
SIGMA
Software


SIGNL
Software
UNIVERSITY


SIGMA
Software
UNIVERSITY

ЛЕКЦІЯ 18

“Робота з файловою системою”

SIGMA SOFTWARE
UNIVERSITY

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE
UNIVERSITY

NSCoder

The `NSCoder` abstract class declares the interface used by concrete subclasses to transfer objects and other values between memory and some other format.

Your application can use an archive as the storage medium of your data model. Instead of designing (and maintaining) a special file format for your data, you can leverage Cocoa's archiving infrastructure and store the objects directly into an archive.

Для підтримки архівації об'єкт має відповідати `NSCoding` протоколу

NSCoding Protocol

OBJ-C

```
- (instancetype)initWithCoder:(NSCoder *)aDecoder
{
    self = [self initWithDictionary:nil];
    if (self) {
        _name = [aDecoder decodeObjectForKey:@"Name"];
        _capital = [aDecoder decodeObjectForKey:@"Capital"];
        _nationality = [aDecoder decodeObjectForKey:@"Nationality"];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder *)aCoder
{
    [aCoder encodeObject:_name forKey:@"Name"];
    [aCoder encodeObject:_capital forKey:@"Capital"];
    [aCoder encodeObject:_nationality forKey:@"Nationality"];
}
```

NSCoding Protocol

```swift
class Product: NSObject, NSCoding {
    var title:String
    var price:Double
    var quantity:Int
    enum Key:String {
        case title = "title"
        case price = "price"
        case quantity = "quantity"
    }
    init(title:String,price:Double, quantity:Int) {
        self.title = title
        self.price = price
        self.quantity = quantity
    }
    func encode(with aCoder: NSCoder) {
        aCoder.encode(title, forKey: Key.title.rawValue)
        aCoder.encode(price, forKey: Key.price.rawValue)
        aCoder.encode(quantity, forKey: Key.quantity.rawValue)
    }
    convenience required init?(coder aDecoder: NSCoder) {
        let price = aDecoder.decodeDouble(forKey: Key.price.rawValue)
        let quantity = aDecoder.decodeInteger(forKey: Key.quantity.rawValue)
        guard let title = aDecoder.decodeObject(forKey: Key.title.rawValue) as? String else { return nil }
        self.init(title:title,price:price,quantity:quantity)
    }
```

