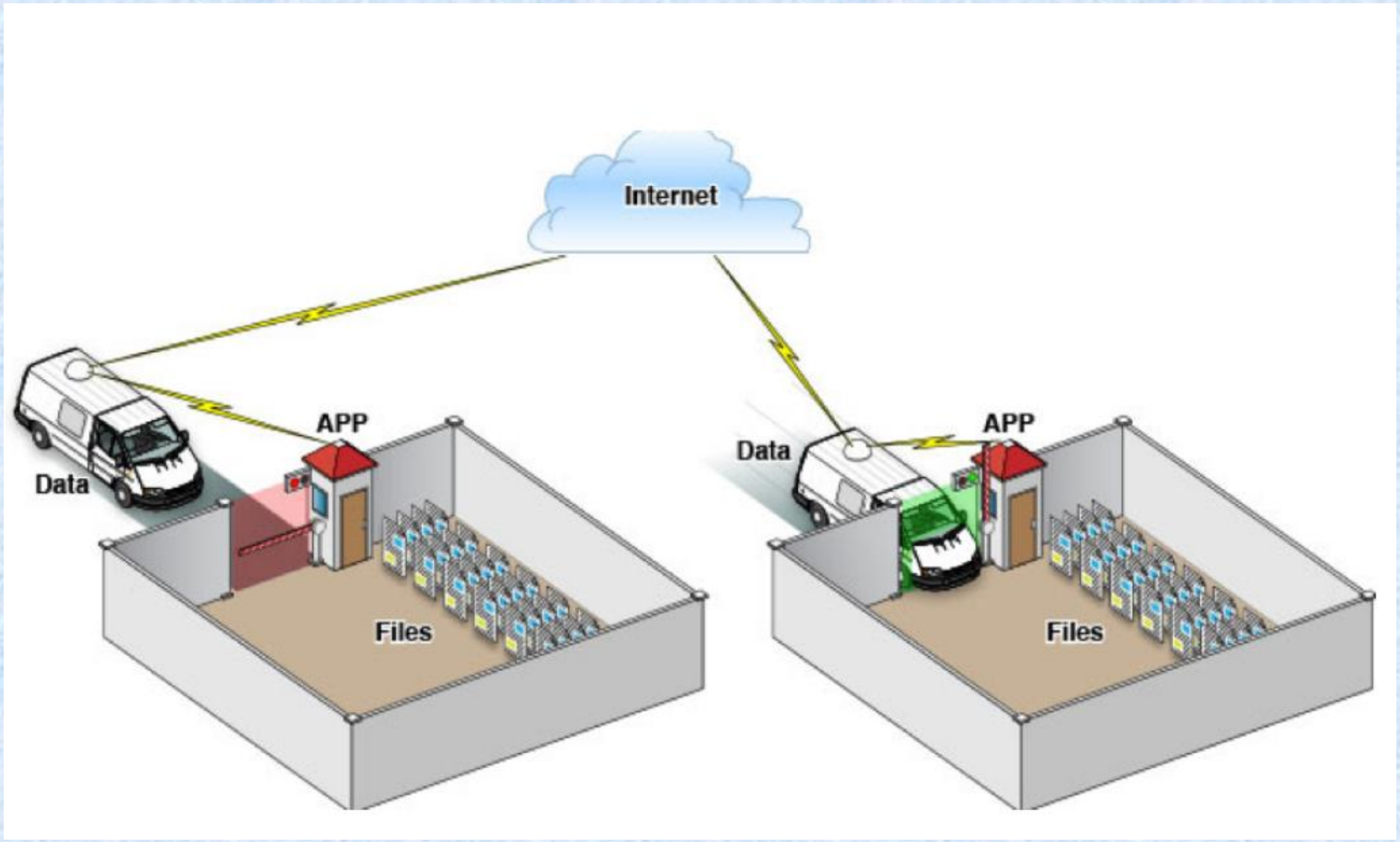
数据存取

一、IOS中SandBox

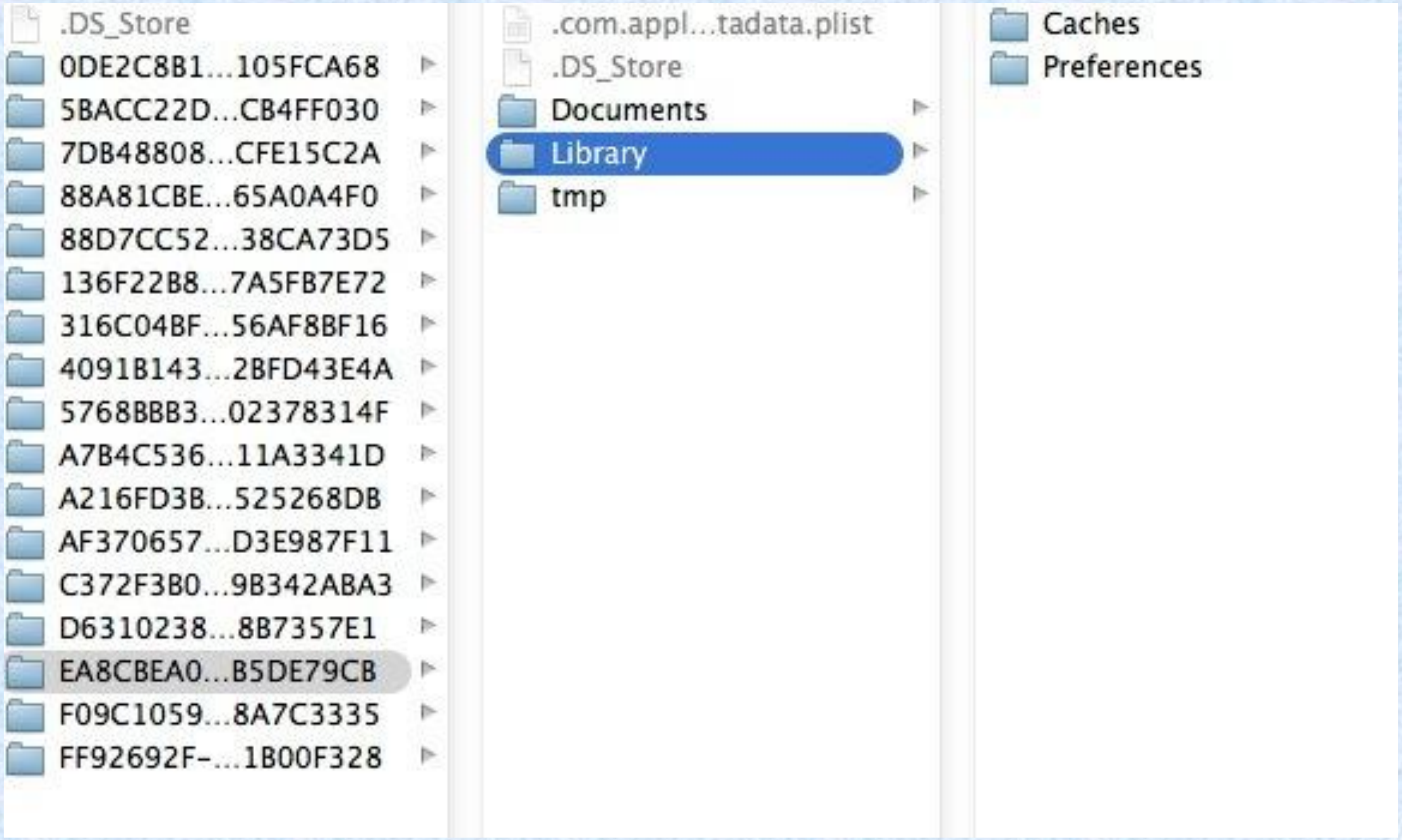
IOS应用程序只能在由自己的（系统分配）的目录空间中读取文件，不可以访问其他目录空间，此目录空间被成为沙盒，应用程序的所有非代码文件都保存在此空间中。。

- 每个应用程序都有自己的目录空间。
- 应用程序不能访问其它应用程序的目录空间。
- 应用程序对数据存取要满足一定的权限。


Internet
Data
APP
Files
Data
APP
Files

SandBox的目录结构

Documents:保存应用程序需要持久化的数据, ITunes同步设备时会备份该目录。
tmp:保存应用程序运行时所需的临时数据, 使用完毕后再将相应的文件从该目录删除。ITunes同步设备时不会备份该目录
Library/Caches:保存应用运行时生成的需要持久化的数据, iTunes同步设备时不会备份该目录。一般存储体积大、不需要备份的非重要数据
Library/Preference:保存应用的所有偏好设置,ITunes同步设备时会备份该目录。


.DS_Store
0DE2C8B1...105FCA68
5BACC22D...CB4FF030
7DB48808...CFE15C2A
88A81CBE...65A0A4F0
88D7CC52...38CA73D5
136F22B8...7A5FB7E72
316C04BF...56AF8BF16
4091B143...2BFD43E4A
5768BBB3...02378314F
A7B4C536...11A3341D
A216FD3B...525268DB
AF370657...D3E987F11
C372F3B0...9B342ABA3
D6310238...8B7357E1
EA8CBEA0...B5DE79CB
F09C1059...8A7C3335
FF92692F-...1B00F328

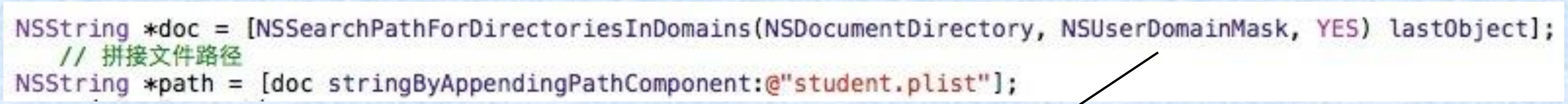
.com.appl...tadata.plist
.DS_Store
Documents
Library
tmp

Caches
Preferences

如何定位到应用程序的目录?

