ALPHAcamp

資料儲存

Smart Wang

2016.06.04

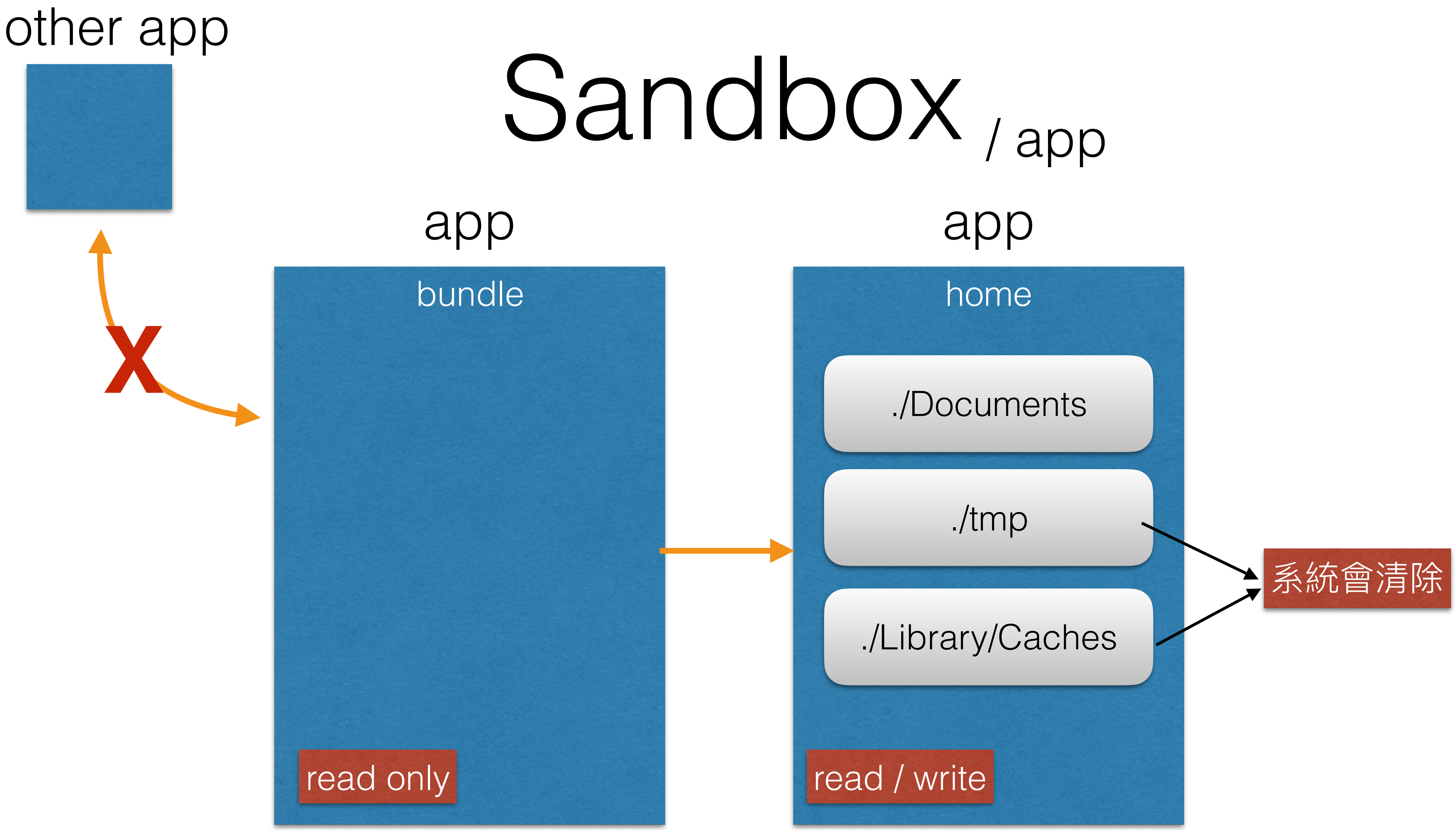
課程大綱

- File
- plist
- NSUserDefaults
- Core Data

Persistence

- 使用者輸入：文字、照片、影片
- 使用者設定

Sandbox / app



每個App都有獨立的檔案空間　　安全　隱私

https://developer.apple.com/app-sandboxing

bundle

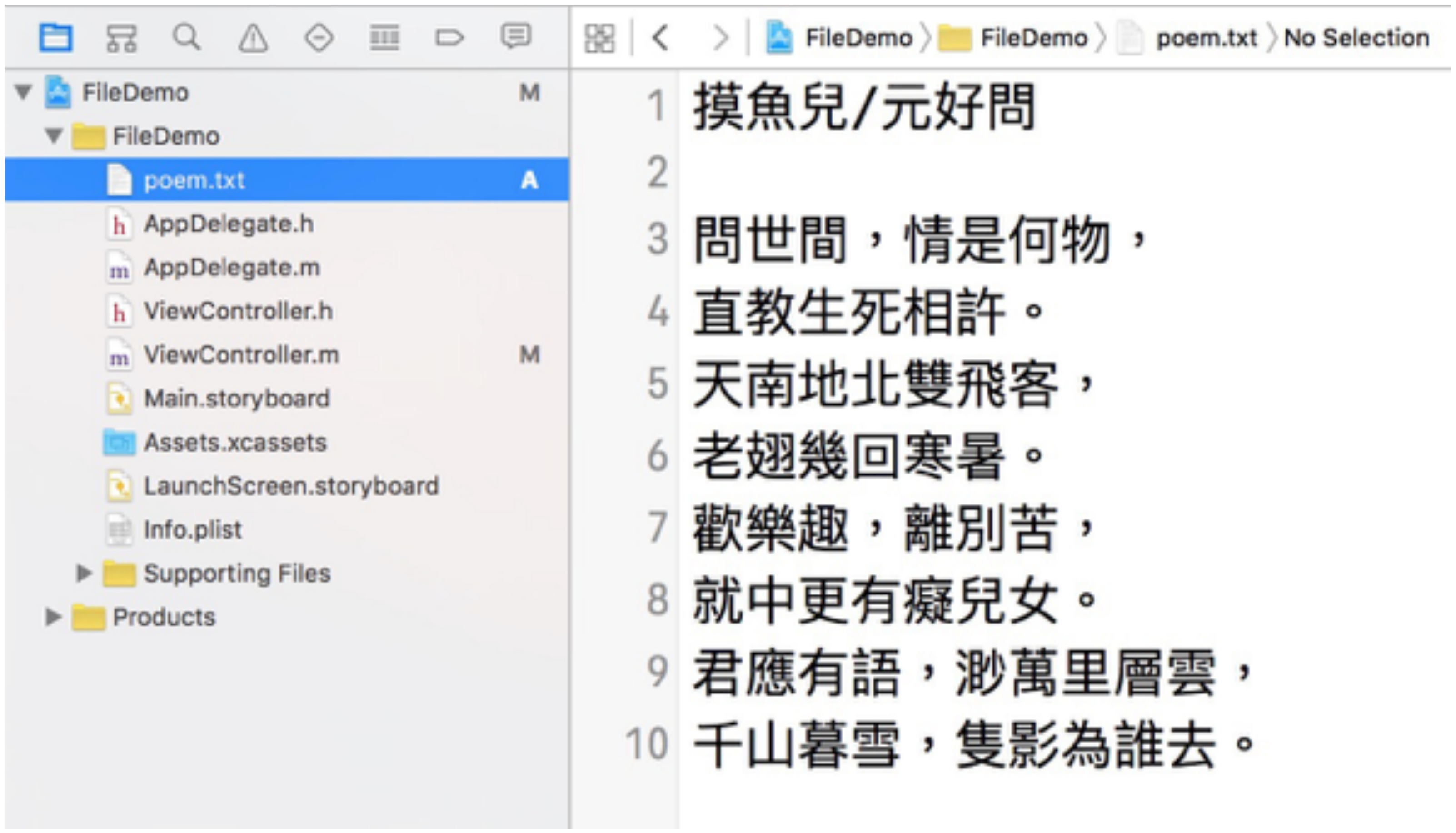
https://developer.apple.com/library/ios/documentation/CoreFoundation/Conceptual/CFBundles/Introduction/Introduction.html

A bundle is a directory with a standardized hierarchical structure that holds executable code and the resources used by that code.