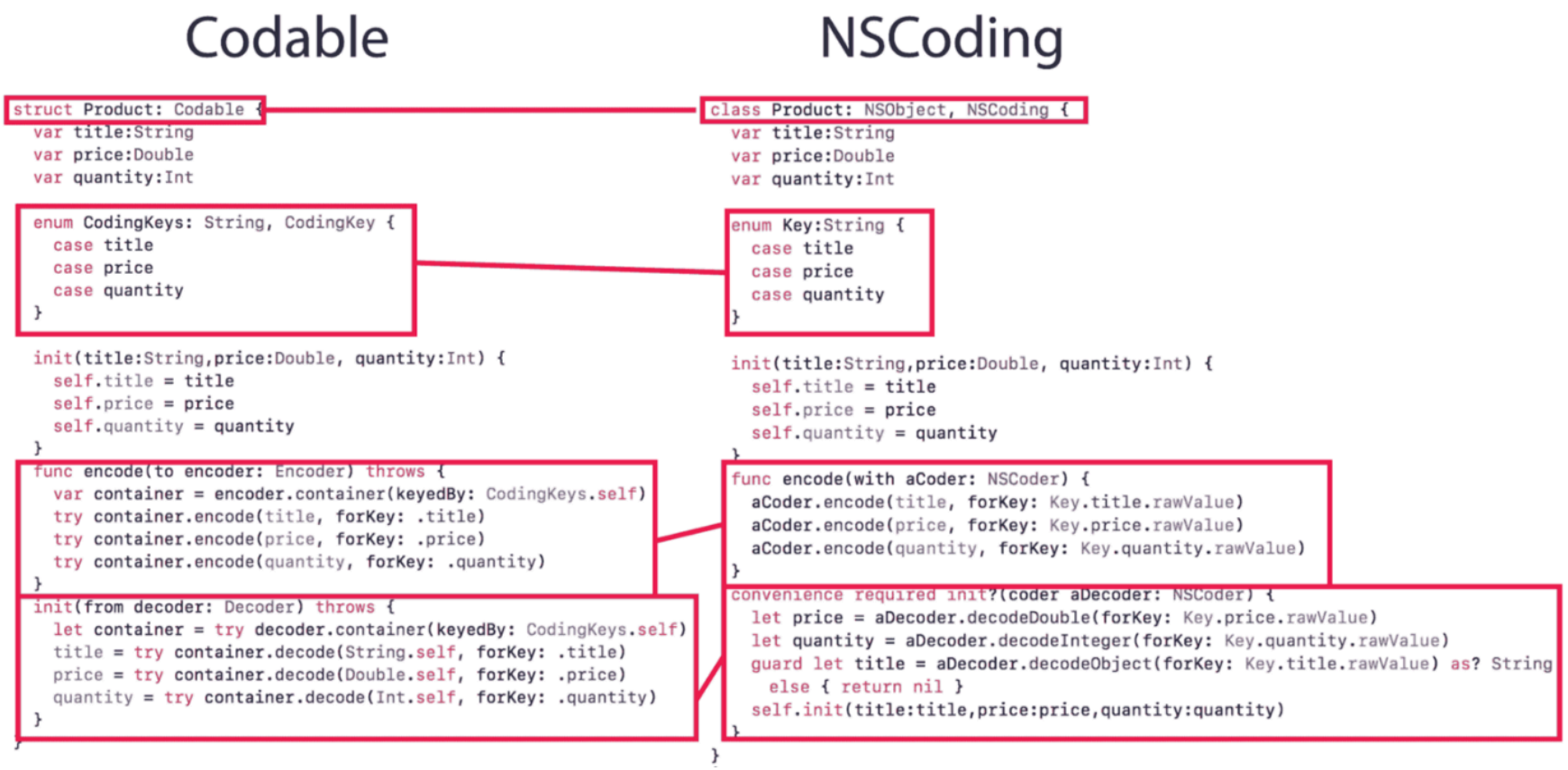
https://medium.com/if-let-swift-programming/migrating-to-codable-from-nscoding-ddc2585f28a4

Codable Protocol

```
struct Product: Codable {
    var title:String
    var price:Double
    var quantity:Int
    enum CodingKeys: String, CodingKey {
        case title
        case price
        case quantity
    }
    init(title:String,price:Double, quantity:Int) {
        self.title = title
        self.price = price
        self.quantity = quantity
    }
    func encode(to encoder: Encoder) throws {
        var container = encoder.container(keyedBy: CodingKeys.self)
        try container.encode(title, forKey: .title)
        try container.encode(price, forKey: .price)
        try container.encode(quantity, forKey: .quantity)
    }
    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)
        title = try container.decode(String.self, forKey: .title)
        price = try container.decode(Double.self, forKey: .price)
        quantity = try container.decode(Int.self, forKey: .quantity)
    }
}
```

https://medium.com/if-let-swift-programming/migrating-to-codable-from-nscoding-ddc2585f28a4

Codable vs NSCoding



https://medium.com/if-let-swift-programming/migrating-to-codable-from-nscoding-ddc2585f28a4

Codable vs NSCoding

Codable (Manual)

```
struct Product: Codable {
  var title:String
  var price:Double
  var quantity:Int

  enum CodingKeys: String, CodingKey {
    case title
    case price
    case quantity
  }

  init(title:String,price:Double, quantity:Int) {
    self.title = title
    self.price = price
    self.quantity = quantity
  }
  func encode(to encoder: Encoder) throws {
    var container = encoder.container(keyedBy: CodingKeys.self)
    try container.encode(title, forKey: .title)
    try container.encode(price, forKey: .price)
    try container.encode(quantity, forKey: .quantity)
  }
  init(from decoder: Decoder) throws {
    let container = try decoder.container(keyedBy: CodingKeys.self)
    title = try container.decode(String.self, forKey: .title)
    price = try container.decode(Double.self, forKey: .price)
    quantity = try container.decode(Int.self, forKey: .quantity)
  }
}
```