1、利用NSHomeDirectory() 来获得SandBox的根目录，然后通过拼接字符串的方式获得子目录。(不建议)

2、利用NSSearchPathForDirectoriesInDomains函数。



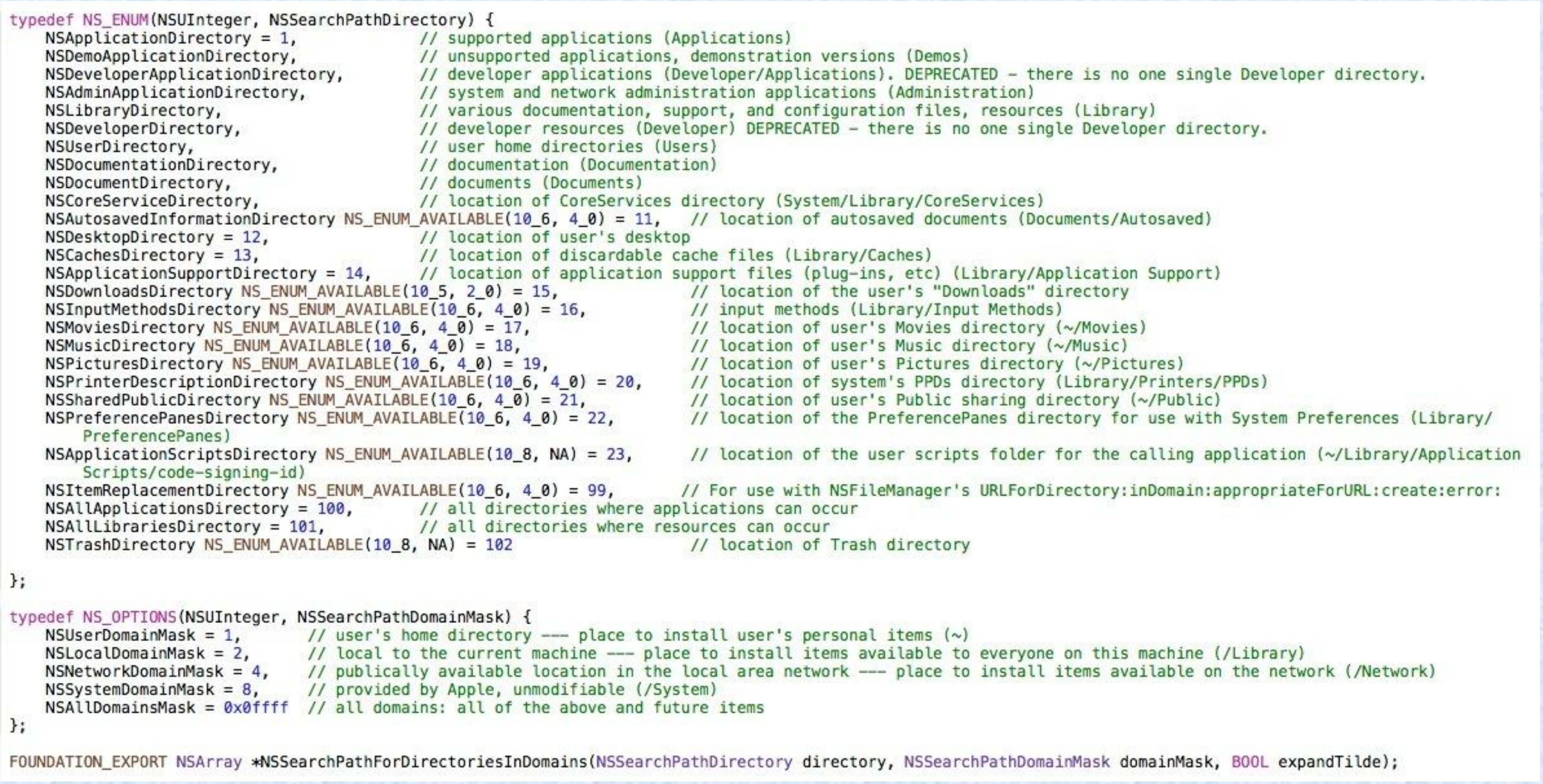
```
NSString *doc = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
   // 拼接文件路径
NSString *path = [doc stringByAppendingPathComponent:@"student.plist"];
```

改为NSCachesDirectory即可获得Caches目录

3、 tmp目录可以通过NSTemporaryDirectory() 获得。

NSString *tmpDir = NSTemporaryDirectory()

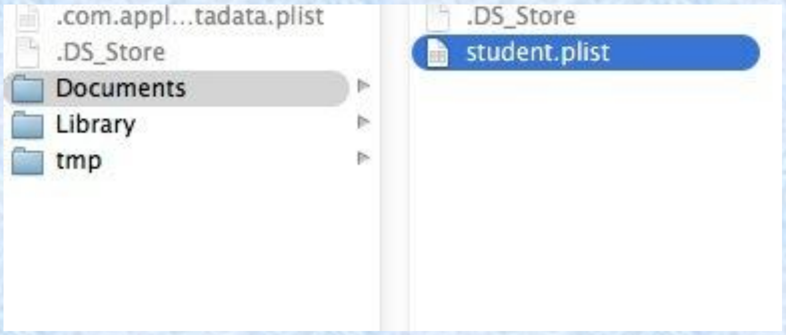
4、 Library/Preference:通过NSUserDefaults类存取该目录下的信息。

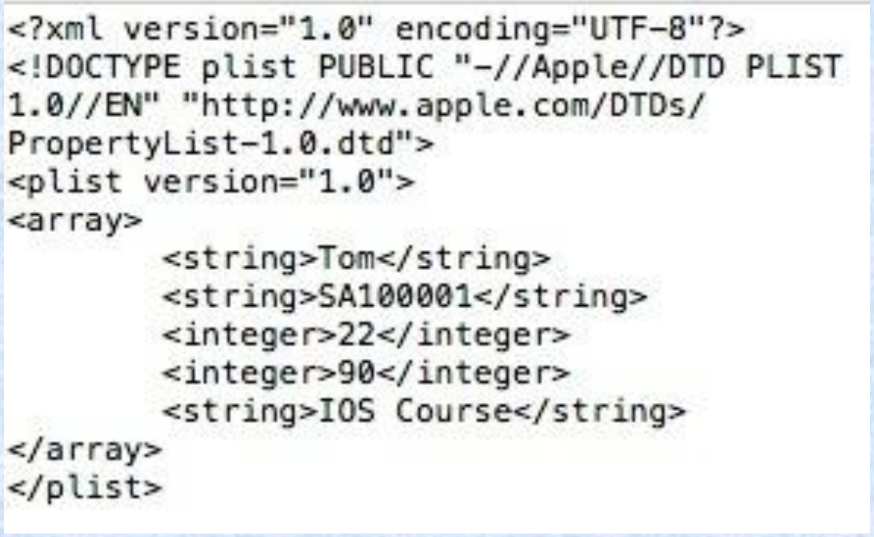


```
typedef NS_ENUM(NSUInteger, NSSearchPathDirectory) {
    NSApplicationDirectory = 1,             // supported applications (Applications)
    NSDemoApplicationDirectory,             // unsupported applications, demonstration versions (Demos)
    NSDeveloperApplicationDirectory,        // developer applications (Developer/Applications). DEPRECATED - there is no one single Developer directory.
    NSAdminApplicationDirectory,            // system and network administration applications (Administration)
    NSLibraryDirectory,                     // various documentation, support, and configuration files, resources (Library)
    NSDeveloperDirectory,                   // developer resources (Developer) DEPRECATED - there is no one single Developer directory.
    NSUserDirectory,                        // user home directories (Users)
    NSDocumentationDirectory,               // documentation (Documentation)
    NSDocumentDirectory,                    // documents (Documents)
    NSCoreServiceDirectory,                 // location of CoreServices directory (System/Library/CoreServices)
    NSAutosavedInformationDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 11,   // location of autosaved documents (Documents/Autosaved)
    NSDesktopDirectory = 12,                // location of user's desktop
    NSCachesDirectory = 13,                 // location of discardable cache files (Library/Caches)
    NSApplicationSupportDirectory = 14,     // location of application support files (plug-ins, etc) (Library/Application Support)
    NSDownloadsDirectory NS_ENUM_AVAILABLE(10_5, 2_0) = 15,              // location of the user's "Downloads" directory
    NSInputMethodsDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 16,           // input methods (Library/Input Methods)
    NSMoviesDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 17,                 // location of user's Movies directory (~/Movies)
    NSMusicDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 18,                  // location of user's Music directory (~/Music)
    NSPicturesDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 19,               // location of user's Pictures directory (~/Pictures)
    NSPrinterDescriptionDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 20,     // location of system's PPDs directory (Library/Printers/PPDs)
    NSSharedPublicDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 21,           // location of user's Public sharing directory (~/Public)
    NSPreferencePanesDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 22,        // location of the PreferencePanes directory for use with System Preferences (Library/
        PreferencePanes)
    NSApplicationScriptsDirectory NS_ENUM_AVAILABLE(10_8, NA) = 23,      // location of the user scripts folder for the calling application (~/Library/Application
        Scripts/code-signing-id)
    NSItemReplacementDirectory NS_ENUM_AVAILABLE(10_6, 4_0) = 99,       // For use with NSFileManager's URLForDirectory:inDomain:appropriateForURL:create:error:
    NSAllApplicationsDirectory = 100,       // all directories where applications can occur
    NSAllLibrariesDirectory = 101,          // all directories where resources can occur
    NSTrashDirectory NS_ENUM_AVAILABLE(10_8, NA) = 102                   // location of Trash directory

};

typedef NS_OPTIONS(NSUInteger, NSSearchPathDomainMask) {
    NSUserDomainMask = 1,       // user's home directory --- place to install user's personal items (~)
    NSLocalDomainMask = 2,      // local to the current machine --- place to install items available to everyone on this machine (/Library)
    NSNetworkDomainMask = 4,    // publically available location in the local area network --- place to install items available on the network (/Network)
    NSSystemDomainMask = 8,     // provided by Apple, unmodifiable (/System)
    NSAllDomainsMask = 0x0ffff  // all domains: all of the above and future items
};

FOUNDATION_EXPORT NSArray *NSSearchPathForDirectoriesInDomains(NSSearchPathDirectory directory, NSSearchPathDomainMask domainMask, BOOL expandTilde);
```