File

讀取專案裡
文字檔案

將檔案poem.txt加到專案裡



```
摸魚兒/元好問

問世間，情是何物，
直教生死相許。
天南地北雙飛客，
老翅幾回寒暑。
歡樂趣，離別苦，
就中更有癡兒女。
君應有語，渺萬里層雲，
千山暮雪，隻影為誰去。
```

讀取專案裡
文字檔案

路徑上的檔案不存在時會回傳nil

```
let path = NSBundle.mainBundle().pathForResource("poem", ofType:
"txt")

let poem = try NSString(contentsOfFile: path!, encoding:
NSUTF8StringEncoding)

print("poem \(poem)")
```

讀檔案字串

NSString

```
public convenience init(contentsOfFile path: String,
encoding enc: UInt) throws
```

encoding

```
public var NSUTF16BigEndianStringEncoding: UInt { get }

public var NSUTF16LittleEndianStringEncoding: UInt { get }

public var NSUTF16StringEncoding: UInt { get }

public var NSUTF32BigEndianStringEncoding: UInt { get }

public var NSUTF32LittleEndianStringEncoding: UInt { get }

public var NSUTF32StringEncoding: UInt { get }

public var NSUTF8StringEncoding: UInt { get }

public var NSUnicodeStringEncoding: UInt { get }

public var NSWindowsCP1250StringEncoding: UInt { get }

public var NSWindowsCP1251StringEncoding: UInt { get }

public var NSWindowsCP1252StringEncoding: UInt { get }

public var NSWindowsCP1253StringEncoding: UInt { get }

public var NSWindowsCP1254StringEncoding: UInt { get }
```

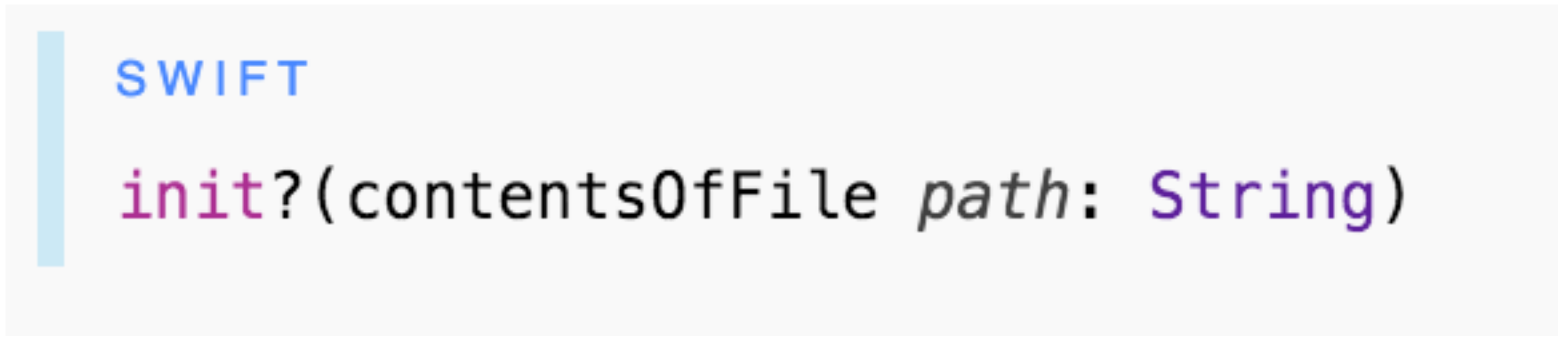
讀檔的init

NSString

NSArray

NSData

NSDictionary


SWIFT

```swift
init?(contentsOfFile path: String)
```

實作練習

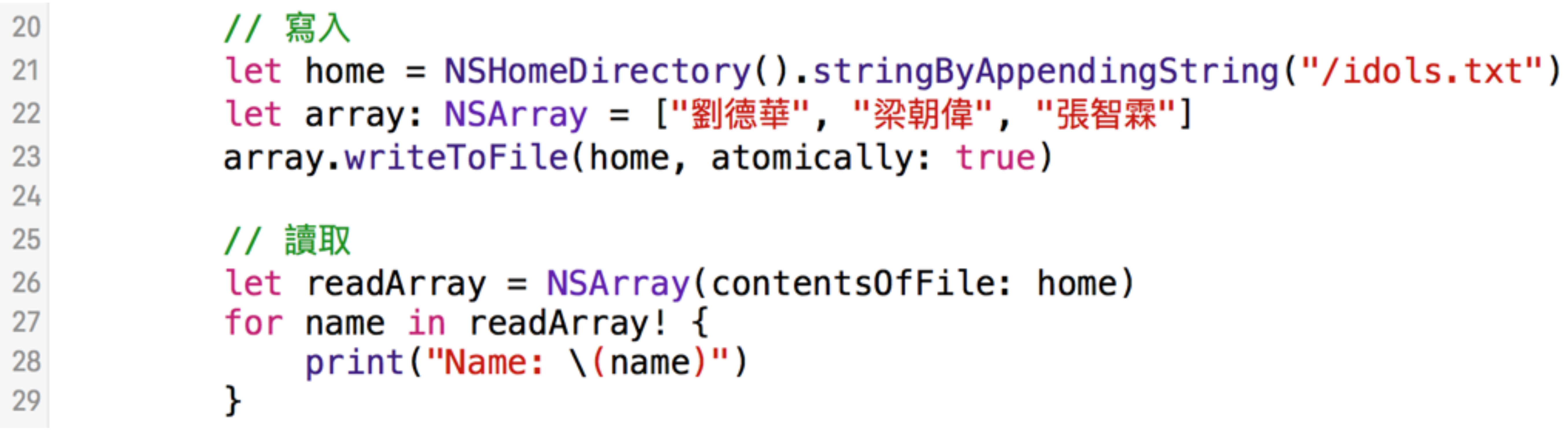
讀取poem.txt的內容，顯示在App的畫面上

目錄

- NSHomeDirectory(): 路徑沒有"/"結尾
- NSTemporaryDirectory(): 路徑有"/"結尾
- NSSearchPathForDirectoriesInDomains()
- NSFileManager

寫入與讀取 v1

NSHomeDirectory()

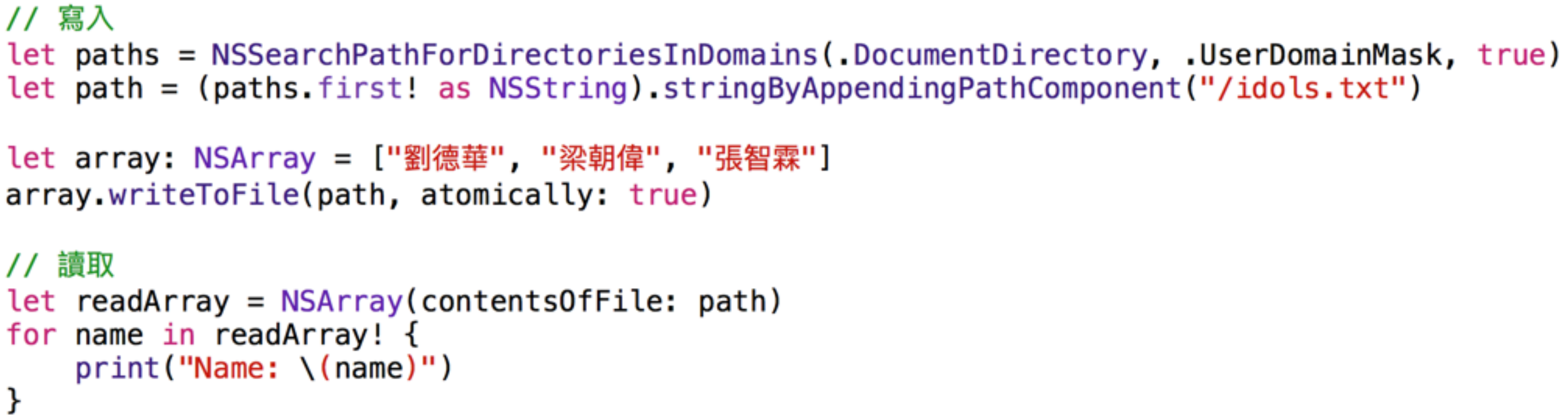


```swift
        // 寫入
        let home = NSHomeDirectory().stringByAppendingString("/idols.txt")
        let array: NSArray = ["劉德華", "梁朝偉", "張智霖"]
        array.writeToFile(home, atomically: true)

        // 讀取
        let readArray = NSArray(contentsOfFile: home)
        for name in readArray! {
            print("Name: \(name)")
        }
```

重覆寫檔，新的會直接覆蓋舊的

寫入與讀取 v2

NSSearchPathForDirectoriesInDomains()



```swift
// 寫入
let paths = NSSearchPathForDirectoriesInDomains(.DocumentDirectory, .UserDomainMask, true)
let path = (paths.first! as NSString).stringByAppendingPathComponent("/idols.txt")

let array: NSArray = ["劉德華", "梁朝偉", "張智霖"]
array.writeToFile(path, atomically: true)

// 讀取
let readArray = NSArray(contentsOfFile: path)
for name in readArray! {
    print("Name: \(name)")
}
```

寫入與讀取 v3

NSFileManager

```
// 寫入
let fileManager = NSFileManager.defaultManager()
let paths = fileManager.URLsForDirectory(.DocumentDirectory, inDomains: .UserDomainMask)
let url = paths.first!.URLByAppendingPathComponent("/idols.txt")

let array: NSArray = ["劉德華", "梁朝偉", "張智霖"]
array.writeToURL(url, atomically: true)

// 讀取
let readArray = NSArray(contentsOfURL: url)
for name in readArray! {
    print("Name: \(name)")
}
```