Codable (Auto generated)

```
struct Product: Codable {
  var title:String
  var price:Double
  var quantity:Int
  init(title:String,price:Double, quantity:Int) {
    self.title = title
    self.price = price
    self.quantity = quantity
  }
}
```

SIGMA SOFTWARE UNIVERSITY

https://medium.com/if-let-swift-programming/migrating-to-codable-from-nscoding-ddc2585f28a4

NSCoder subclasses

- NSArchiver (deprecated)
- NSUnarchiver (deprecated)
- NSKeyedArchiver
- NSKeyedUnarchiver

NSKeyedArchiver

OBJ-C

```
- (void)save
{
    NSData *data = [NSKeyedArchiver archivedDataWithRootObject:self];
    if (data != nil) {
        [[NSUserDefaults standardUserDefaults] setObject:data forKey:@"someKey"];
        [[NSUserDefaults standardUserDefaults] synchronize];
    }
}

- (instancetype)load
{
    NSData *data = [[NSUserDefaults standardUserDefaults] objectForKey:@"someKey"];
    if (data != nil) {
        CountryModel *country = [NSKeyedUnarchiver unarchiveObjectWithData:data];
        return country;
    }
    return nil;
}
```

SIGMA SOFTWARE UNIVERSITY

NSKeyedArchiver

```
func retrieveProducts() -> [Product]? {
    let unarchivedData = NSKeyedUnarchiver.unarchiveObject(withFile: productsFile.path)
    //Work with Codable
    if let data = unarchivedData as? Data {
        do {
            let decoder = PropertyListDecoder()
            let products = try decoder.decode([Product].self, from: data)
            return products
        } catch {
            print("Retrieve Failed")
            return nil
        }
    }
        // Work with NSCoding
    else if let products = unarchivedData as? [Product] {
        return products
    } else {
        return nil
    }
}
```

SIGMA SOFTWARE UNIVERSITY

https://medium.com/if-let-swift-programming/migrating-to-codable-from-nscoding-ddc2585f28a4

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

Property list (.plist)

Property lists organize data into named values and lists of values using several object types. These types give you the means to produce data that is meaningfully structured, transportable, storable, and accessible, but still as efficient as possible.

Property lists часто використовуються в програмах як на OS X так і на iOS.

Plist має стандартну XML стурктуру.

Plist. Data types

- Dictionary
- Array
- String
- Date
- Number

Plist. XML representation

```
<?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>Lincoln</key>
    <dict>
        <key>DOB</key>
        <date>1809-02-12T09:18:00Z</date>
        <key>Name</key>
        <string>Abraham Lincoln</string>
        <key>Scores</key>
       <array>
            <integer>8</integer>
            <real>4.9000000953674316</real>
       </array>
    </dict>
</dict>
</plist>
```

SIGMA SOFTWARE UNIVERSITY

Plist. Serialization

NSPropertyListSerialization class.

The `NSPropertyListSerialization` class provides methods that convert property list objects to and from several serialized formats.

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

NSUserDefaults

The NSUserDefaults class provides a programmatic interface for interacting with the defaults system. The defaults system allows an application to customize its behavior to match a user's preferences.

Створюється по замовчуванню на кожного користувача

NSUserDefaults використовує .plist файл для зберігання інформації.

Значення що повертаються з NSUserDefaults є *immutable*, навіть якшо значення які були записані були mutable.

NSUserDefaults. Data types

- NSDictionary
- NSArray
- NSString
- NSData
- NSDate
- NSNumber

NSUserDefaults. Access

- standartUserDefaults – singleton об’єкт для доступу до даних
- resetStandartUserDefaults – синхронізує будь які зміни і знищує їх.
- synchronize – синхронізує будь-які зміни і записує їх до файлу.

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

Робота з файловою системою

NSCoder

Property list (.plist)

NSUserDefaults

File system in iOS

SIGMA SOFTWARE UNIVERSITY

Файлова система в iOS. Огляд

Файлова система в iOS основана на UNIX файловій системі.