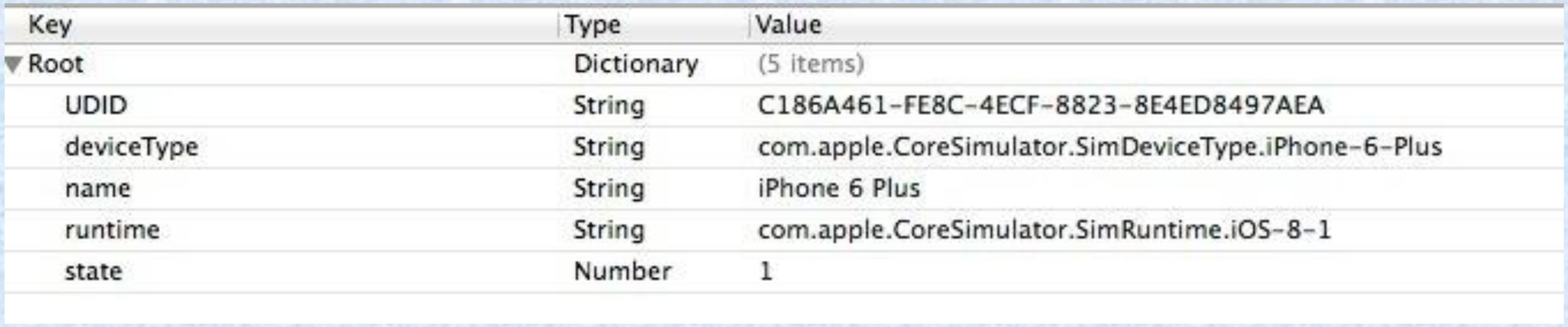
二、属性列表(.plist)

在IOS中属性列表提供了一个方便的方式来存储*简单的结构化数据*。它是以XML格式存储的。属性列表文件的后缀名为.plist。





```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<array>
        <string>Tom</string>
        <string>SA100001</string>
        <integer>22</integer>
        <integer>90</integer>
        <string>IOS Course</string>
</array>
</plist>
```

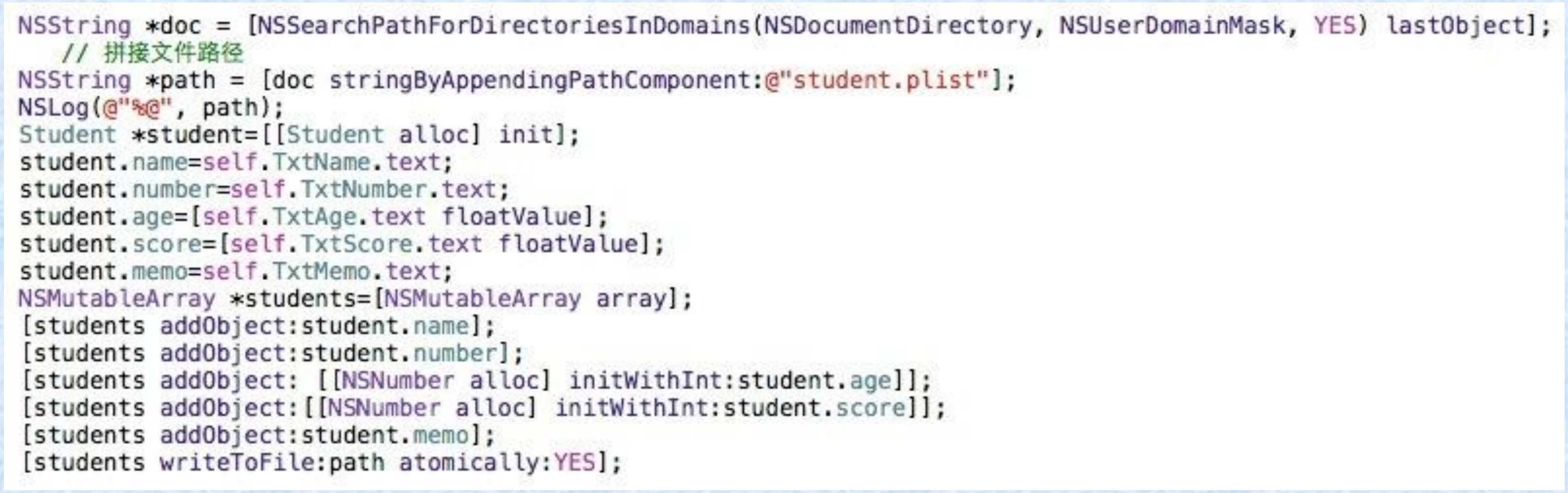


Key	Type	Value
▼ Root	Dictionary	(5 items)
UDID	String	C186A461-FE8C-4ECF-8823-8E4ED8497AEA
deviceType	String	com.apple.CoreSimulator.SimDeviceType.iPhone-6-Plus
name	String	iPhone 6 Plus
runtime	String	com.apple.CoreSimulator.SimRuntime.iOS-8-1
state	Number	1

如何使用属性列表？

●注意并不是所有的数据都能保存在属性列表中。

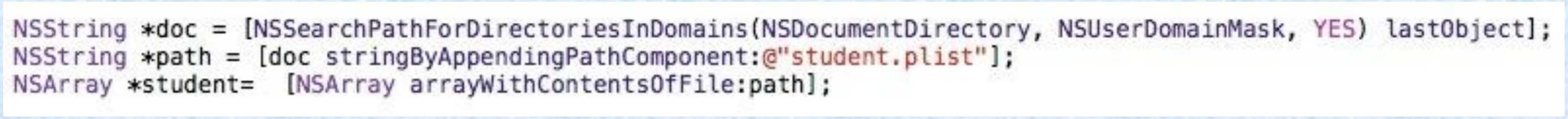
●如果对象是NSString、NSDictionary、NSArray、NSData、 NSNumber等类型(writeToFile:atomically:方法)那么它可以被保存到属性列表中。



```
NSString *doc = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
   // 拼接文件路径
NSString *path = [doc stringByAppendingPathComponent:@"student.plist"];
NSLog(@"%@", path);
Student *student=[[Student alloc] init];
student.name=self.TxtName.text;
student.number=self.TxtNumber.text;
student.age=[self.TxtAge.text floatValue];
student.score=[self.TxtScore.text floatValue];
student.memo=self.TxtMemo.text;
NSMutableArray *students=[NSMutableArray array];
[students addObject:student.name];
[students addObject:student.number];
[students addObject: [[NSNumber alloc] initWithInt:student.age]];
[students addObject:[[NSNumber alloc] initWithInt:student.score]];
[students addObject:student.memo];
[students writeToFile:path atomically:YES];
```

读取属性列表

利用上述对象中的xxxContentsOfFiles方法。



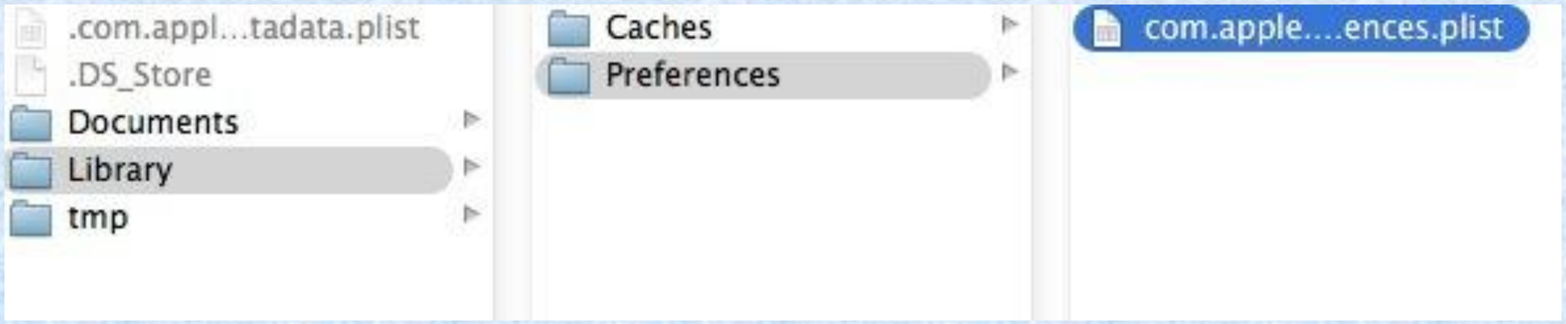
```
NSString *doc = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
NSString *path = [doc stringByAppendingPathComponent:@"student.plist"];
NSArray *student=  [NSArray arrayWithContentsOfFile:path];
```

