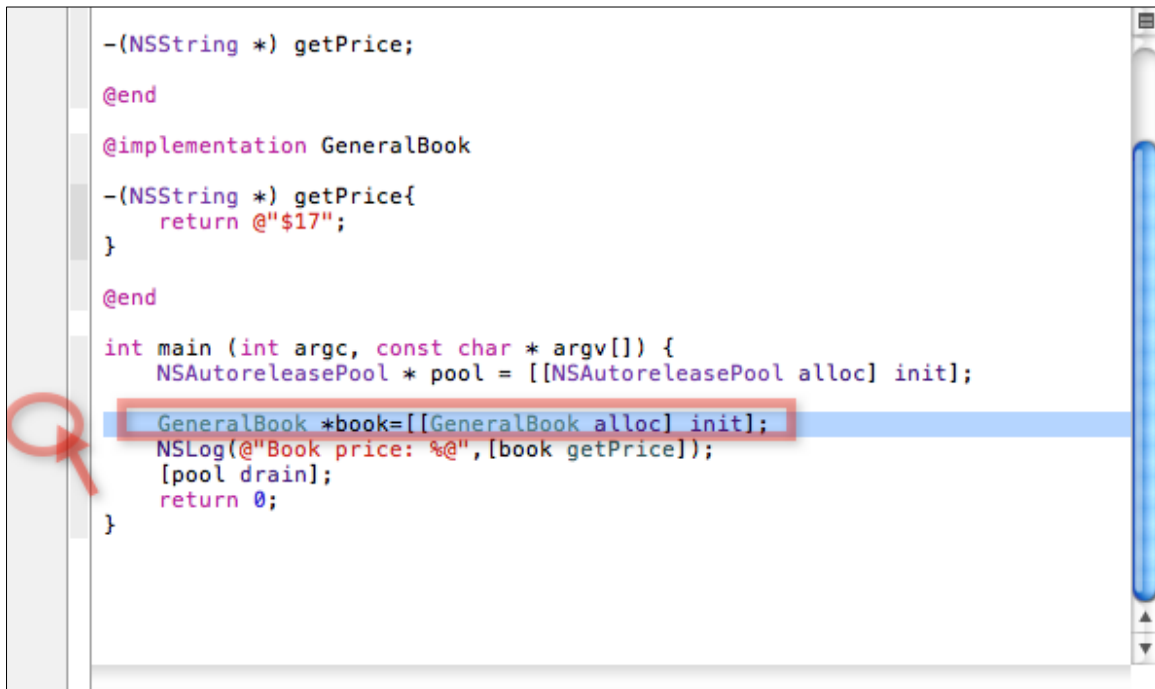


16. 调试代码

最朴素的调试方法，是通过 NSLog 打印出程序运行中的结果，然后根据这些结果判断程序运行的流程和结果值是否符合预期。对于简单的问题，使用这种方式也许就够用了。

但是，如果开发的是商业项目，它往往足够复杂，需要借助 Xcode 提供的专门调试工具。所有的编程工具的调试思路都是一样的。首先，你要在代码中设置断点。想象一下，程序的执行是顺序的，你可能怀疑某个地方的代码除了问题（引发 bug），那么就在这段代码开始的地方，比如是个方法的第一行，或者循环的开始部分，设置一个断点。那么程序在调试时会在运行到断点时中止，接下来，你可以一行一行的执行代码，判断执行顺序是否是自己预期的，或者变量的值是否和自己想的一样。

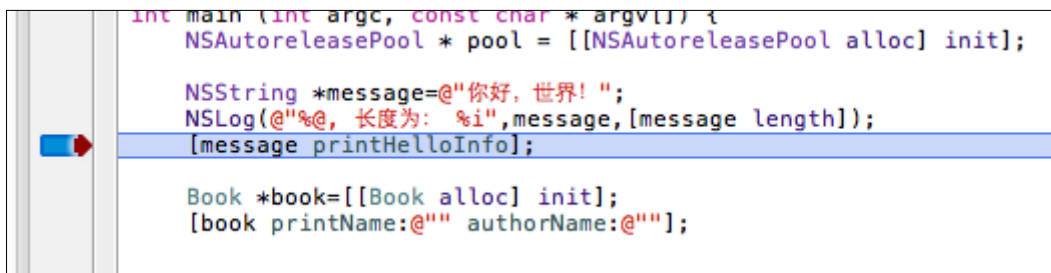
设置断点很简单，比如想对红框表示的行设置断点，就单击该行左侧红圈位置：



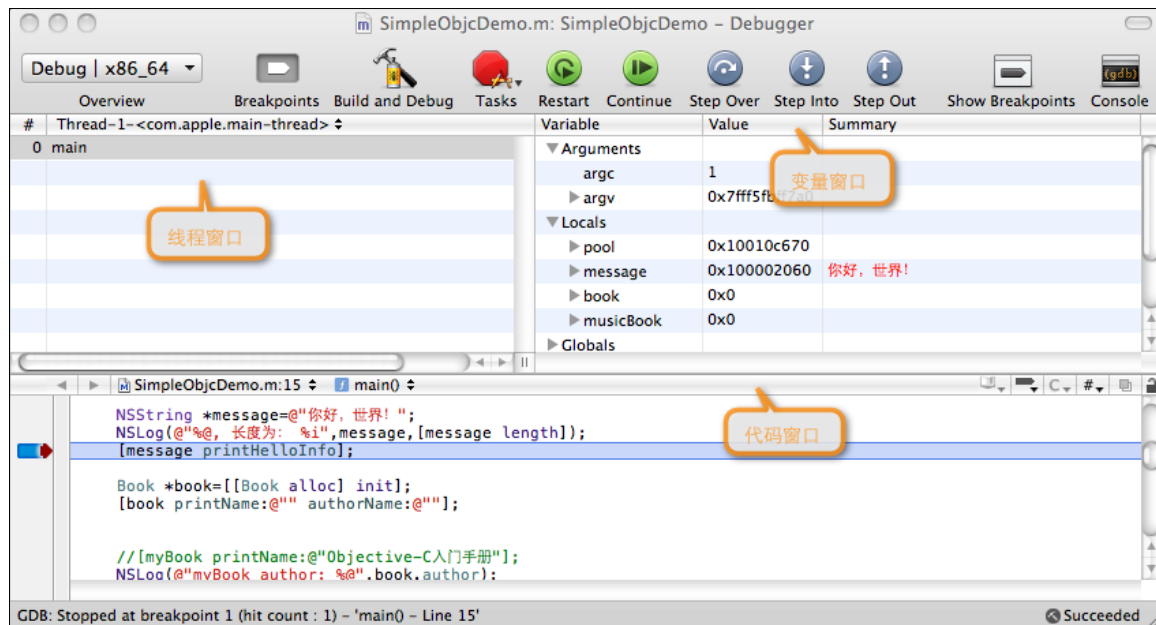
单击后会出现断点标志：



然后，运行代码，比如用快捷键：command+enter。这时将运行代码，并停止在断点处：



可通过 shift+command+y，调出调试对话框：



这和其他语言 IDE 工具的界面大同小异，因为都具有类似的功能。可通过：
continue，继续执行程序

step over, step into, step out，用于单步调试，分别表示：

step over：将执行当前方法内的下一个语句

step into：如果当前语句是方法调用，将单步执行当前语句调用方法内部第一行

step out：将跳出当前语句所在方法，到方法外的第一行

通过调试工具，可以对应用做全面和细致的调试。

原文地址：<http://www.61ic.com/Mobile/iPhone/201107/36218.html>