如果讀取不到，array將是nil

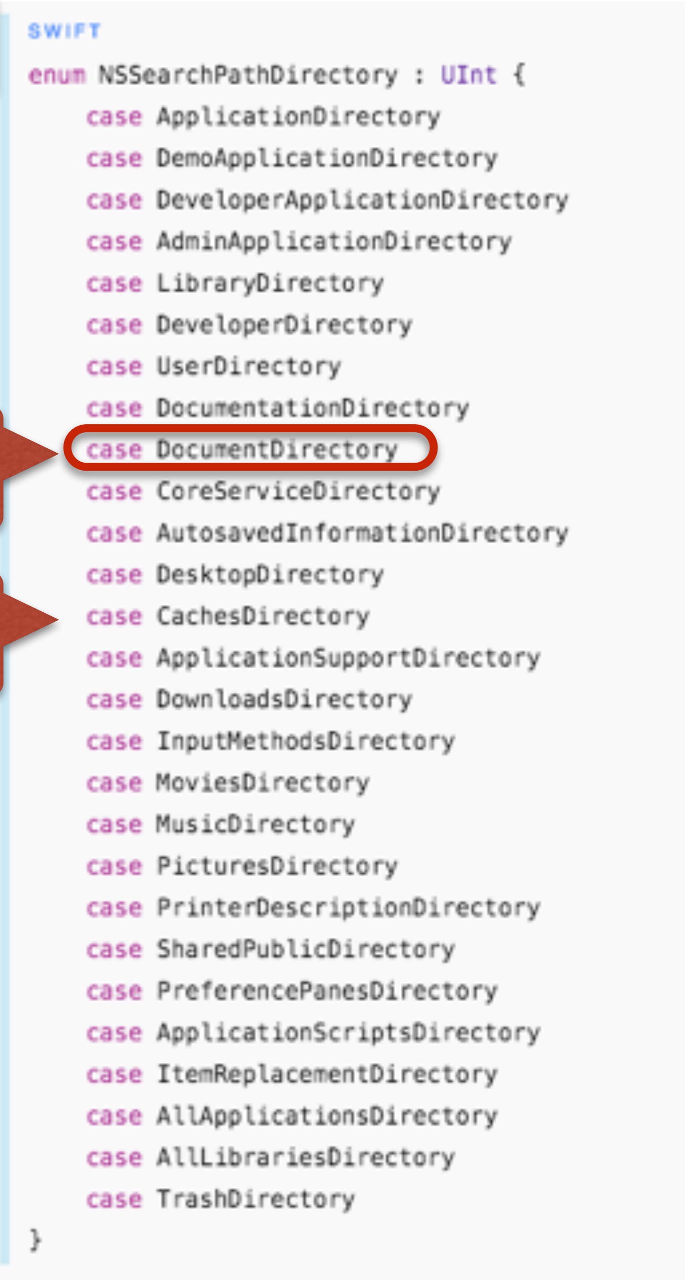
writeToFile:atomically:

```
public func writeToFile(path: String, atomically
useAuxiliaryFile: Bool) -> Bool
```

NSString NSArray NSData NSDictionary

```
public func
NSSearchPathForDirectoriesInDomains(directory:
NSSearchPathDirectory, _ domainMask:
NSSearchPathDomainMask, _ expandTilde: Bool) ->
[String]
```

directory

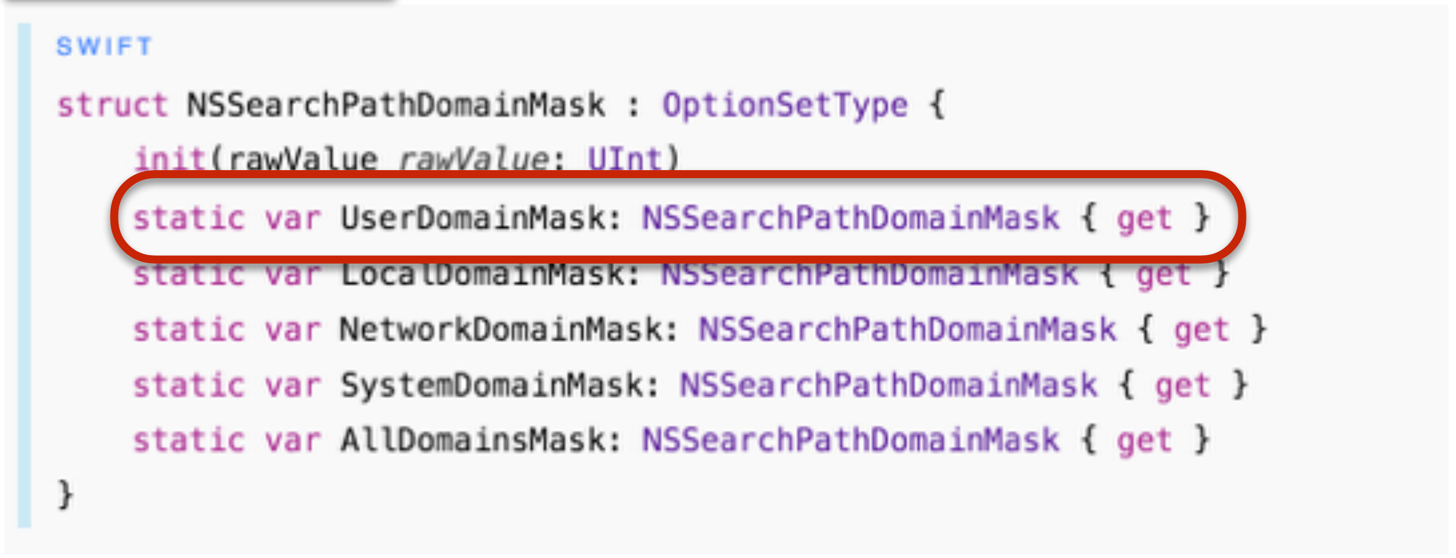


```
SWIFT
enum NSSearchPathDirectory : UInt {
    case ApplicationDirectory
    case DemoApplicationDirectory
    case DeveloperApplicationDirectory
    case AdminApplicationDirectory
    case LibraryDirectory
    case DeveloperDirectory
    case UserDirectory
    case DocumentationDirectory
    case DocumentDirectory
    case CoreServiceDirectory
    case AutosavedInformationDirectory
    case DesktopDirectory
    case CachesDirectory
    case ApplicationSupportDirectory
    case DownloadsDirectory
    case InputMethodsDirectory
    case MoviesDirectory
    case MusicDirectory
    case PicturesDirectory
    case PrinterDescriptionDirectory
    case SharedPublicDirectory
    case PreferencePanesDirectory
    case ApplicationScriptsDirectory
    case ItemReplacementDirectory
    case AllApplicationsDirectory
    case AllLibrariesDirectory
    case TrashDirectory
}
```

常用

常用

domainMask



```
SWIFT
struct NSSearchPathDomainMask : OptionSetType {
    init(rawValue rawValue: UInt)
    static var UserDomainMask: NSSearchPathDomainMask { get }
    static var LocalDomainMask: NSSearchPathDomainMask { get }
    static var NetworkDomainMask: NSSearchPathDomainMask { get }
    static var SystemDomainMask: NSSearchPathDomainMask { get }
    static var AllDomainsMask: NSSearchPathDomainMask { get }
}
```

expandTilde false

```
Path: ["/Users/SmartWang/Library/Developer/CoreSimulator/Devices/A060DC44-5492-4F70-B7F2-4E74F719B10D/data/Containers/Data/Application/FD3F72D9-B474-4F39-8805-3B8B98E5C132/Documents"]
```

true

```
Path: ["~/Documents"]
```

動態建立的檔案路徑

模擬器路徑

```
/Users/PeterPan/Library/Developer/CoreSimulator/Devices/4725B36F-7422-43DB-9F30-4EF8595D18B2/
data/Containers/Data/Application/E2F8D72B-ACD7-448B-98E1-B19BFED212C1/Documents/我的偶像.txt
```

實機路徑

```
/var/mobile/Containers/Data/Application/
6112DBFC-0AB6-4E7E-976B-19A2E1CB5CA8/Documents/我的偶像.txt
```

實作練習

第一個畫面顯示電影列表和新增button，
點選新增button進入新增頁面，可在其中新增電影，
新增成功後會到列表，列表上將多出新的電影。
重新啟動App後，之前做的修改還是會存在。

NSFileManager

- 建立、複製、移動檔案和目錄
- 取得或改變檔案或目錄的屬性

create directory

```
do {
    try
NSFileManager.defaultManager().createDirectoryAtURL(url,
withIntermediateDirectories: true, attributes: nil)
} catch {

}
```

```
 public func createDirectoryAtURL(url: NSURL,
withIntermediateDirectories createIntermediates: Bool, attributes:
[String : AnyObject]?) throws
```

withIntermediateDirectories

If NO, this method fails if any of the intermediate parent directories does not exist.