问题：可否再把数组中的自定义对象保存到属性列表文件中？

三、偏好设置

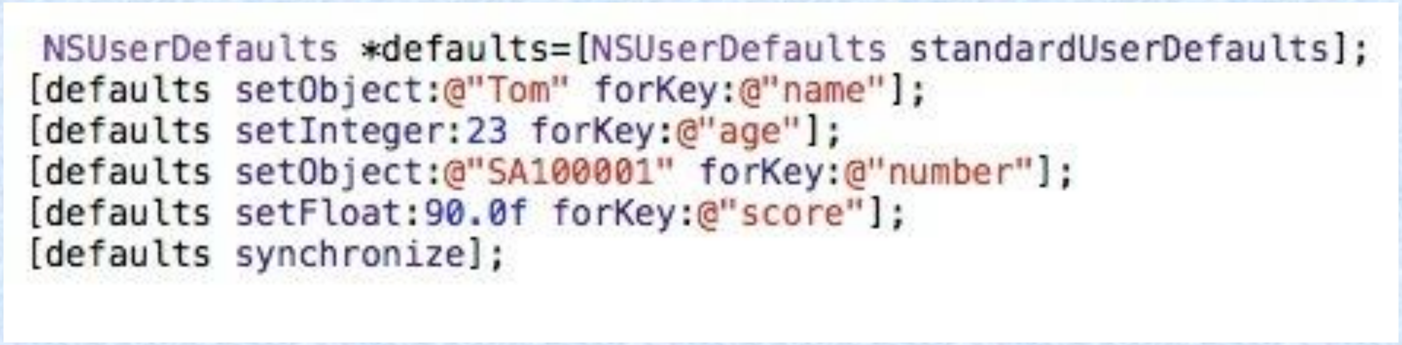
1、什么是偏好设置

偏好设置是面向应用程序的特定的设置，用于配置应用程序的行为和外观。IOS提供了NSUserDefaults来存取偏好设置。

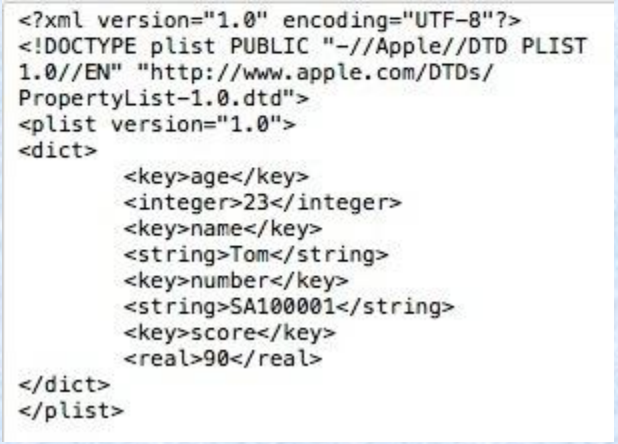


注意：它也是一个.plist文件。是保存在preferences目录下。使用NSUserDefaults来存取。

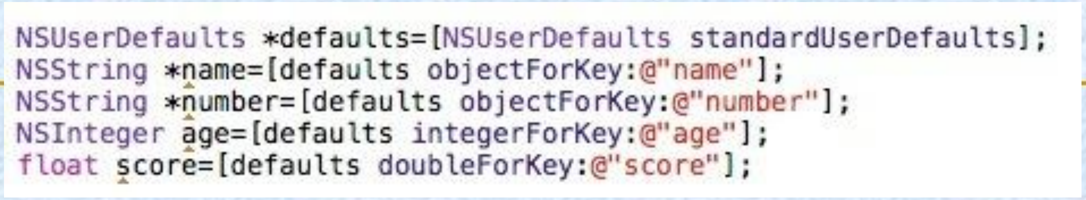
NSUserDefaults存取的例子：



```objc
 NSUserDefaults *defaults=[NSUserDefaults standardUserDefaults];
[defaults setObject:@"Tom" forKey:@"name"];
[defaults setInteger:23 forKey:@"age"];
[defaults setObject:@"SA100001" forKey:@"number"];
[defaults setFloat:90.0f forKey:@"score"];
[defaults synchronize];
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST
1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>age</key>
        <integer>23</integer>
        <key>name</key>
        <string>Tom</string>
        <key>number</key>
        <string>SA100001</string>
        <key>score</key>
        <real>90</real>
</dict>
</plist>
```



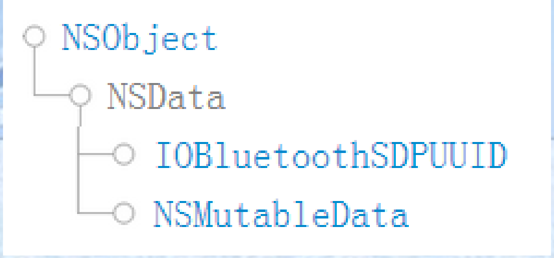
```objc
NSUserDefaults *defaults=[NSUserDefaults standardUserDefaults];
NSString *name=[defaults objectForKey:@"name"];
NSString *number=[defaults objectForKey:@"number"];
NSInteger age=[defaults integerForKey:@"age"];
float score=[defaults doubleForKey:@"score"];
```

注意：

1、偏好设置是专门用来保存应用程序的配置信息的，一般情况不要在偏好设置中保存其他数据。偏好设置会将所有的数据都保存到同一个文件中。

2、偏好设置对数据进行保存的时间是不确定的，如果要立即存储数据可使用[defaults synchronize];

3、除对象外，还可以存取基本的数据类型。



四、归档与解档（对象的编码和解码）

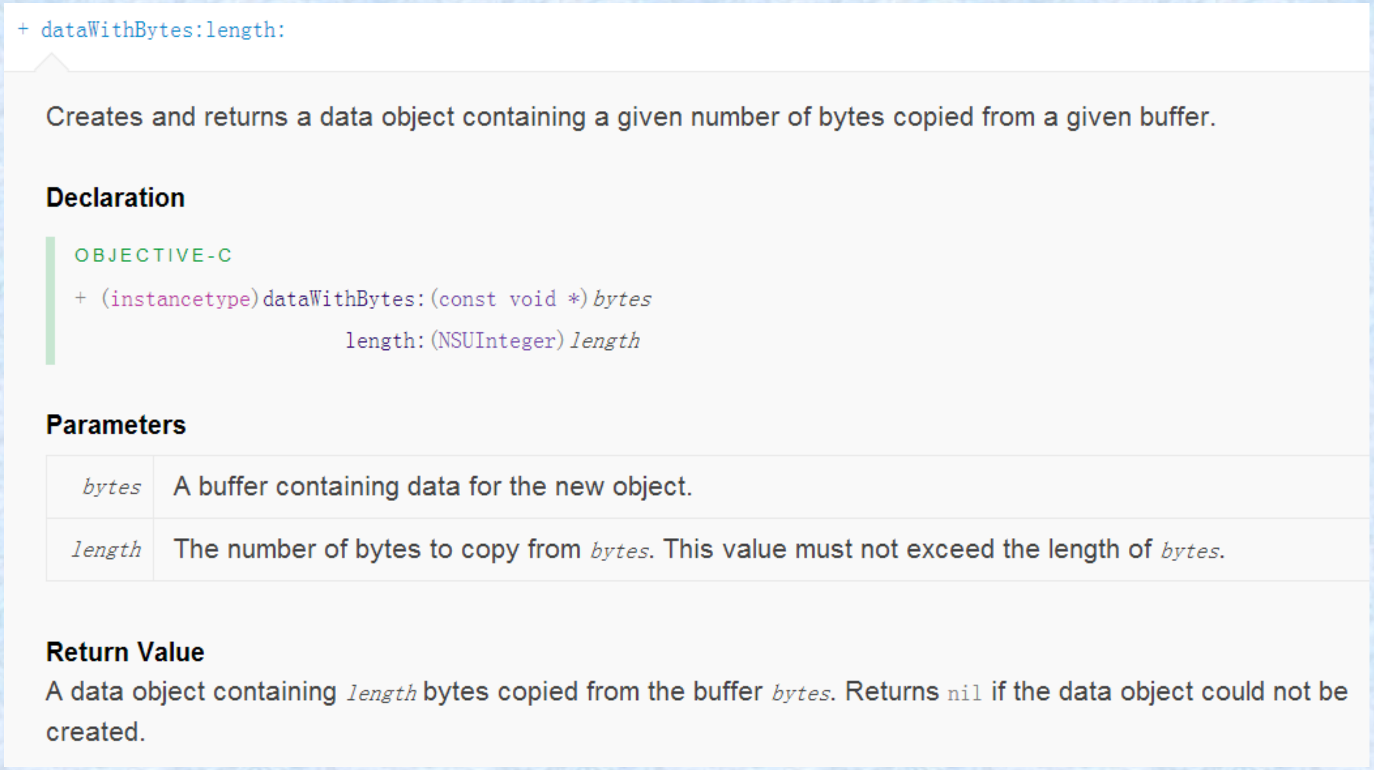
1、NSData

NSData and its mutable subclass NSMutableData provide data objects, object-oriented wrappers for byte buffers. Data objects let simple allocated buffers (that is, data with no embedded pointers) take on the behavior of Foundation objects.

NSData creates static data objects, and SMutableData creates dynamic data objects. NSData and NSMutableData are typically used for data storage and are also useful in Distributed Objects applications, where data contained in data objects can be copied or moved between applications.

NSData类型可用来实现缓冲区，用于存储二进制的数据类型。例如：图像、视频等。(参考JAVA中的字节数组）

创建与初始化:


+ dataWithBytes:length:

Creates and returns a data object containing a given number of bytes copied from a given buffer.