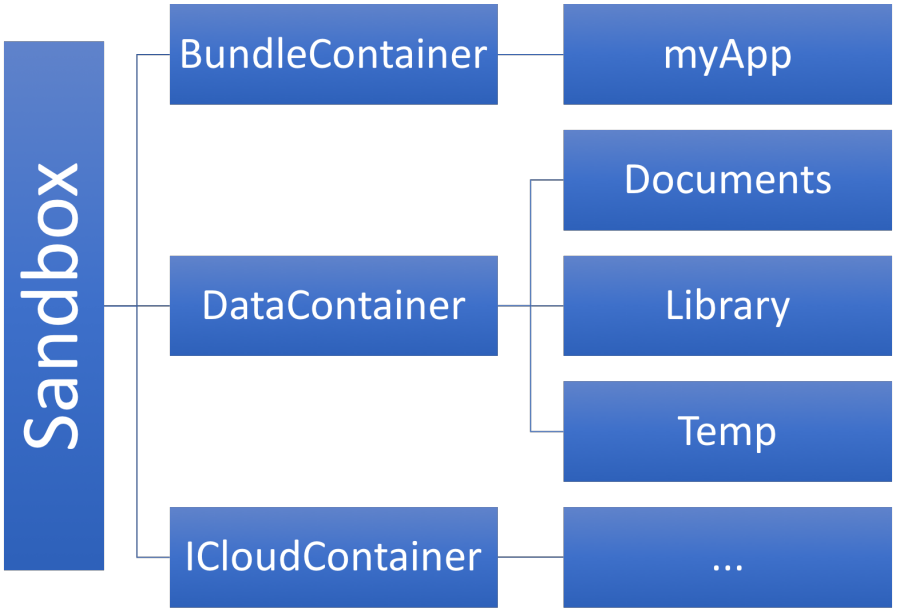
«The iOS file system is geared toward apps running on their own. To keep the system simple, users of iOS devices do not have direct access to the file system and apps are expected to follow this convention.»

"An iOS app’s interactions with the file system are limited mostly to the directories inside the app’s sandbox. During installation of a new app, the installer creates a number of containers for the app. Each container has a specific role.»

File system in iOS.
Кожна програма як острів

The bundle container holds the app's bundle, whereas the data container holds data for both the application and the user.

Додаток може запитати доступ до додаткових контейнерів—наприклад, iCloud container—під час роботи.

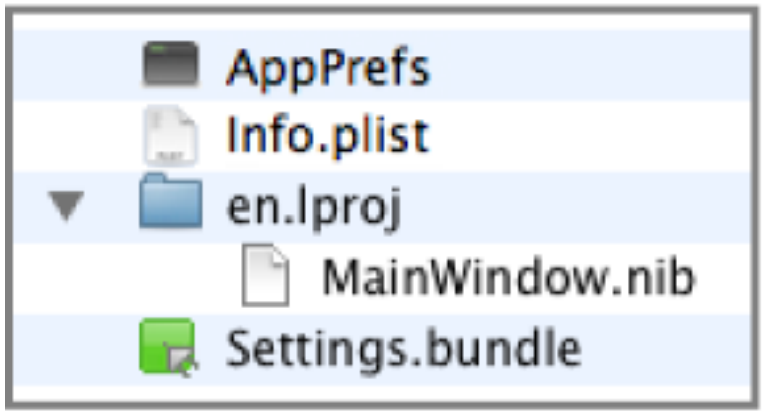


SIGMA SOFTWARE UNIVERSITY

Bundle контейнер

A bundle це папка в файловій системі яка тримає в собі ресурси і виконавчий код та всі ресурси такі як малюнки та звуки.

AppName`.app` — is the app's bundle. Ця папки тримає в собі програму і всі її ресурси



SIGMA SOFTWARE UNIVERSITY

Доступ до bundle контейнера

An NSBundle object helps you access the code and resources in a bundle directory on disk.

Each application has a main bundle, which is the bundle that contains the application code. To get an application's main bundle, call the class method *mainBundle*.

