SCIENCES SORBONNE UNIVERSITÉ

Handling persistent data

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

You need to save an retrieve data…

- Several exercices would have benefit from such a feature
  - iSouvenir
  - MyActivities
  - Asteroids
  - etc.

Benefits, maintaining your App's context

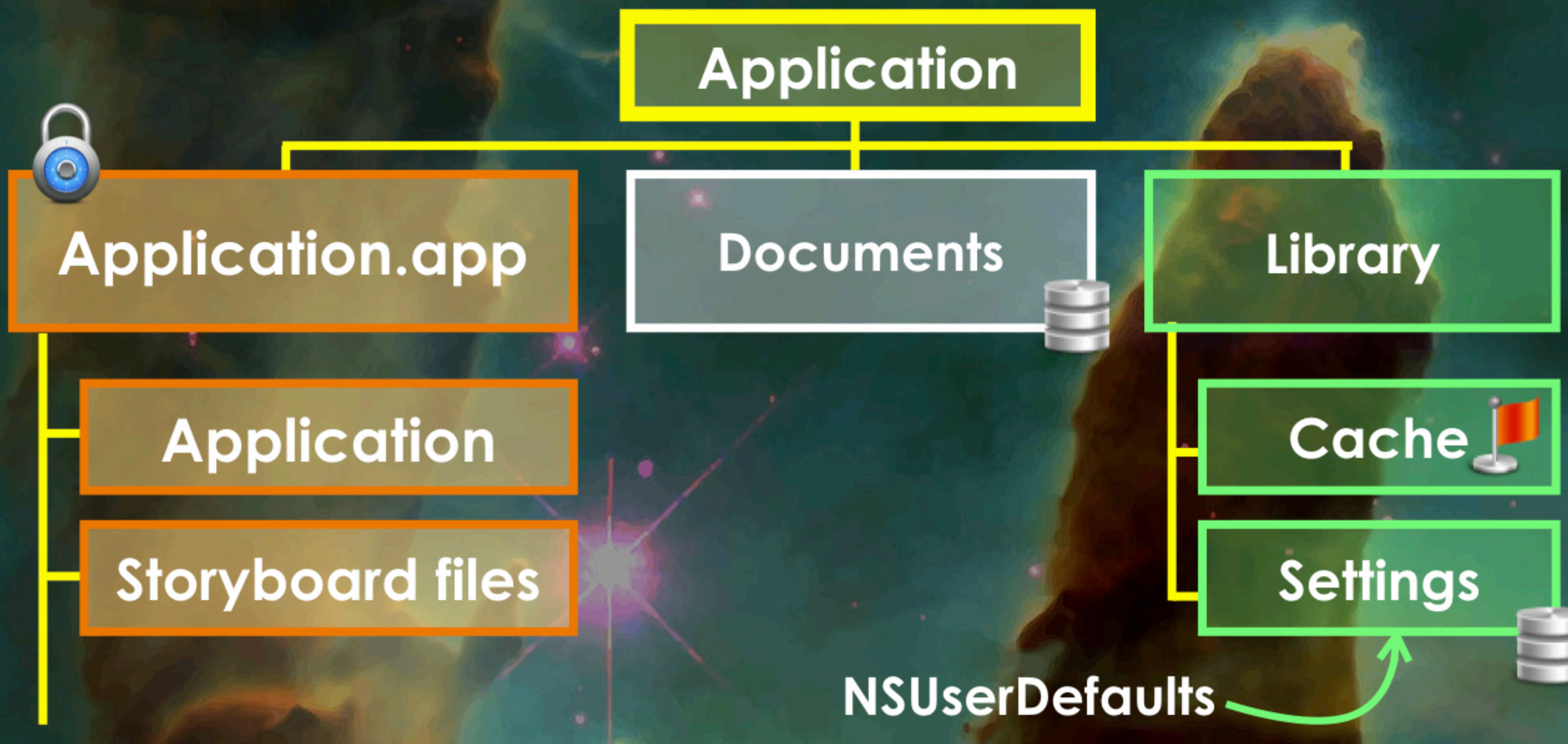
- Remember multitasking management
  - Saving context when going into background
  - Being killed by iOS
  - Retrieving this context when restarting