Declaration

OBJECTIVE-C

```
+ (instancetype)dataWithBytes:(const void *)bytes
                       length:(NSUInteger)length
```

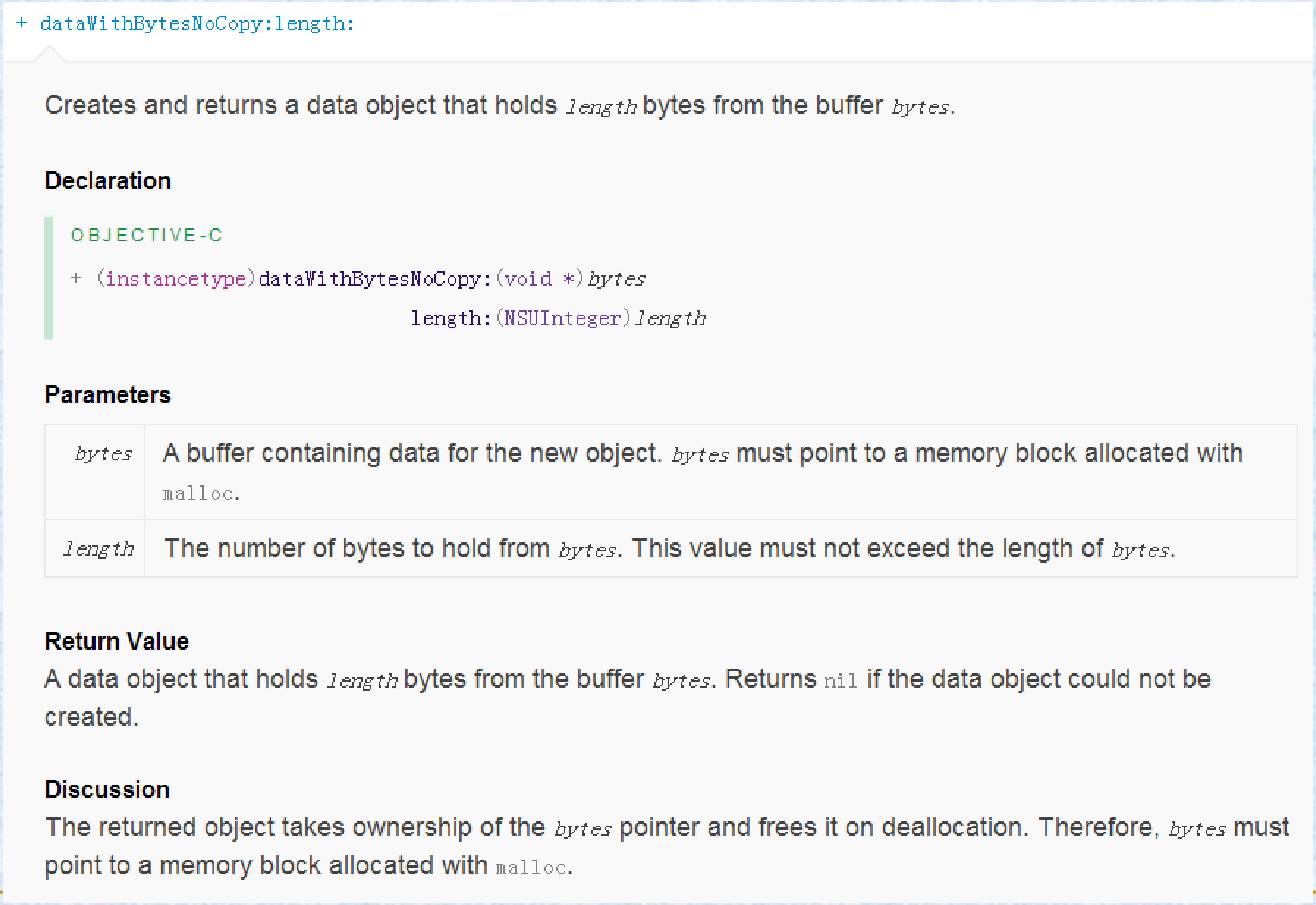
Parameters

bytes	A buffer containing data for the new object.
length	The number of bytes to copy from *bytes*. This value must not exceed the length of *bytes*.

Return Value

A data object containing *length* bytes copied from the buffer *bytes*. Returns `nil` if the data object could not be created.

```
const char *str="Hello world";
NSData *data=[NSData dataWithBytes:str length:strlen(str)+1];
```



+ dataWithBytesNoCopy:length:

Creates and returns a data object that holds *length* bytes from the buffer *bytes*.

Declaration

OBJECTIVE-C

```
+ (instancetype)dataWithBytesNoCopy:(void *)bytes
                             length:(NSUInteger)length
```

Parameters

bytes	A buffer containing data for the new object. *bytes* must point to a memory block allocated with `malloc`.
length	The number of bytes to hold from *bytes*. This value must not exceed the length of *bytes*.

Return Value

A data object that holds *length* bytes from the buffer *bytes*. Returns `nil` if the data object could not be created.

Discussion

The returned object takes ownership of the *bytes* pointer and frees it on deallocation. Therefore, *bytes* must point to a memory block allocated with `malloc`.

NSData可以与NSString, NSArray, NSDictionary, NSDate, UIImage等对象之间进行转化。

Eg: NSData转NSString

```
    NSString *str = [[NSString alloc] initWithData:data
                        encoding:NSUTF8StringEncoding];
```

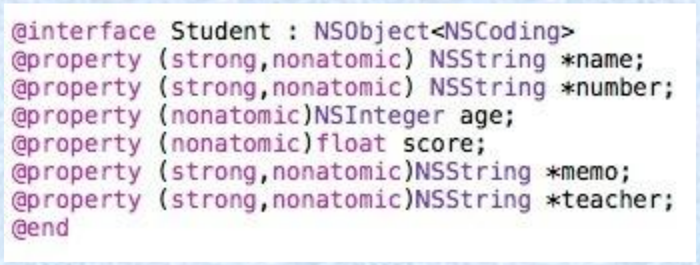
NSString转NSData

```
    NSString *str = @"Hello world";
    NSData *data
            =[str dataUsingEncoding:NSUTF8StringEncoding]
```

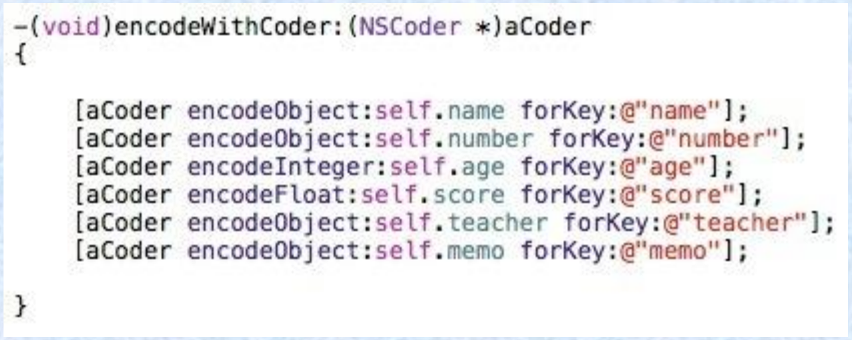
2、对自定义对象进行编码和解码

　　如果要对自定义对象进行解码和编码，必须要实现NSCoding协议中的encodeWithCoder方法和initWithCoder方法。

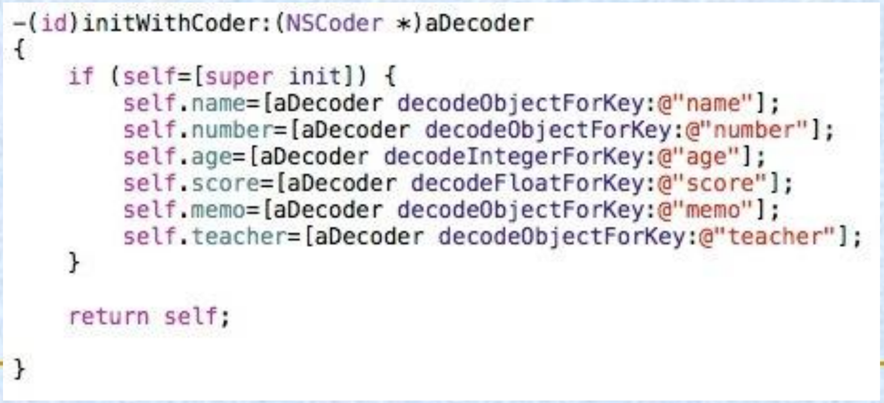
对象Student



```
@interface Student : NSObject<NSCoding>
@property (strong,nonatomic) NSString *name;
@property (strong,nonatomic) NSString *number;
@property (nonatomic)NSInteger age;
@property (nonatomic)float score;
@property (strong,nonatomic)NSString *memo;
@property (strong,nonatomic)NSString *teacher;
@end
```

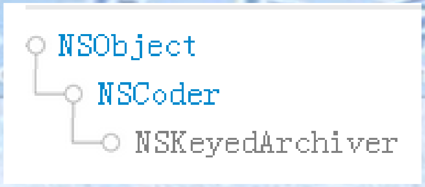


```
-(void)encodeWithCoder:(NSCoder *)aCoder
{

    [aCoder encodeObject:self.name forKey:@"name"];
    [aCoder encodeObject:self.number forKey:@"number"];
    [aCoder encodeInteger:self.age forKey:@"age"];
    [aCoder encodeFloat:self.score forKey:@"score"];
    [aCoder encodeObject:self.teacher forKey:@"teacher"];
    [aCoder encodeObject:self.memo forKey:@"memo"];

}
```