search

```
do {
    let contentArray = try
NSFileManager.defaultManager().contentsOfDirectoryAtURL(url,
includingPropertiesForKeys: nil, options: .SkipsHiddenFiles)
} catch {

}
```

```
public func contentsOfDirectoryAtURL(url: NSURL,
includingPropertiesForKeys keys: [String]?, options mask:
NSDirectoryEnumerationOptions) throws -> [NSURL]
```

delete

```
do {
    try NSFileManager.defaultManager().removeItemAtURL(url)
} catch {
}
```

```
 public func removeItemAtURL(URL: NSURL) throws
```

範例

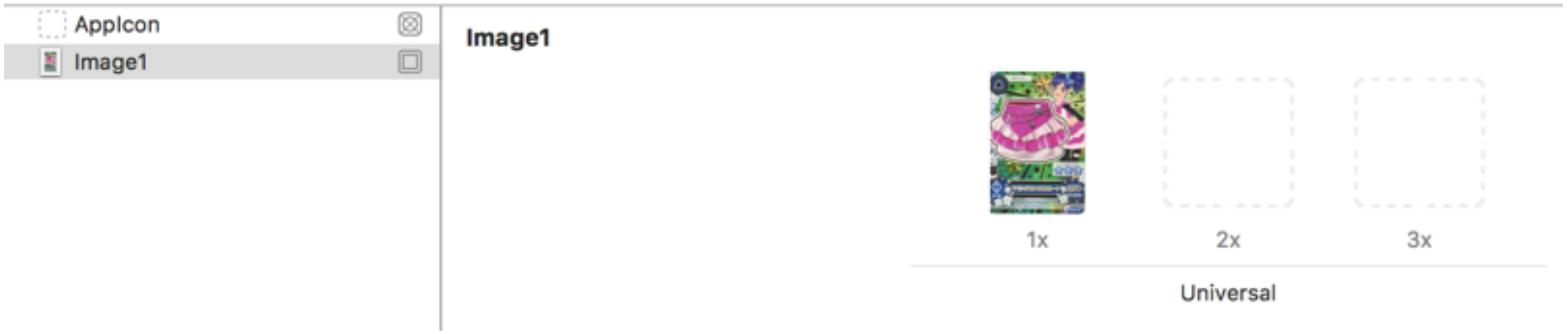
```
let fileManager = NSFileManager.defaultManager()
let docUrls = fileManager.URLsForDirectory(.DocumentDirectory, inDomains: .UserDomainMask)
let docUrl = docUrls.first
let url = docUrl?.URLByAppendingPathComponent("Peter/Text")
do {
    try fileManager.createDirectoryAtURL(url!, withIntermediateDirectories: true, attributes: nil)
    let url1 = url?.URLByAppendingPathComponent("text1.txt")
    try "相戀的失戀的請跟我來，一邊跳一邊向快樂崇拜".writeToURL(url1!, atomically: true, encoding: 
NSUTF8StringEncoding)

    let url2 = url?.URLByAppendingPathComponent("text2.txt")
    try "開心不開心的都跟我來，美麗而神聖的時光不等待".writeToURL(url2!, atomically: true, encoding: 
NSUTF8StringEncoding)

    var contentArray = try fileManager.contentsOfDirectoryAtURL(url!, includingPropertiesForKeys: 
nil, options: .SkipsHiddenFiles)
    for dataUrl in contentArray {
        print("data \(dataUrl)")
    }

    try fileManager.removeItemAtURL(url1!)
        contentArray = try fileManager.contentsOfDirectoryAtURL(url!, includingPropertiesForKeys: 
nil, options: .SkipsHiddenFiles)

    for dataUrl in contentArray {
        print("data \(dataUrl)")
    }

} catch {

}
```

讀取專案裡的圖片

配合 image assets

```
let image = UIImage(named: "Image1")
image1.image = image
```

不用附檔名

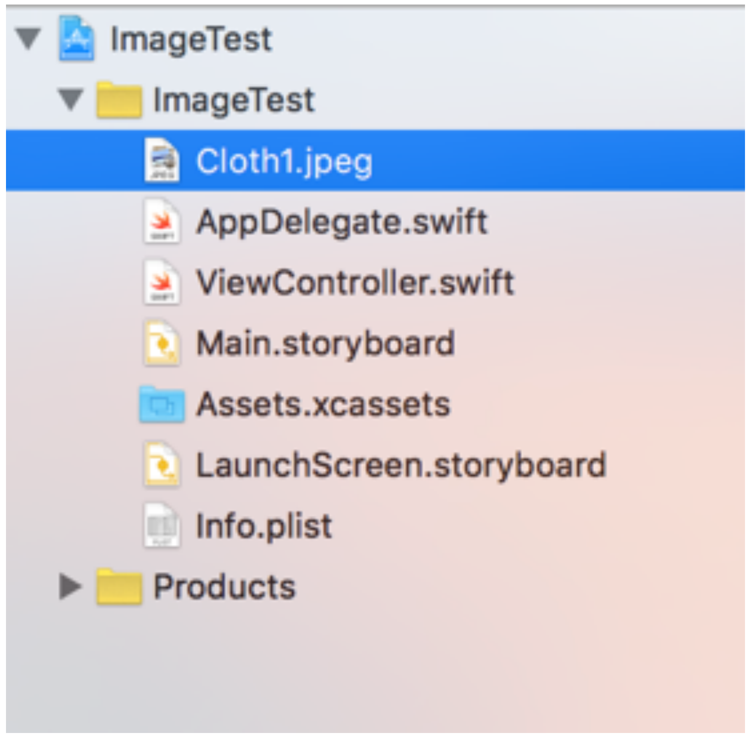


讀取專案裡的圖片

配合 image file

```swift
let image = UIImage(named: "Cloth1.jpeg")
image1.image = image
```

有附檔名



UIImage的存取

UIImage
 ex: 拍照，網路下載的圖片

```
public func UIImagePNGRepresentation(image: UIImage) -> NSData?

public func UIImageJPEGRepresentation(image: UIImage, _
compressionQuality: CGFloat) -> NSData?
```

```
public init?(contentsOfFile path: String)
public init?(data: NSData)
```

讀寫圖片

寫入 image file

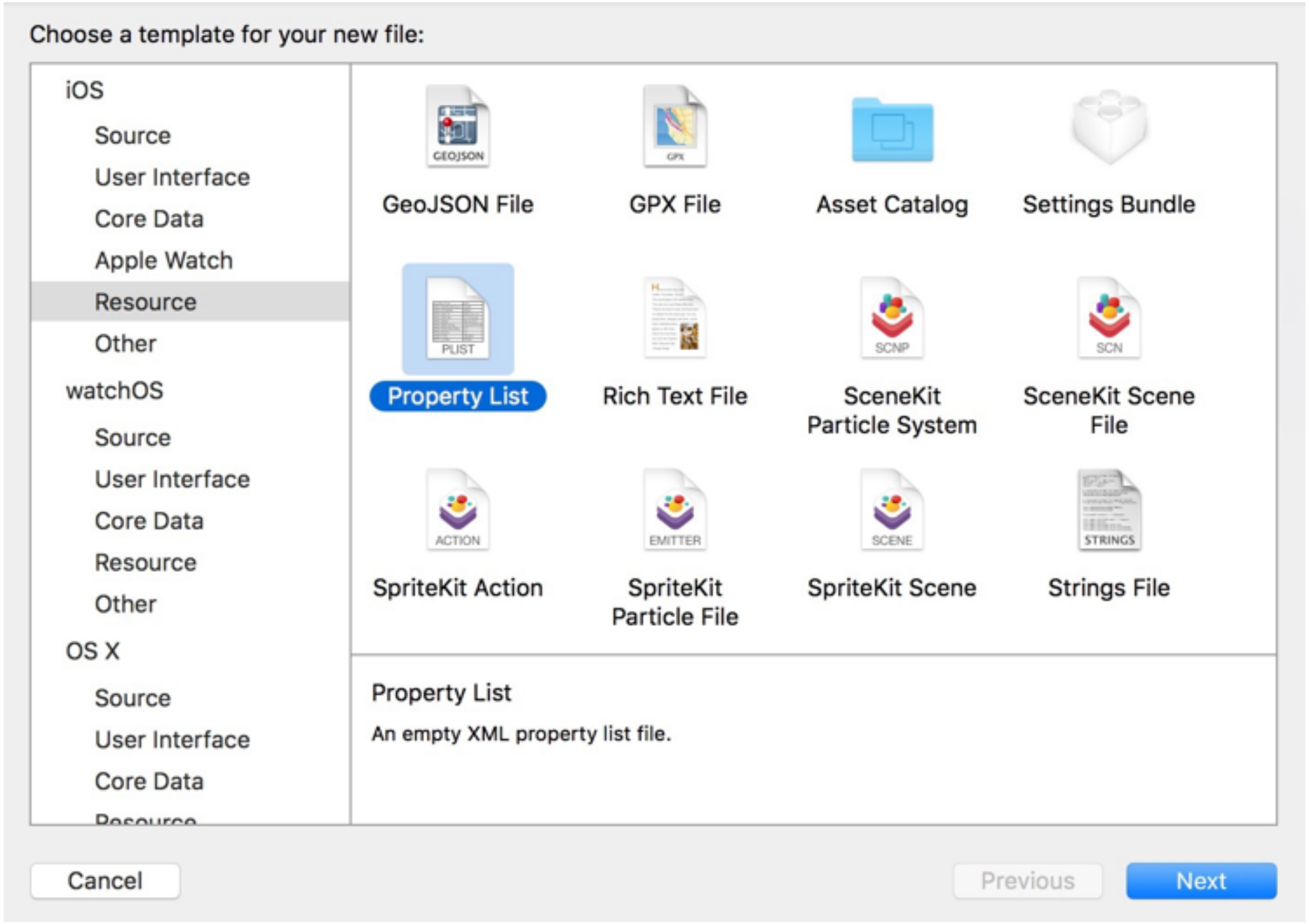
```
let image = UIImage(named: "Cloth1.jpeg")
image1.image = image

let imageData: NSData = UIImageJPEGRepresentation(image!, 1)!
let home = NSHomeDirectory().stringByAppendingString("/Documents/writeImg.jpg")
imageData.writeToFile(home, atomically: true)
```