The filesystem in iOS

Applications are sandboxed

- Security & confidentiality vs practicability
  - **Main difference with Android**

File system's structure (for an App)



Principles

Transparent location in the device's file system

- Due to sandboxing
  - **Impossible to build an absolute path without help**

Several ways to store data

- Serialization
  - **iOS handles it**
- Data (small size?)
  - **Binary**
- SQlite
  - **Database**
  - **C API**
  - **Not presented**
- CoreData
  - **Sophisticated handling of containers**
  - **Not presented**

FileManager (handling paths)

A way to handle your App's file system

- Unix-like file system… are we surprised?
- The shared file manager
  - **Property default**
- Some useful features

```
var homeDirectoryForCurrentUser: URL { get }
func NSHomeDirectory() -> String

var temporaryDirectory: URL { get }
func NSTemporaryDirectory() -> String
```

- A way to build a path
  - **Get home directory as a string**

```
func NSSearchPathForDirectoriesInDomains
                (_ directory: FileManager.SearchPathDirectory,
                 _ domainMask: FileManager.SearchPathDomainMask,
                 _ expandTilde: Bool) -> [String]
```

  - **Concatenate with a relative path (start with «/»)**

FileManager

Write/read data

```
func createFile(atPath path: String,
                contents data: Data?,
                attributes attr: [FileAttributeKey : Any]? = nil) -> Bool

func contents(atPath path: String) -> Data?
```

Other useful functions

```
func removeItem(at URL: URL) throws
func removeItem(atPath path: String) throws

func contentsOfDirectory(atPath path: String) throws -> [String]
```

More at RTFM!

Persistency (principles)

Object serialization

- To be handled for all classes to be stored

NSCoding protocol

- Decoding…
  - **We already met**

```
init?(coder aDecoder: NSCoder)
```

- Encoding

```
func encode(with aCoder: NSCoder)
```

Caution

- Invoke coder/decoder for super…
  - **Unless the object is on top of the hierarchy**

Persistency (in practice)

NSKeyedArchiver & NSKeyedUnarchiver

- Rely on NSCoding
  - Handle invocation of the appropriates primitives
  - handle complex structured classes

Exemple on a MyObject class

- Encoding

```
let myData = MyObject()
...
let coder = NSKeyedArchiver(requiringSecureCoding: false)
coder.encode(myData, forKey: NSKeyedArchiveRootObjectKey)
```

- Decoding

```
do {
    let decoder = try NSKeyedUnarchiver(forReadingFrom: fetchData)
    decoder.requiresSecureCoding = false
    let myData = try NSKeyedUnarchiver.unarchiveTopLevelObjectWithData
                                    (fetchData) as? MyObject
} catch {
    print("I have a problem")
}
```

Persistency (from Data)

Read & Write Data from a file

```
init(contentsOf url: URL, options: Data.ReadingOptions = default) throws

func write(to url: URL, options: Data.WritingOptions = default) throws
```

Not a standalone method

- Related to serialization
- Notice that default serialisation is provided
  - String
  - Double
  - Bool
  - CGFloat
  - NSArray (when elements are serializables)
  - NSDictionary (when elements are serializables)
  - etc.

Miscellaneous things

iCloud storage?

- Included in FIleManager
  - **Dedicated methods**
  - **Use of a (readonly) ubiquityIdentityToken property**

NSSecureCoding

- Extended version of NSCoding
  - **Robust against object substitution attacks**
  - **Similar principles but a way to handle errors**

Creating a directory (from FileManager)

```
func url(for directory: FileManager.SearchPathDirectory,
         in domain: FileManager.SearchPathDomainMask,
         appropriateFor url: URL?,
         create shouldCreate: Bool) throws -> URL
```

As a conclusion…

- Very standard
- Constraints due to data protection
  - Sandboxing
  - No access to data from other applications
    - In fact, «..» works…

      … but forget the AppStore
- You are almost ready for it!

As a conclusion…

- Very standard
- Constrain… to data protection
  - Sandboxing
  - No access t…
    - In fact, «..» w…

One more thing…

Changes in iOS12, be aware of backward compatibility (discussed in a next video)

- You are almost ready for it!