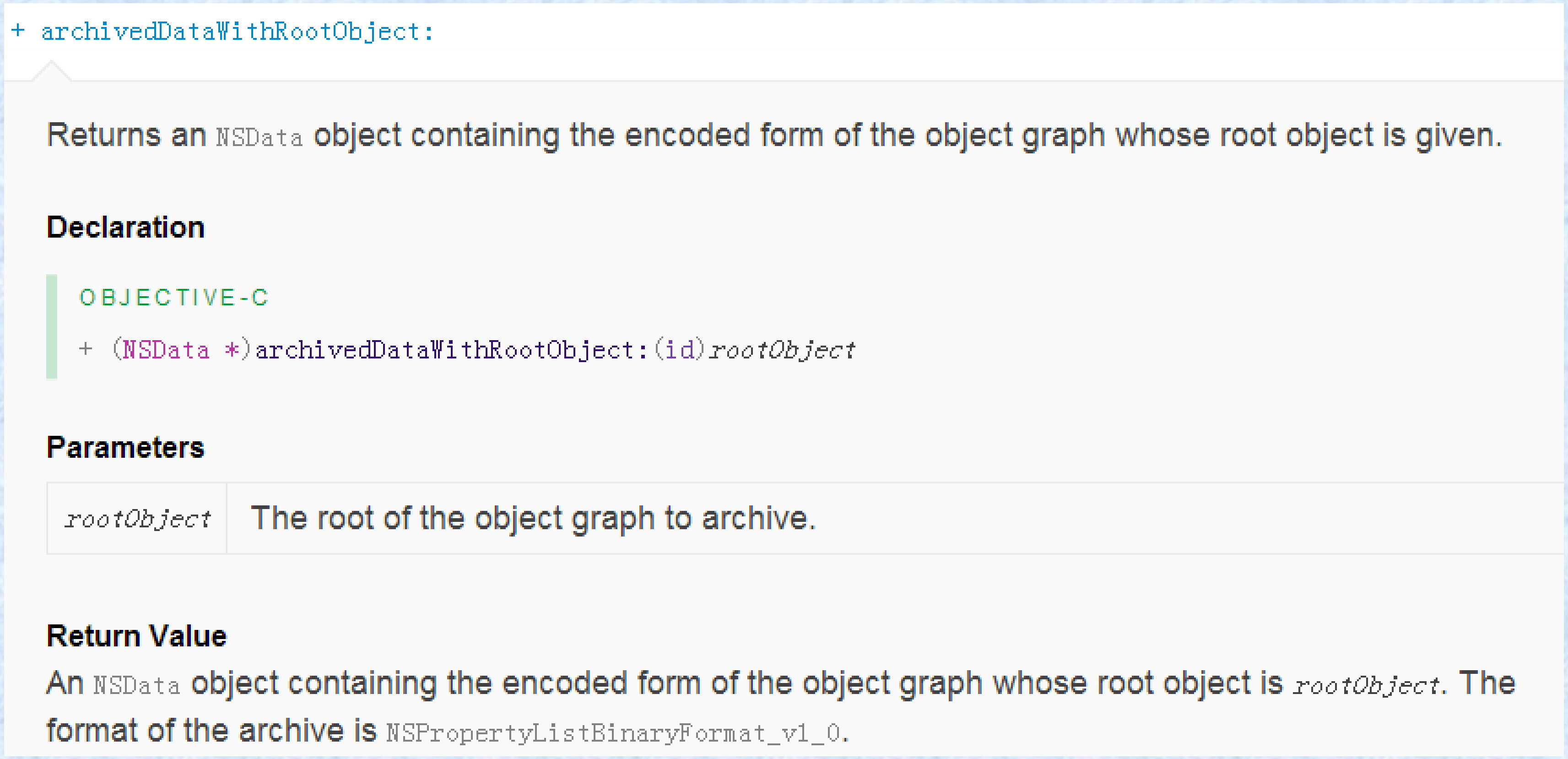
```
-(id)initWithCoder:(NSCoder *)aDecoder
{
    if (self=[super init]) {
        self.name=[aDecoder decodeObjectForKey:@"name"];
        self.number=[aDecoder decodeObjectForKey:@"number"];
        self.age=[aDecoder decodeIntegerForKey:@"age"];
        self.score=[aDecoder decodeFloatForKey:@"score"];
        self.memo=[aDecoder decodeObjectForKey:@"memo"];
        self.teacher=[aDecoder decodeObjectForKey:@"teacher"];
    }

    return self;

}
```

3、归档NSKeyedArchiver



NSKeyedArchiver, a concrete subclass of NSCoder, provides a way to encode objects (and scalar values) into an architecture-independent format that can be stored in a file. When you archive a set of objects, the class information and instance variables for each object are written to the archive. NSKeyedArchiver’s companion class, NSKeyedUnarchiver, decodes the data in an archive and creates a set of objects equivalent to the original set.

归档至NSData



`+ archivedDataWithRootObject:`

Returns an `NSData` object containing the encoded form of the object graph whose root object is given.

Declaration

OBJECTIVE-C

```
+ (NSData *)archivedDataWithRootObject:(id)rootObject
```

Parameters

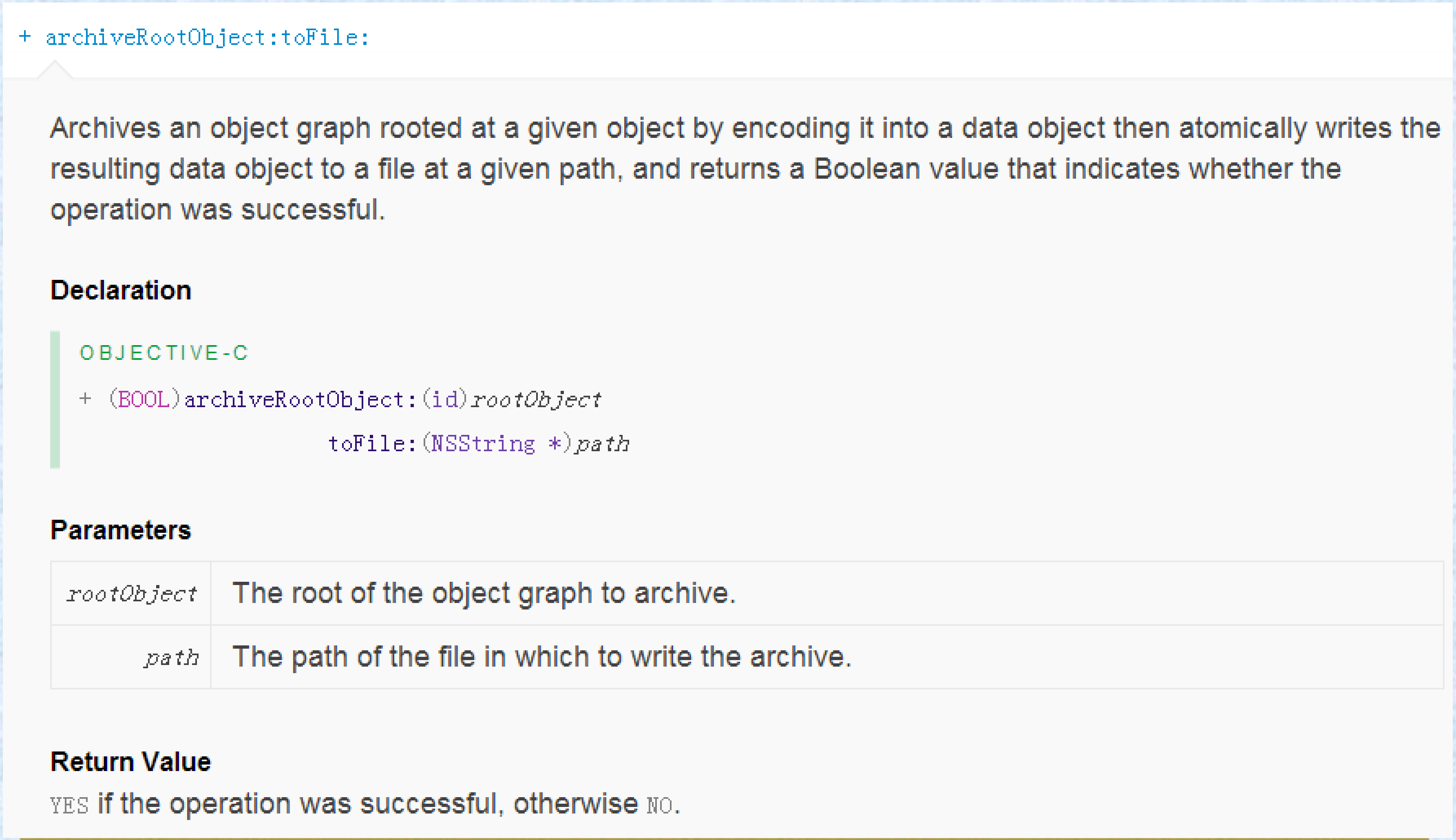
rootObject	The root of the object graph to archive.

Return Value

An `NSData` object containing the encoded form of the object graph whose root object is *rootObject*. The format of the archive is `NSPropertyListBinaryFormat_v1_0`.

```
NSData *data=[NSKeyedArchiver  archivedDataWithRootObject:student];
```

归档至文件



+ archiveRootObject:toFile:

Archives an object graph rooted at a given object by encoding it into a data object then atomically writes the resulting data object to a file at a given path, and returns a Boolean value that indicates whether the operation was successful.

Declaration

OBJECTIVE-C

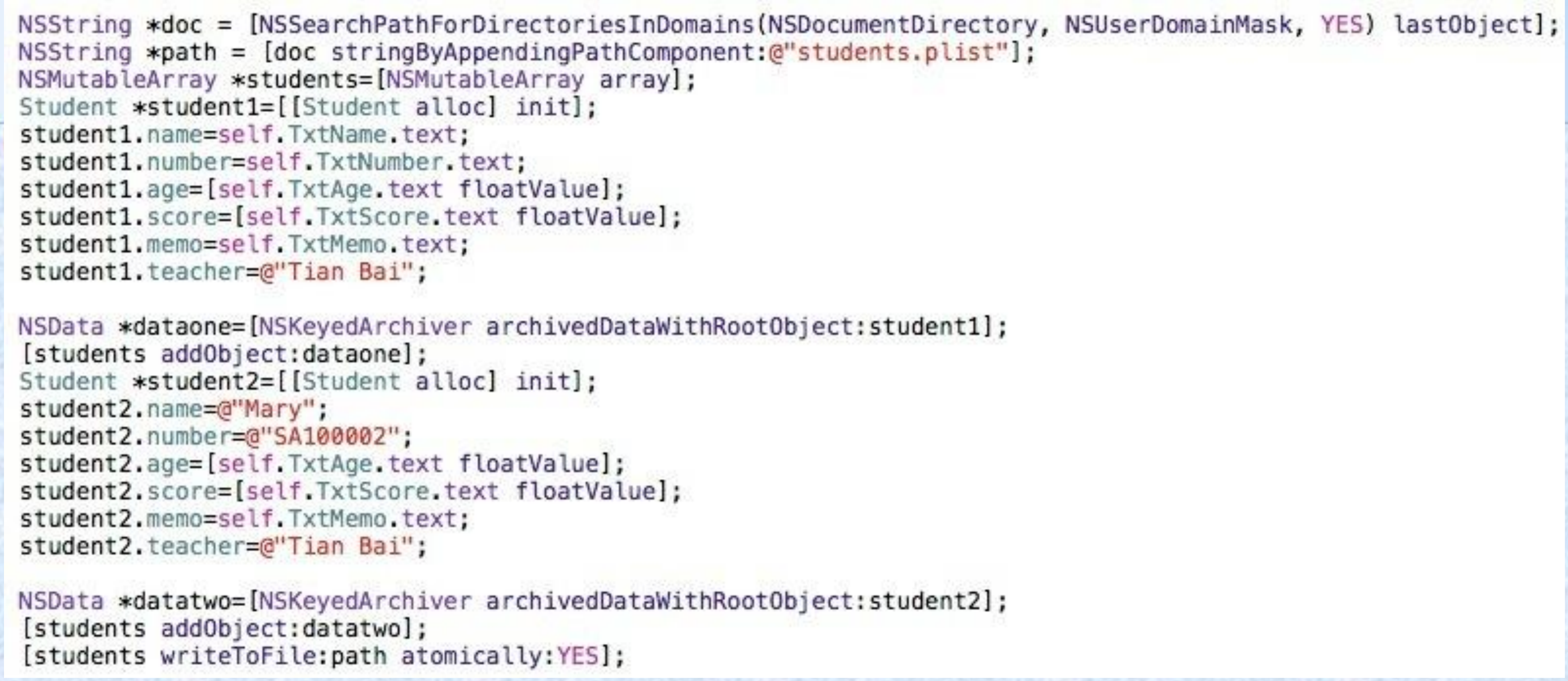
```
+ (BOOL)archiveRootObject:(id)rootObject
                   toFile:(NSString *)path
```