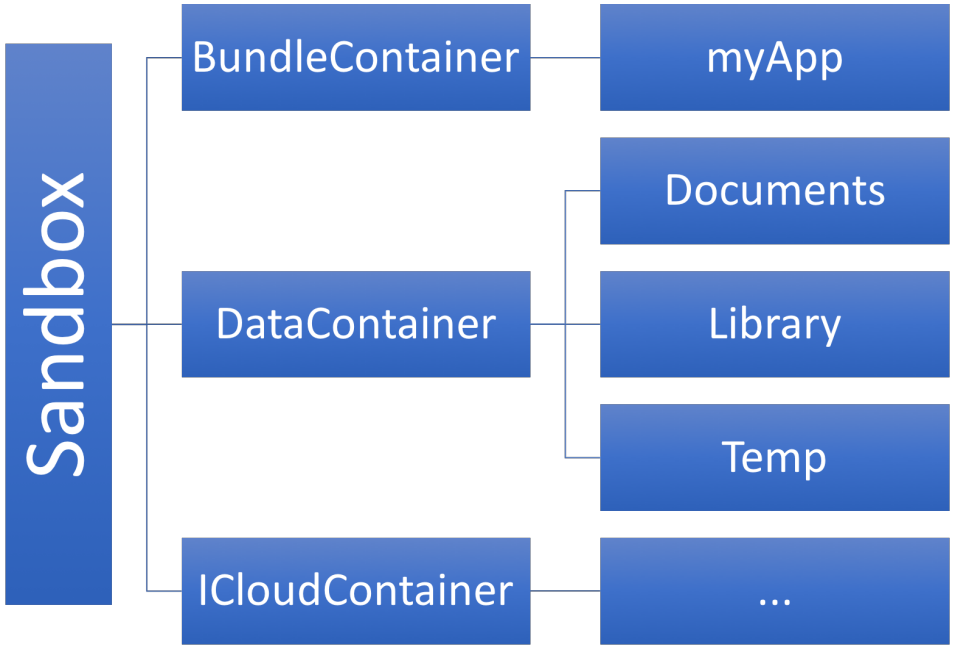
Other `NSBundle` methods return paths to bundle resources when given a filename, extension, and (optionally) a bundle subdirectory. After you have a path to a resource, you can load it into memory using the appropriate class.

Data Container

For security purposes, an iOS app has limited a number of places where it can write its data.

When an app is installed on a device (either from iTunes or the App Store), the system creates a number of containers for the app.

These containers represent the universe for that app and contain everything the app can access directly.



SIGMA SOFTWARE UNIVERSITY

iOS Стандартні папки

1. Documents/

Use this directory to store user-generated content. The contents of this directory can be made available to the user through file sharing; therefore, his directory should only contain files that you may wish to expose to the user.

Вміст цієї папки зберігається на iTunes.

OBJ-C

```
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES).firstObject;
```

iOS Стандартні папки

2. Documents/Inbox

Use this directory to access files that your app was asked to open by outside entities. Specifically, the Mail program places email attachments associated with your app in this directory. Document interaction controllers may also place files in it.
Your app can read and delete files in this directory but cannot create new files or write to existing files.

Вміст цієї папки зберігається на iTunes.

iOS Стандартні папки

3. tmp/

Use this directory to write temporary files that do not need to persist between launches of your app. Your app should remove files from this directory when they are no longer needed; however, the system may purge this directory when your app is not running.

Вміст цієї папки і підпапок не зберігаються на iTunes.

OBJ-C

```
NSTemporaryDirectory();
```

SIGMA SOFTWARE UNIVERSITY

iOS Стандартні папки

4. Library/

This is the top-level directory for any files that are not user data files. You typically put files in one of several standard subdirectories. iOS apps commonly use the `Application Support` and `Caches` subdirectories; however, you can create custom subdirectories.

Вміст цієї папки (крім Caches підпаки) не зберігаються на iTunes.

OBJ-C

```
NSSearchPathForDirectoriesInDomains(NSLibraryDirectory, NSUserDomainMask, YES).firstObject;
NSSearchPathForDirectoriesInDomains(NSApplicationSupportDirectory, NSUserDomainMask, YES).firstObject;
```

Доступ до контейнера даних

«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.»

«The `NSFileManager` class provides convenient access to a shared file manager object that is suitable for most types of file-related manipulations. defaultManager»

Якшо ви плануєте використовувати файловий менеджер інтерактивно - наприклад якщо ви хочете долучати делегат - створіть новий екземпляр об’єкта.


SIGMA
Software
UNIVERSITY