讀取寫入的 image file

```
let home = NSHomeDirectory().stringByAppendingString("/Documents/writeImg.jpg")
let imageData = NSData(contentsOfFile: home)
let image = UIImage(data: imageData!, scale: 1)
image1.image = image
```

plist

property list



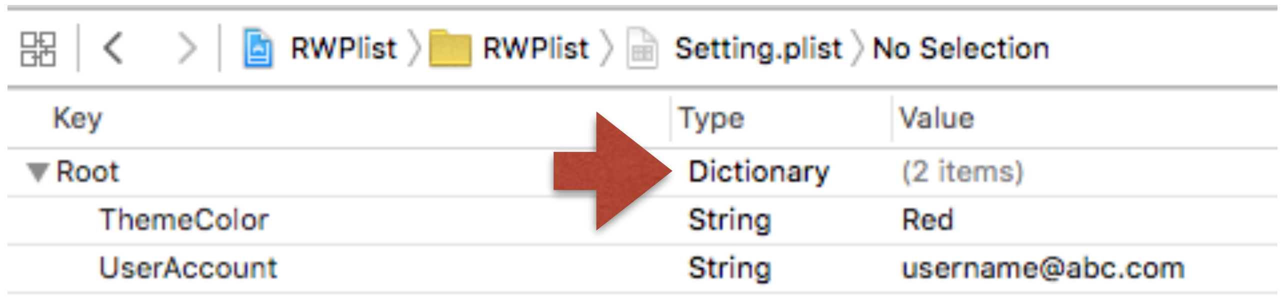
Choose a template for your new file:

- iOS
  - Source
  - User Interface
  - Core Data
  - Apple Watch
  - Resource
  - Other
- watchOS
  - Source
  - User Interface
  - Core Data
  - Resource
  - Other
- OS X
  - Source
  - User Interface
  - Core Data
  - Resource

Property List

An empty XML property list file.

property list



Key	Type	Value
▼ Root	Dictionary	(2 items)
ThemeColor	String	Red
UserAccount	String	username@abc.com

讀取property list

```
let path = NSBundle.mainBundle().pathForResource("Setting", ofType: "plist")
let fm = NSFileManager.defaultManager()
if fm.fileExistsAtPath(path!) {
    print("File exist")

    let dict = NSDictionary(contentsOfFile: path!)
    let value = dict!["ThemeColor"]
    print(value!)
}
```

output

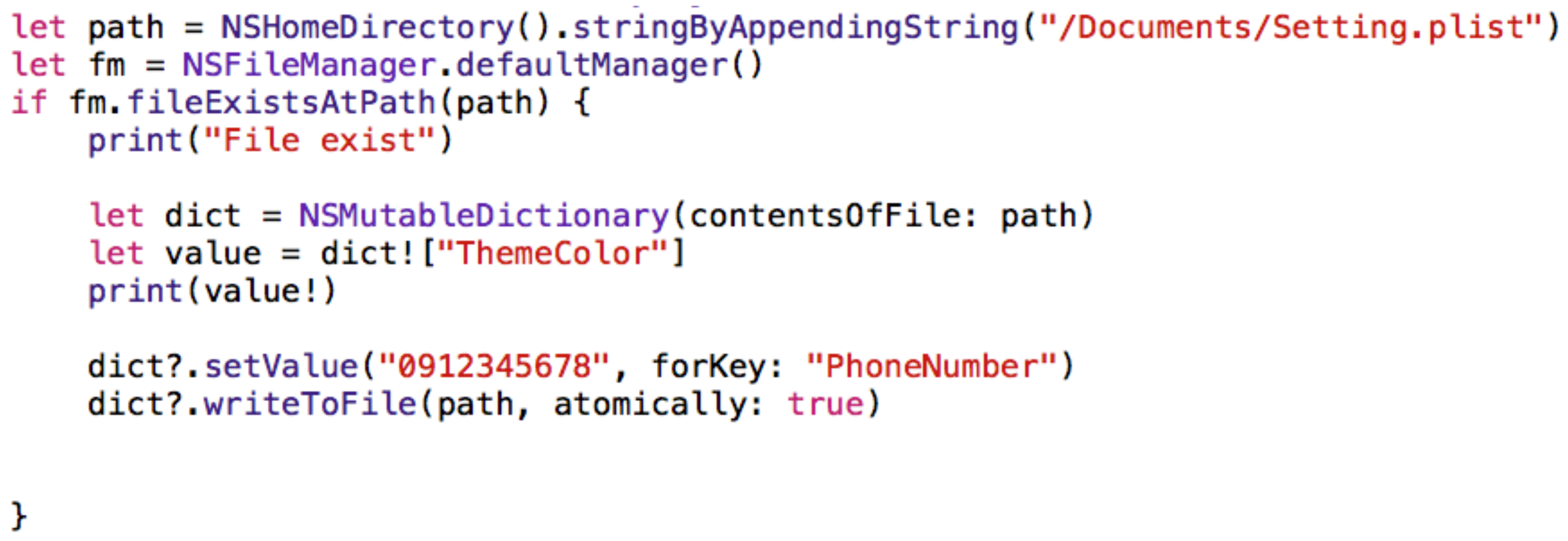
```
File exist
Red
```

複製property list

copy原本專案裡的plist到可以寫入的document directory

```
let srcPath = NSBundle.mainBundle().pathForResource("Setting", ofType: "plist")
let dstPath = NSHomeDirectory().stringByAppendingString("/Documents/Setting.plist")

let fm = NSFileManager.defaultManager()
if !fm.fileExistsAtPath(dstPath) {
    do {
        try fm.copyItemAtPath(srcPath!, toPath: dstPath)
    } catch {
        //
    }
}
```

寫入property list



```
let path = NSHomeDirectory().stringByAppendingString("/Documents/Setting.plist")
let fm = NSFileManager.defaultManager()
if fm.fileExistsAtPath(path) {
    print("File exist")

    let dict = NSMutableDictionary(contentsOfFile: path)
    let value = dict!["ThemeColor"]
    print(value!)

    dict?.setValue("0912345678", forKey: "PhoneNumber")
    dict?.writeToFile(path, atomically: true)


}
```

實作練習

將使用者設定(自定plist)ThemeColor、UserAccount、PhoneNumber讀出在畫面上，修改設定之後，再寫回plist。重新啟動app時，顯示修改後的設定。

NSUserDefaults

NSUserDefaults

user’s defaults database

每個App都有的property list

永久存在

適用例子: App的相關設定資料

只適合儲存少量資料

存太多資料時，存取也會較花時間

NSUserDefaults

寫入

```
let userDefault = NSUserDefaults.standardUserDefaults()
userDefault.setObject("白彼得", forKey: "name")
userDefault.synchronize()
```

讀取

```
let userDefault = NSUserDefaults.standardUserDefaults()
let name = userDefault.objectForKey("name")
print("name \(name)")
```