Parameters

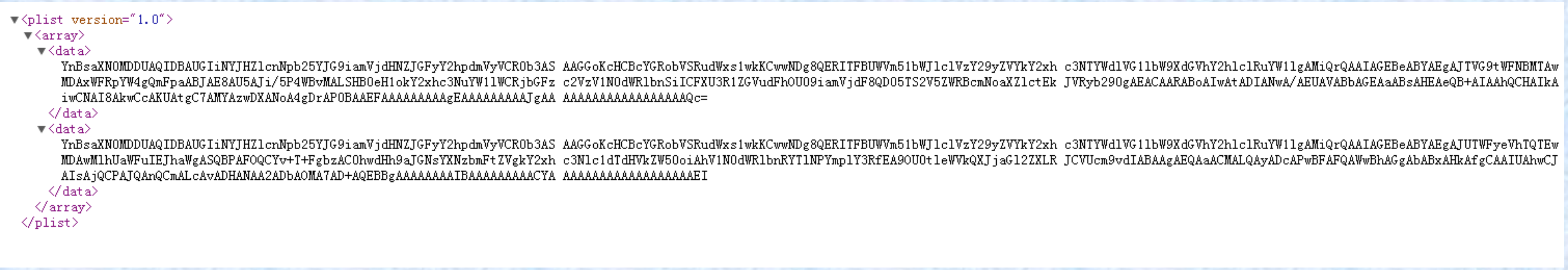
rootObject	The root of the object graph to archive.
path	The path of the file in which to write the archive.

Return Value

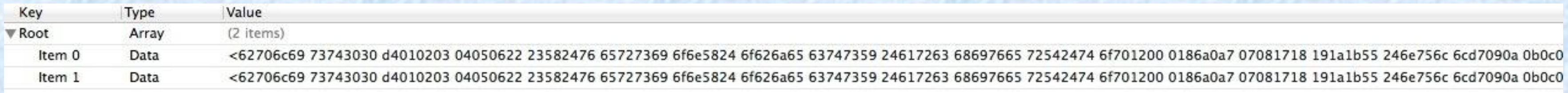
YES if the operation was successful, otherwise NO.



```
NSString *doc = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
NSString *path = [doc stringByAppendingPathComponent:@"students.plist"];
NSMutableArray *students=[NSMutableArray array];
Student *student1=[[Student alloc] init];
student1.name=self.TxtName.text;
student1.number=self.TxtNumber.text;
student1.age=[self.TxtAge.text floatValue];
student1.score=[self.TxtScore.text floatValue];
student1.memo=self.TxtMemo.text;
student1.teacher=@"Tian Bai";

NSData *dataone=[NSKeyedArchiver archivedDataWithRootObject:student1];
[students addObject:dataone];
Student *student2=[[Student alloc] init];
student2.name=@"Mary";
student2.number=@"SA100002";
student2.age=[self.TxtAge.text floatValue];
student2.score=[self.TxtScore.text floatValue];
student2.memo=self.TxtMemo.text;
student2.teacher=@"Tian Bai";

NSData *datatwo=[NSKeyedArchiver archivedDataWithRootObject:student2];
[students addObject:datatwo];
[students writeToFile:path atomically:YES];
```

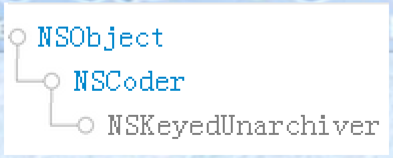


```
▼<plist version="1.0">
  ▼<array>
    ▼<data>
      YnBsaXN0MDDUAQIDBAUGIiNYJHZlcnNpb25YJG9iamVjdHNZJGFyY2hpdmVyVCR0b3AS AAGGoKcHCBcYGRobVSRudWxs1wkKCwwNDg8QERITFBUWVm51bWJlclVzY29yZVYkY2xh c3NTYWdlVG1lbW9XdGVhY2hlclRuYW1lgAMiQrQAAIAGEBeABYAEgAJTVG9tWFNBMTAw
      MDAxWFRpYW4gQmFpaABJAE8AU5AJi/5P4WBvMALSHB0eH1okY2xhc3NuYW1lWCRjbGFz c2VzV1N0dWRlbnSiICFXU3R1ZGVudFhOU09iamVjdF8QD05TS2V5ZWRBcmNoaXZlctEk JVRyb290gAEACAARABoAIwAtADIANwA/AEUAVABbAGEAaABsAHEAeQB+AIAAhQCHAIkA
      iwCNAI8AkwCcAKUAtgC7AMYAzwDXANoA4gDrAP0BAAEFAAAAAAAAAgEAAAAAAAAAJgAA AAAAAAAAAAAAAAAAAQc=
    </data>
    ▼<data>
      YnBsaXN0MDDUAQIDBAUGIiNYJHZlcnNpb25YJG9iamVjdHNZJGFyY2hpdmVyVCR0b3AS AAGGoKcHCBcYGRobVSRudWxs1wkKCwwNDg8QERITFBUWVm51bWJlclVzY29yZVYkY2xh c3NTYWdlVG1lbW9XdGVhY2hlclRuYW1lgAMiQrQAAIAGEBeABYAEgAJUTWFyeVhTQTEw
      MDAwMlhUaWFuIEJhaWgASQBPAFOQCYv+T+FgbzAC0hwdHh9aJGNsYXNzbmFtZVgkY2xh c3Nlc1dTdHVkZW50oiAhV1N0dWRlbnRYTlNPYmplY3RfEA9OU0tleWVkQXJjaGl2ZXLR JCVUcm9vdIABAAgAEQAaACMALQAyADcAPwBFAFQAWwBhAGgAbABxAHkAfgCAAIUAhwCJ
      AIsAjQCPAJQAnQCmALcAvADHANAA2ADbAOMA7AD+AQEBBgAAAAAAAAIBAAAAAAAAACYA AAAAAAAAAAAAAAAAAAAEI
    </data>
  </array>
</plist>
```



Key	Type	Value
▼Root	Array	(2 items)
Item 0	Data	<62706c69 73743030 d4010203 04050622 23582476 65727369 6f6e5824 6f626a65 63747359 24617263 68697665 72542474 6f701200 0186a0a7 07081718 191a1b55 246e756c 6cd7090a 0b0c0
Item 1	Data	<62706c69 73743030 d4010203 04050622 23582476 65727369 6f6e5824 6f626a65 63747359 24617263 68697665 72542474 6f701200 0186a0a7 07081718 191a1b55 246e756c 6cd7090a 0b0c0

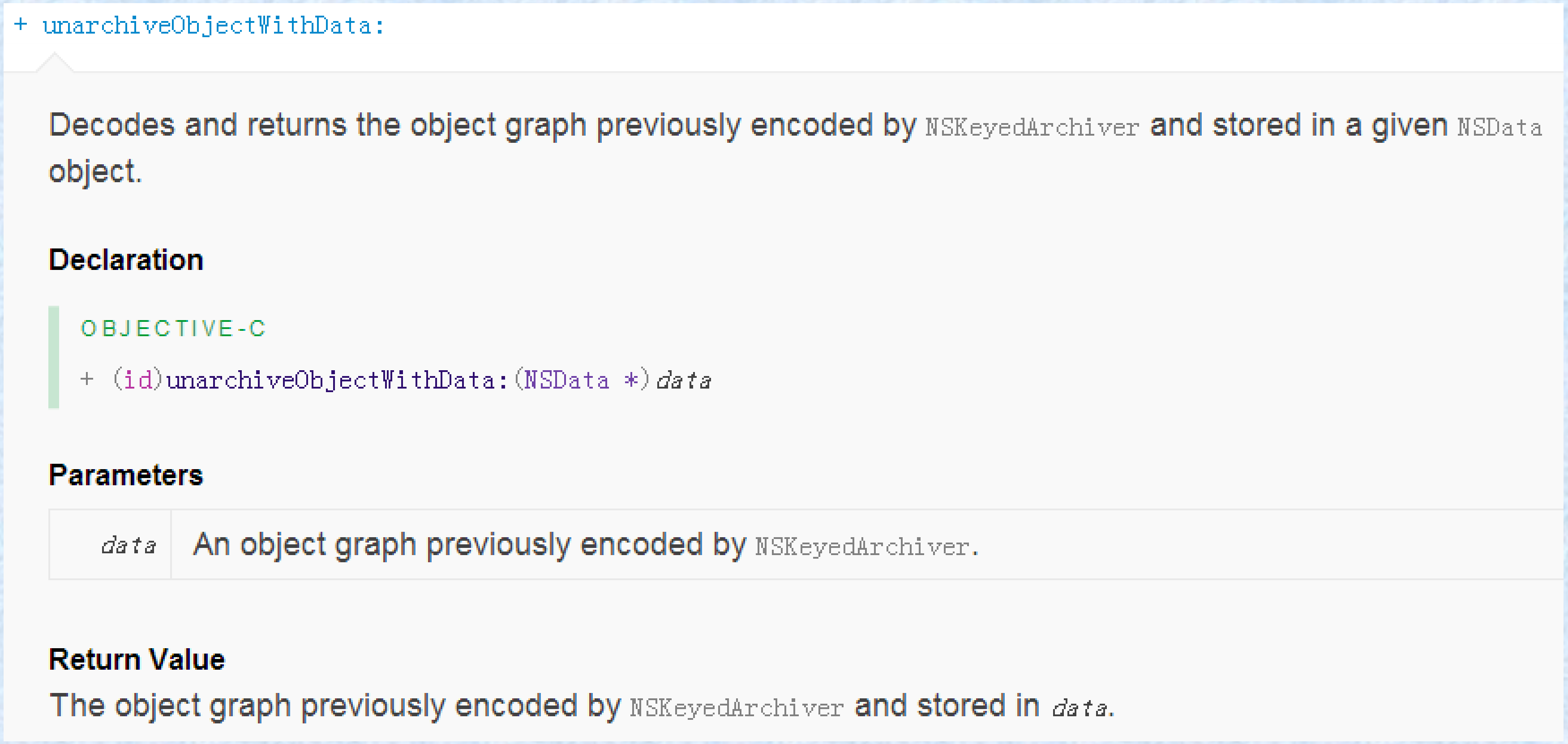
3、解档NSKeyedUnarchiver



```
NSObject
└ NSCoder
  └ NSKeyedUnarchiver
```

