Persistance des données

Longue vie aux données !

Persistance des données

- Property Lists
- Les préférences
- Accès au système de fichiers
- Core Data

Property Lists

Property Lists

- Fichiers .plist
- Stocke des objets sérialisés
- Permet de stocker des tableaux ou des dictionnaires d’objets “standards”
  - NSData, NSString (String), NSNumber (Int/Double/Float), NSDate, NSArray (Array), NSDictionary
- Il faut savoir lire et écrire sur le disque !

Property Lists

- NSArray ou NSDictionary
- `convenience init?(contentsOfURL url: NSURL)`
- `func writeToURL(url: NSURL, atomically: Bool) -> Bool`

Les préférences

- NSUserDefaults
  - Utilisé pour stocker des petites quantités d’informations
  - Basé sur un fichier Property List
  - Stocké dans le dossier Library/Preferences de l’app

- Permet de stocker :
  - des types courants (Int / Float / Double / Bool / NSURL)
  - des objets compatibles plist (NSData, NSDate, NSArray (Array), NSDictionary (Dictionary)
- Garde un cache en mémoire pour les réglages
  - Cache synchronisé «régulièrement» avec le système de fichier
  - Ou manuellement via la méthode synchronize

- `class func standardUserDefaults() -> NSUserDefaults`
- `func setBool(value: Bool, forKey defaultName: String)`
- `func setFloat(value: Float, forKey defaultName: String)`
- `...`
- `func objectForKey(defaultName: String) -> AnyObject?`
- `func boolForKey(defaultName: String) -> Bool`
- `...`

```
let prefs = NSUserDefaults.standardUserDefaults()
prefs.setBool(true, forKey: "UserAgreed")
prefs.setObject("Ludovic", forKey: "firstName")

let firstName = prefs.stringForKey("firstName")
print(firstName)
```

```
Optional("Ludovic")
```

L’archivage

L’archivage

- Permet d’écrire des objets sur le disque
- Méthode utilisée par Xcode pour écrire vos UI/Storyboards
- Permet d’enregistrer n’importe quel type de graphe d’objet
  - Du moment que les objets se conforment au protocole NSCoding
    - Ajout de méthodes dans la classe.
    - Classe qui hérite de NSObject pour le bon fonctionnement

NSCoding

- Méthodes obligatoires :
- `func encodeWithCoder(aCoder: NSCoder)`
- `init(coder aDecoder: NSCoder)`

NSCoding

```
@objc func encodeWithCoder(aCoder: NSCoder) {

    aCoder.encodeObject(age, forKey: "age")
    aCoder.encodeObject(name, forKey: "name")
    aCoder.encodeObject(size, forKey: "size")
    aCoder.encodeObject(gender, forKey: "gender")
    aCoder.encodeObject(childrens, forKey: "childrens")
}

@objc required init(coder aDecoder: NSCoder) {

    age = aDecoder.decodeObjectForKey("age") as! Int
    name = aDecoder.decodeObjectForKey("name") as! String
    size = aDecoder.decodeObjectForKey("size") as! Float
    gender = aDecoder.decodeObjectForKey("gender") as! String
    childrens = aDecoder.decodeObjectForKey("childrens") as! [Human]

}
```

L’archivage

- NSKeyedArchiver
  - Archive un graphe d’objet dans du NSData
  - `class func archivedDataWithRootObject(rootObject: AnyObject) -> NSData`
- NSKeyedUnarchiver
  - Désarchive un graphe d’objet d’un objet NSData
  - `class func unarchiveObjectWithData(data: NSData) -> AnyObject?`
- Il faut savoir lire et écrire du NSData sur le disque !

L’archivage

```
func archive() -> NSData {

    let myRootObject = NSArray()
    let data = NSKeyedArchiver.archivedDataWithRootObject(myRootObject)

    return data
}

func unarchiveWithData(data: NSData) {

    let unarchivedObject = NSKeyedUnarchiver.unarchiveObjectWithData(data)
    //Do something with the unarchived data
}
```