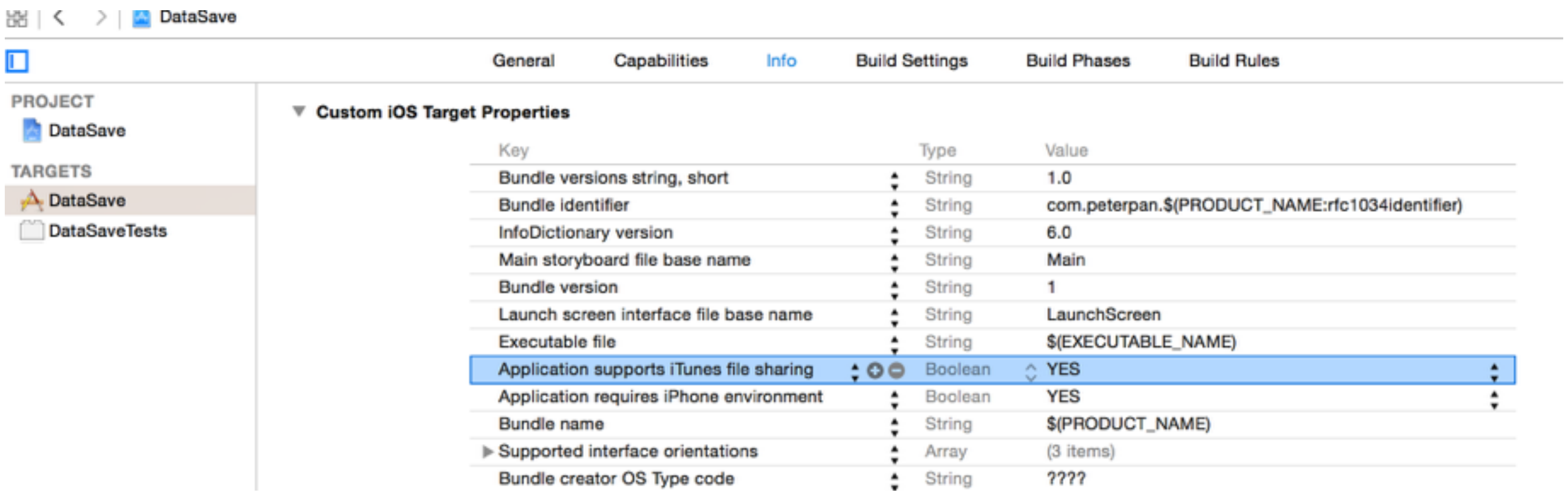
可寫入的資料: NSDate NSString

NSNumber NSData NSDictionary NSArray

synchronize

寫入disk，定期被呼叫，自己呼叫會更保險

iOS & iTunes File Sharing



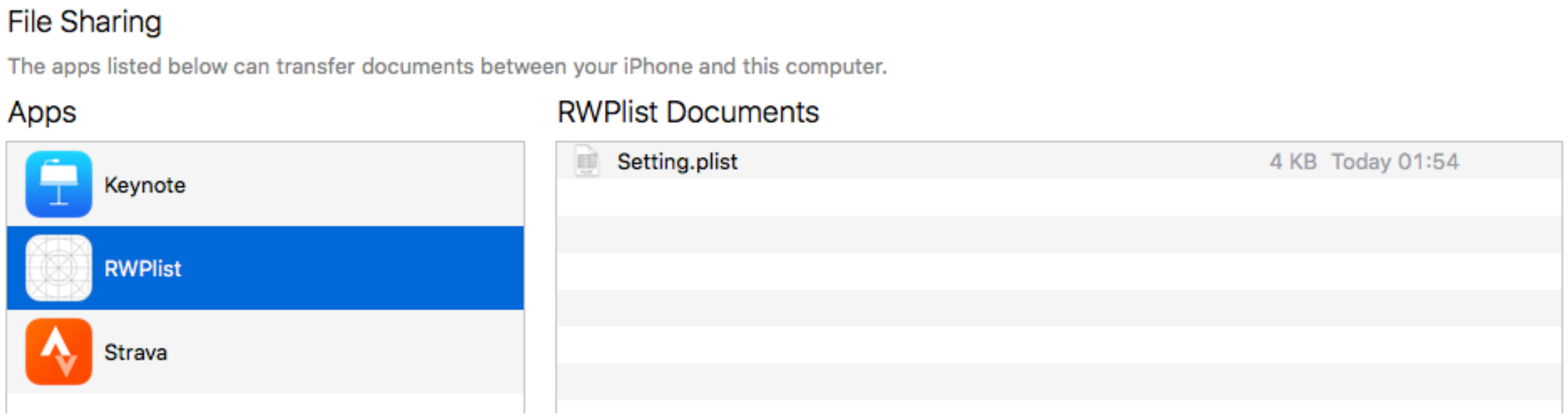
Custom iOS Target Properties

Key	Type	Value
Bundle versions string, short	String	1.0
Bundle identifier	String	com.peterpan.$(PRODUCT_NAME:rfc1034identifier)
InfoDictionary version	String	6.0
Main storyboard file base name	String	Main
Bundle version	String	1
Launch screen interface file base name	String	LaunchScreen
Executable file	String	$(EXECUTABLE_NAME)
Application supports iTunes file sharing	Boolean	YES
Application requires iPhone environment	Boolean	YES
Bundle name	String	$(PRODUCT_NAME)
Supported interface orientations	Array	(3 items)
Bundle creator OS Type code	String	????

Application supports iTunes file sharing

分享Documents下的檔案

iOS & iTunes File Sharing



File Sharing

The apps listed below can transfer documents between your iPhone and this computer.