NSKeyedUnarchiver, a concrete subclass of NSCoder, defines methods for decoding a set of named objects (and scalar values) from a keyed archive. Such archives are produced by instances of the NSKeyedArchiverclass.

从NSData解档


```
+ unarchiveObjectWithData:
```

Decodes and returns the object graph previously encoded by `NSKeyedArchiver` and stored in a given `NSData` object.

Declaration

OBJECTIVE-C

```
+ (id)unarchiveObjectWithData:(NSData *)data
```

Parameters

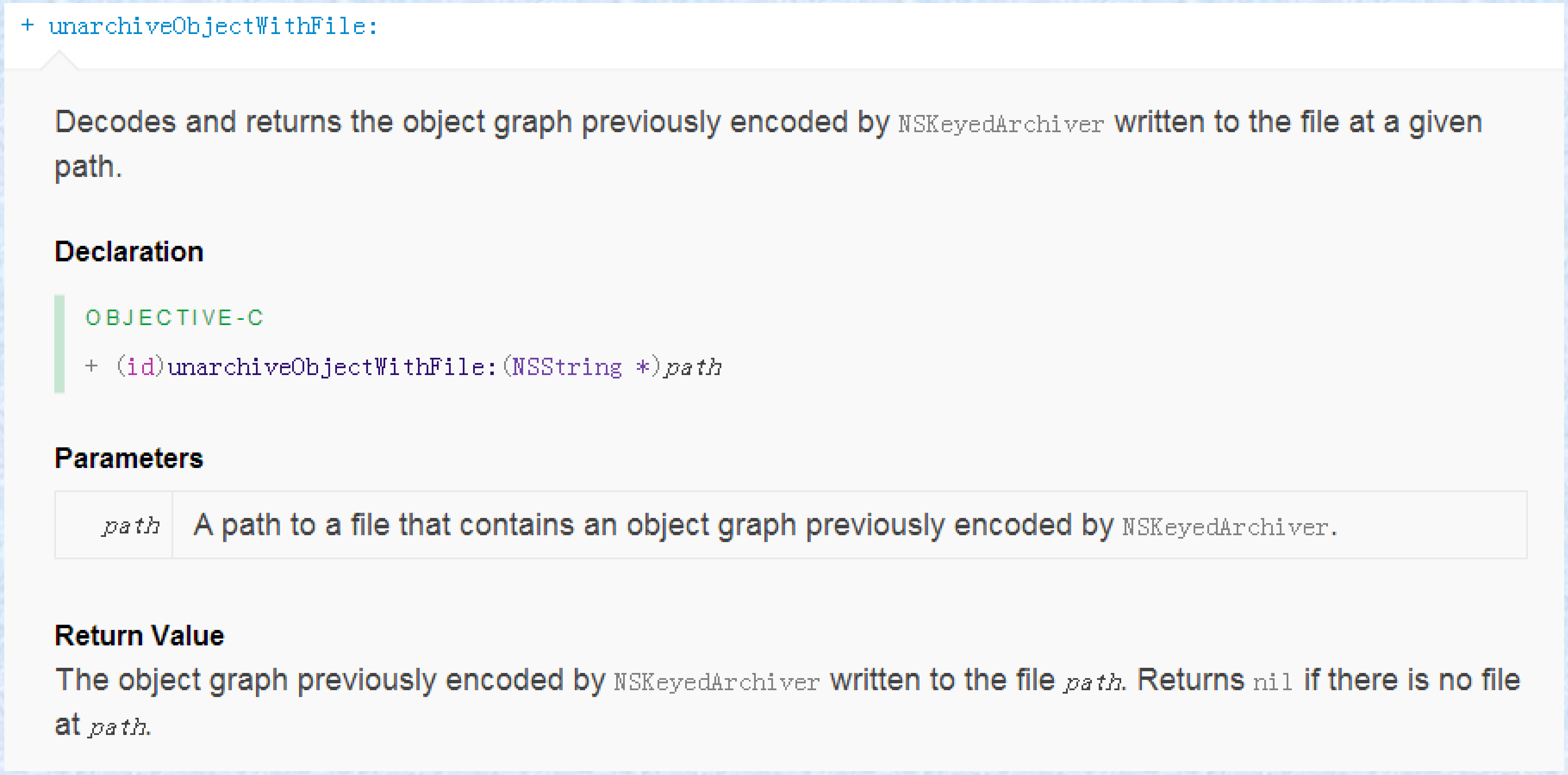
data	An object graph previously encoded by `NSKeyedArchiver`.

Return Value

The object graph previously encoded by `NSKeyedArchiver` and stored in *data*.

```
Student *student=[NSKeyedUnarchiver  unarchiveObjectWithData:data];
```

从文件解档



+ unarchiveObjectWithFile:

Decodes and returns the object graph previously encoded by `NSKeyedArchiver` written to the file at a given path.

Declaration

OBJECTIVE-C

```
+ (id)unarchiveObjectWithFile:(NSString *)path
```

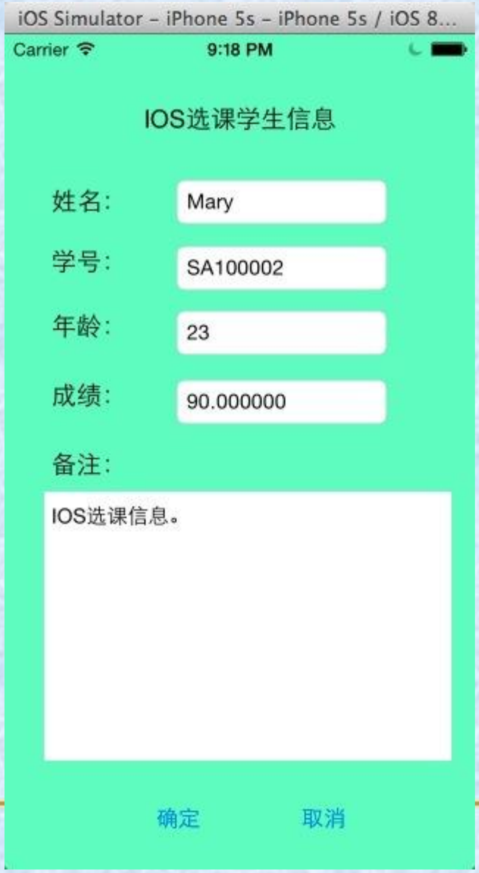
Parameters

path	A path to a file that contains an object graph previously encoded by `NSKeyedArchiver`.

Return Value

The object graph previously encoded by `NSKeyedArchiver` written to the file *path*. Returns `nil` if there is no file at *path*.

```
NSString *doc = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
NSString *path = [doc stringByAppendingPathComponent:@"student.plist"];
NSMutableArray *students=[NSMutableArray arrayWithContentsOfFile:path] ;
Student *student=[NSKeyedUnarchiver unarchiveObjectWithData:students[1]];
self.TxtName.text=student.name;
self.TxtNumber.text=student.number;
self.TxtAge.text=[NSString stringWithFormat:@"%ld",(long)student.age];
self.TxtScore.text=[NSString stringWithFormat:@"%f",student.score];
self.TxtMemo.text=student.memo;
```


iOS Simulator - iPhone 5s - iPhone 5s / iOS 8...

Carrier 9:18 PM

IOS选课学生信息

姓名: Mary

学号: SA100002

年龄: 23

成绩: 90.000000

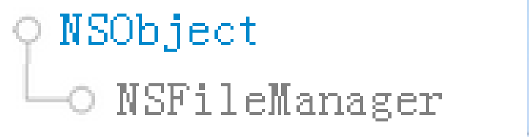
备注:

IOS选课信息。

注意：
1、自定义对象需要遵守NSCoding协议，并实现该协议中的encodeWithCoder方法和initWithCoder方法。

2、如果是继承某个自定义对象，则子类一定要重写上述两个方法

3、保存后的文件的后缀名可以随意命名。



五、利用NSFileManager存取文件

An NSFileManager object lets you examine the contents of the file system and make changes to it. A file manager object is usually your first interaction with the file system. You use it to locate, create, copy, and move files and directories. You also use it to get information about a file or directory or change some of its attributes.

The NSFileManager class provides convenient access to a shared file manager object that is suitable for most types of file-related manipulations. If you plan to use the file manager object interactively—for example, if you plan to assign a delegate object to it—create your own instance of the class.