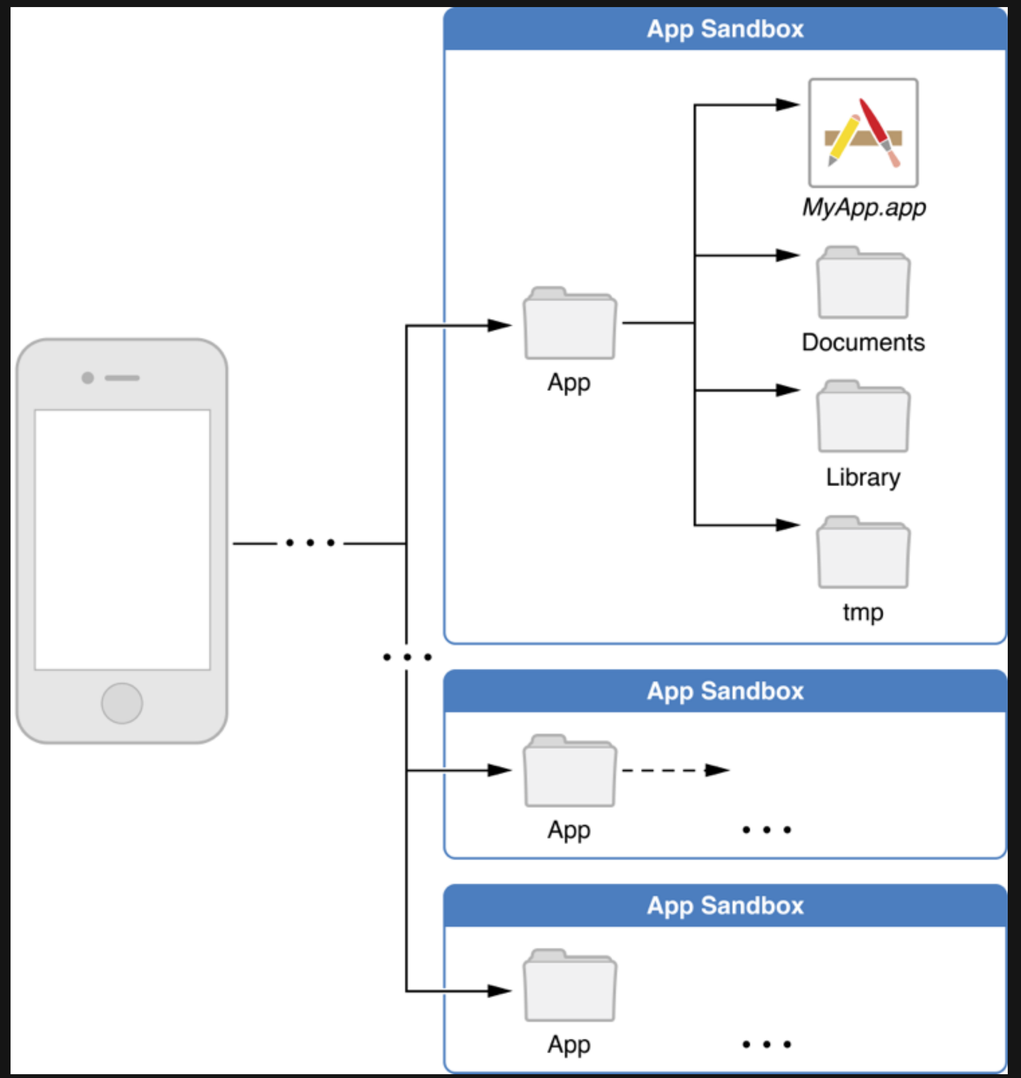
Accès au système de fichiers

Comment écrire sur le disque ?

- Système UNIX
  - Utilisation de chemins UNIX classiques
- Les apps sont en sandbox
  - On ne peut donc pas écrire n’importe où !
  - Augmente la sécurité et la confidentialité

Comment écrire sur le disque ?

- Dans la sandbox de l'app :
  - Bundle de l'app
    - Contient l'app et ses ressources
    - Bundle signé ! Si modifié, l'app ne fonctionnera plus.
  - Dossier Documents
    - Pour stocker des données importantes
  - Et d'autres...



Comment écrire sur le disque ?

1. Récupérer le chemin d'un dossier de base (Documents, Caches, etc.)
2. Ajouter le chemin souhaité pour son fichier
3. Écrire le fichier à l'emplacement souhaité

```
func saveToDisk(data: NSData) {

    let fileManager = NSFileManager()

    var URL =
fileManager.URLsForDirectory(.DocumentDirectory,
inDomains:.UserDomainMask).first as! NSURL
    URL = URL.URLByAppendingPathComponent("myFile")
    data.writeToURL(URL, atomically: true)
}
```

Comment lire sur le disque ?

1. Récupérer le chemin d'un dossier de base (Documents, Caches, etc.)
2. Ajouter le chemin souhaité pour son fichier
3. Lire le fichier à l'emplacement souhaité

```
func readFromDisk() -> NSData? {

    let fileManager = NSFileManager()

    var URL = fileManager.URLsForDirectory(.DocumentDirectory,
inDomains: .UserDomainMask).first as! NSURL
    URL = URL.URLByAppendingPathComponent("myFile")

    if fileManager.fileExistsAtPath(URL.absoluteString!) {

        let data = NSData(contentsOfURL: URL)

        return data
    }

    return nil
}
```

- Pour plein d’autres possibilités sur le système de fichier, la documentation de NSFileManager est un bon point de départ !
  - Créer des dossiers, savoir si un fichier existe, effacer un fichier, etc.

Core Data

- Framework généraliste pour la persistance des données
- Backend SQLite ou XML
- Mono utilisateur
- Fonctionnalités avancées (prédicat, schéma de migration…)

- Plus d’infos sur Core Data dans un chapitre dédié !

Pour aller plus loin…

- NSUserDefaults Documentation
- Archives and Serializations Programming Guide