Apps

- Keynote
- RWPlist
- Strava

RWPlist Documents

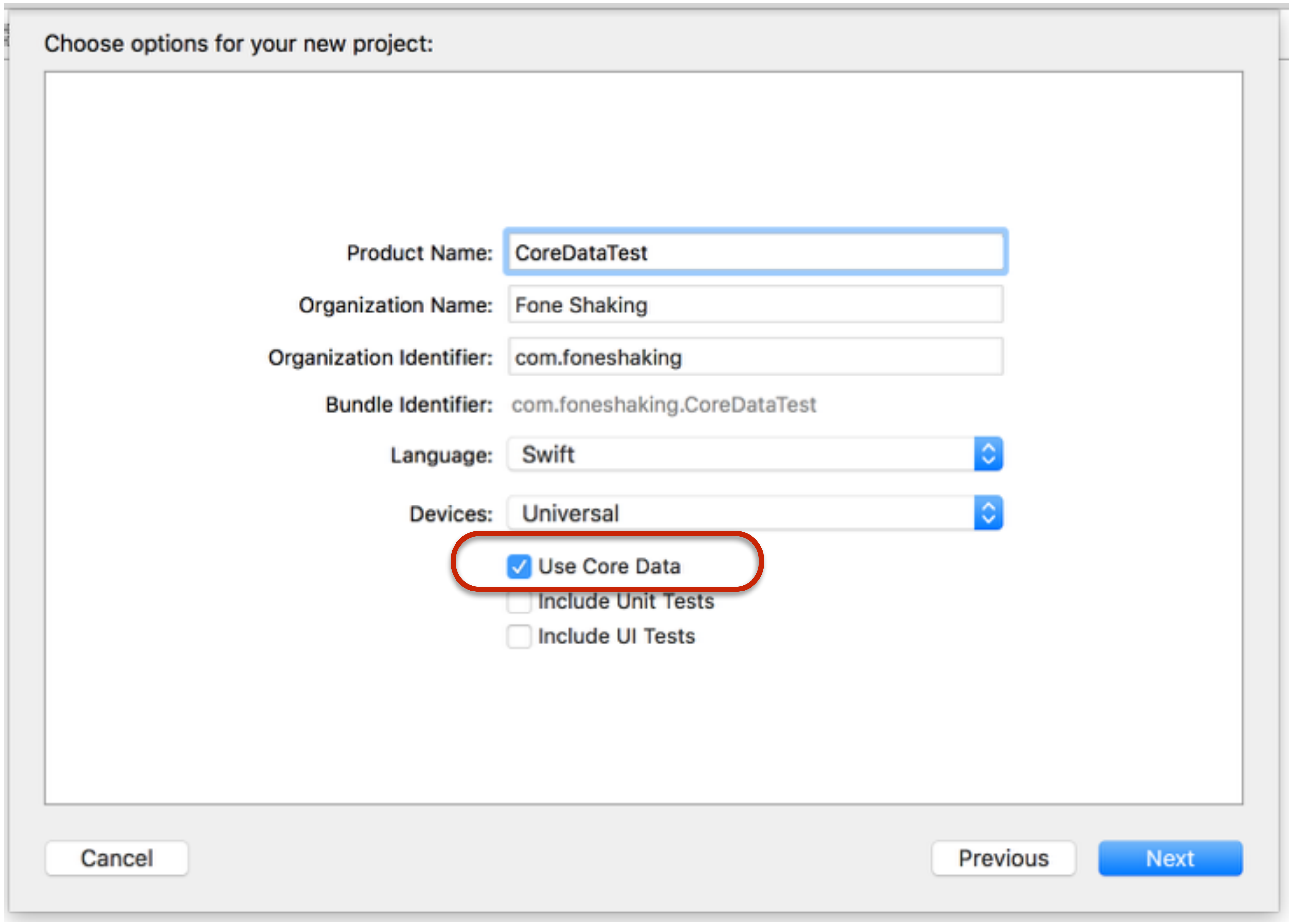
Setting.plist	4 KB	Today 01:54

Core Data

Core Data

- Model
- 以物件方式儲存與讀取資料

new project

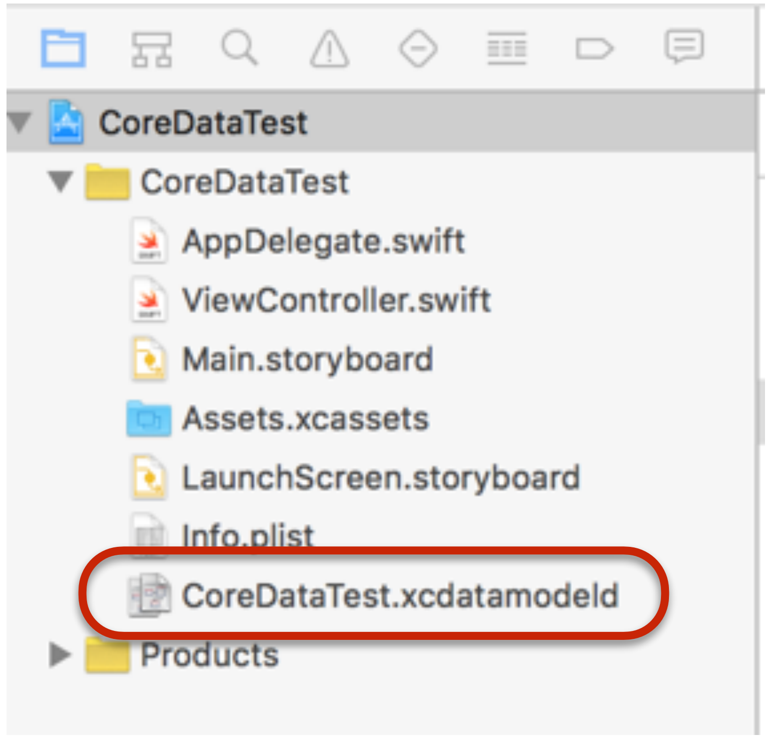


Choose options for your new project:

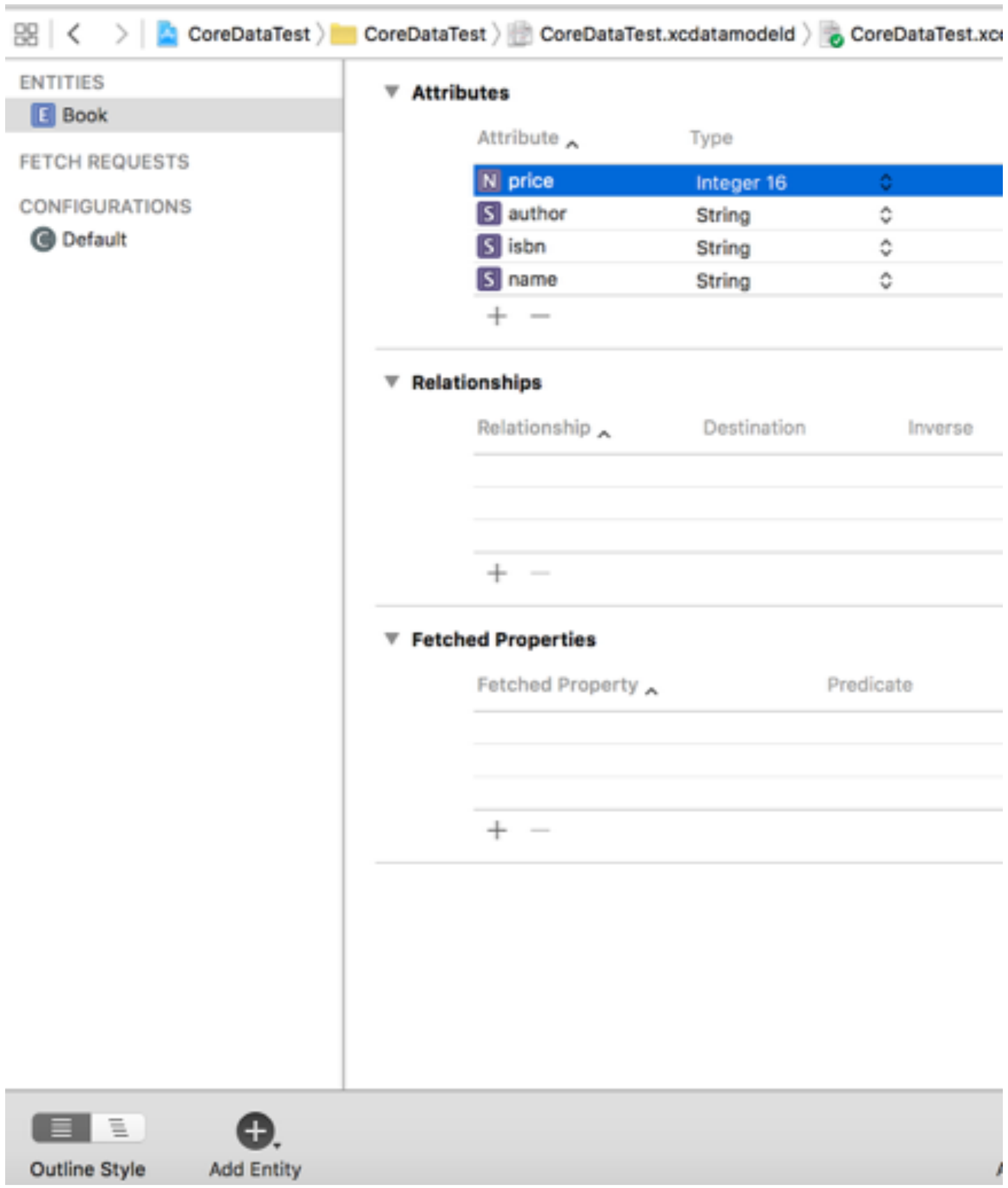
Product Name:	CoreDataTest
Organization Name:	Fone Shaking
Organization Identifier:	com.foneshaking
Bundle Identifier:	com.foneshaking.CoreDataTest
Language:	Swift
Devices:	Universal

- [✓ Use Core Data]
- Include Unit Tests
- Include UI Tests

model



CoreDataTest
CoreDataTest
AppDelegate.swift
ViewController.swift
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
CoreDataTest.xcdatamodeld
Products



CoreDataTest 〉 CoreDataTest 〉 CoreDataTest.xcdatamodeld 〉 CoreDataTest.xc

ENTITIES
Book
FETCH REQUESTS
CONFIGURATIONS
Default

Attributes

Attribute	Type
N price	Integer 16
S author	String
S isbn	String
S name	String

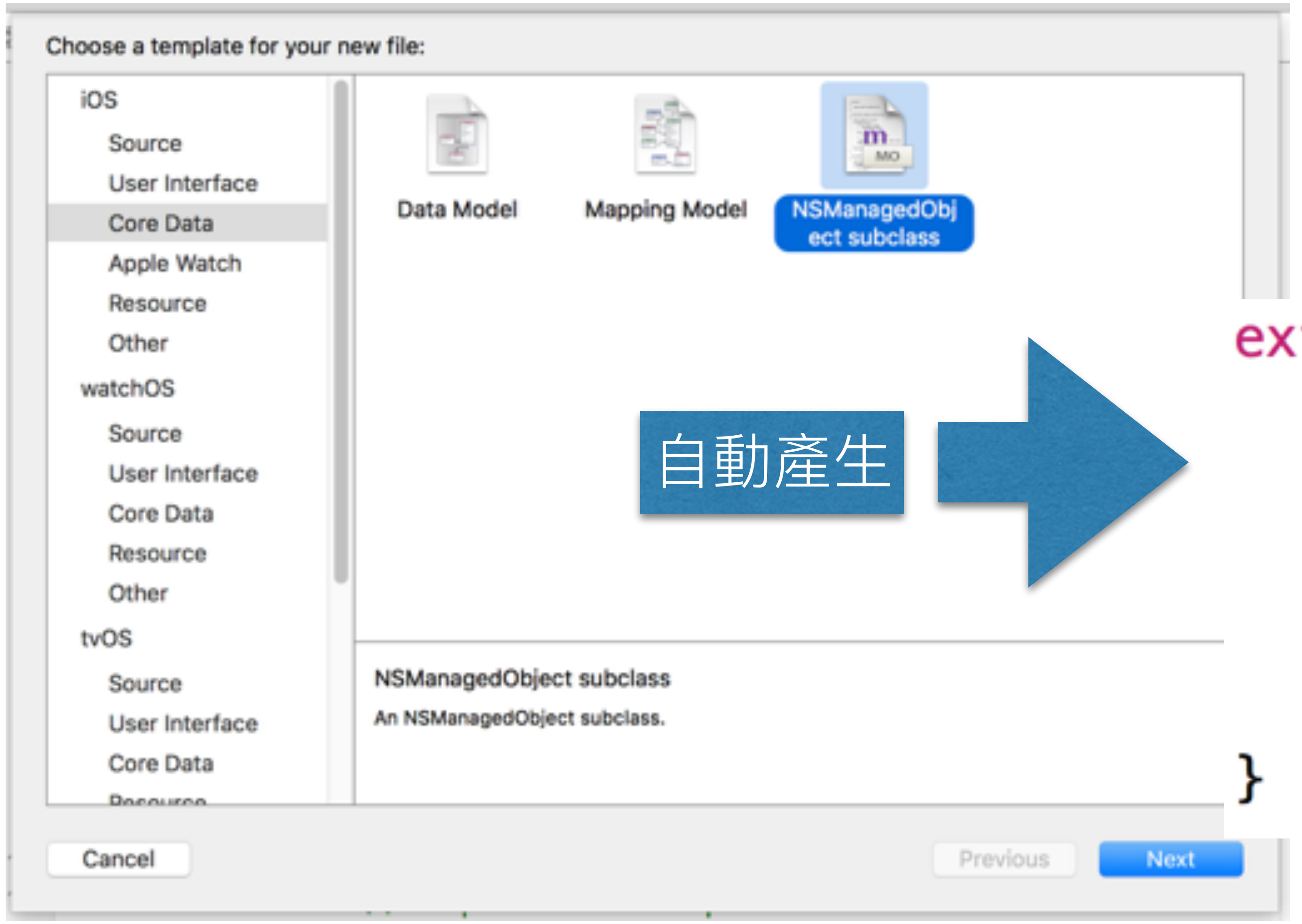
Relationships

Relationship	Destination	Inverse

Fetched Properties

Fetched Property	Predicate

entity class



NSManagedObject subclass

An NSManagedObject subclass.

自動產生

```swift
extension Book {

    @NSManaged var author: String?
    @NSManaged var isbn: String?
    @NSManaged var name: String?
    @NSManaged var price: NSNumber?

}
```

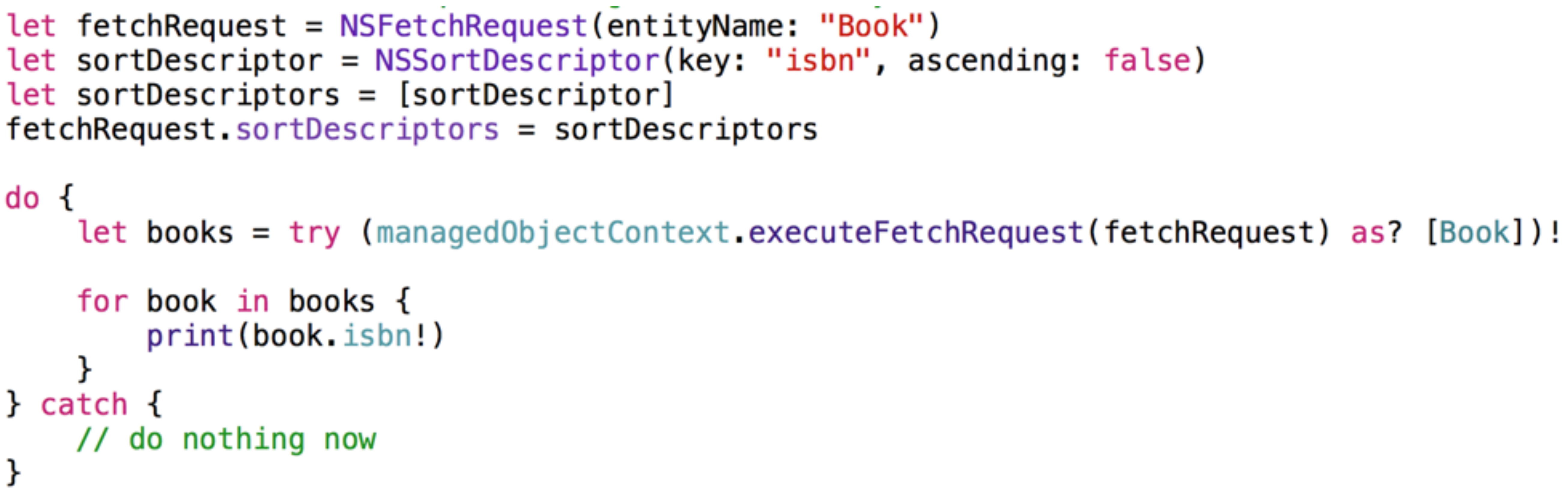
```swift
//  Choose "Create NSManagedObject Subclass…" from the Core Data editor menu
//  to delete and recreate this implementation file for your updated model.
//
```

do not modify the class

insert

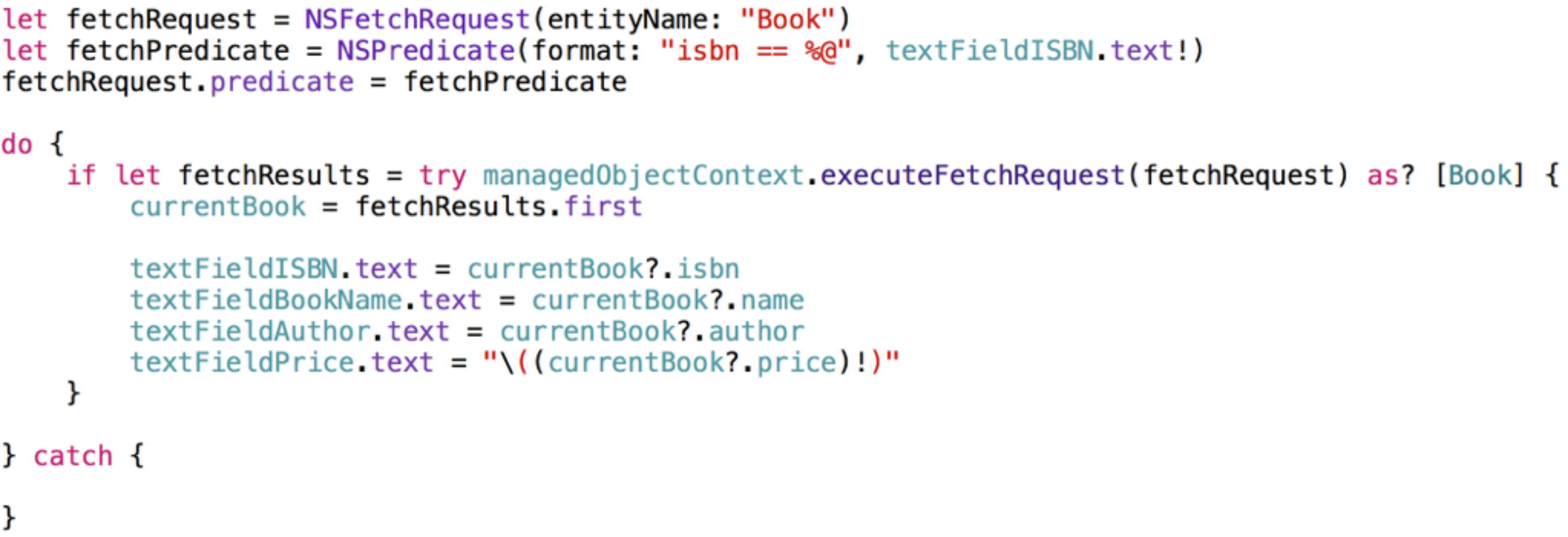
```
let saveBook = NSEntityDescription.insertNewObjectForEntityForName("Book", inManagedObjectContext:
    self.managedObjectContext) as! Book
saveBook.isbn = textFieldISBN.text
saveBook.name = textFieldBookName.text
saveBook.author = textFieldAuthor.text
saveBook.price = Int(textFieldPrice.text!)

appDelegate.saveContext()
```

query all



```
let fetchRequest = NSFetchRequest(entityName: "Book")
let sortDescriptor = NSSortDescriptor(key: "isbn", ascending: false)
let sortDescriptors = [sortDescriptor]
fetchRequest.sortDescriptors = sortDescriptors

do {
    let books = try (managedObjectContext.executeFetchRequest(fetchRequest) as? [Book])!

    for book in books {
        print(book.isbn!)
    }
} catch {
    // do nothing now
}
```

query 1



```
let fetchRequest = NSFetchRequest(entityName: "Book")
let fetchPredicate = NSPredicate(format: "isbn == %@", textFieldISBN.text!)
fetchRequest.predicate = fetchPredicate

do {
    if let fetchResults = try managedObjectContext.executeFetchRequest(fetchRequest) as? [Book] {
        currentBook = fetchResults.first

        textFieldISBN.text = currentBook?.isbn
        textFieldBookName.text = currentBook?.name
        textFieldAuthor.text = currentBook?.author
        textFieldPrice.text = "\((currentBook?.price)!)"
    }

} catch {

}
```

update

```swift
currentBook?.isbn = textFieldISBN.text
currentBook?.name = textFieldBookName.text
currentBook?.author = textFieldAuthor.text
currentBook?.price = Int(textFieldPrice.text!)

appDelegate.saveContext()
```

delete

```
managedObjectContext.deleteObject(currentBook!)
appDelegate.saveContext()

clearUI()
```

進階資料存取

SQLite, iCloud

~ END ~

https://www.alphacamp.co