Persistance des données

Longue vie aux données !

Persistance des données

- Les préférences
- L’archivage
- Accès au système de fichiers
- Core Data

Les préférences

- UserDefaults
  - 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 (String / Int / Float / Double / Bool / URL)
  - des objets compatibles plist (Data, Date, Array, Dictionary)
- Garde un cache en mémoire pour les réglages
  - Cache synchronisé automatiquement avec le système de fichier

- `class var standard: UserDefaults`
- `func set(_ value: Bool, forKey defaultName: String)`
- `func set(_ value: Float, forKey defaultName: String)`
- `...`
- `func object(forKey: String) -> Any?`
- `func bool(forKey: String) -> Bool`
- `...`

```swift
let prefs = UserDefaults.standard
prefs.set(true, forKey: "UserAgreed")
prefs.set("Ludovic", forKey: "firstName")

let firstName = prefs.string(forKey: "firstName")
print(firstName)
```

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

```swift
@propertyWrapper
struct UserDefault<T> {

    let key: String
    let defaultValue: T
    let defaults: UserDefaults = .standard

    init(_ key: String, defaultValue: T) {
        self.key = key
        self.defaultValue = defaultValue
    }

    var wrappedValue: T {
        get {
            return defaults.object(forKey: key) as? T ?? defaultValue
        }
        set {
            defaults.set(newValue, forKey: key)
        }
    }
}
```

```swift
struct UserDefaultsConfig {

    @UserDefault("has_seen_app_introduction", defaultValue: false)
    static var hasSeenAppIntroduction: Bool
}
```


```swift
@AppStorage("userAgreedTerms") var agreed: Bool = false
```

L’archivage

Codable

- Permet de transformer des objets en Data
- Permet d’enregistrer n’importe quel type de graphe d’objet
  - Du moment que les objets se conforment au protocole Codable
    - Adoption transparente la plupart du temps

Codable

- Codable
  - Protocoles Encodable et Decodable
  - Les types Swift de bases ne demandent pas d’adaptations
  - Permet l’utilisation de JSONEncoder/Decoder ou de PropertyListEncoder/Decoder

Codable

```
func archive(object: TypeDeObject) -> Data? {
    let encoder = JSONEncoder()
    let data = try? encoder.encode(object)
    return data
 }

func parse(data: Data) {

    if let object = try? JSONDecoder().decode(TypeDeObject.self, from: data) {
         //Use the object
    }
}
```

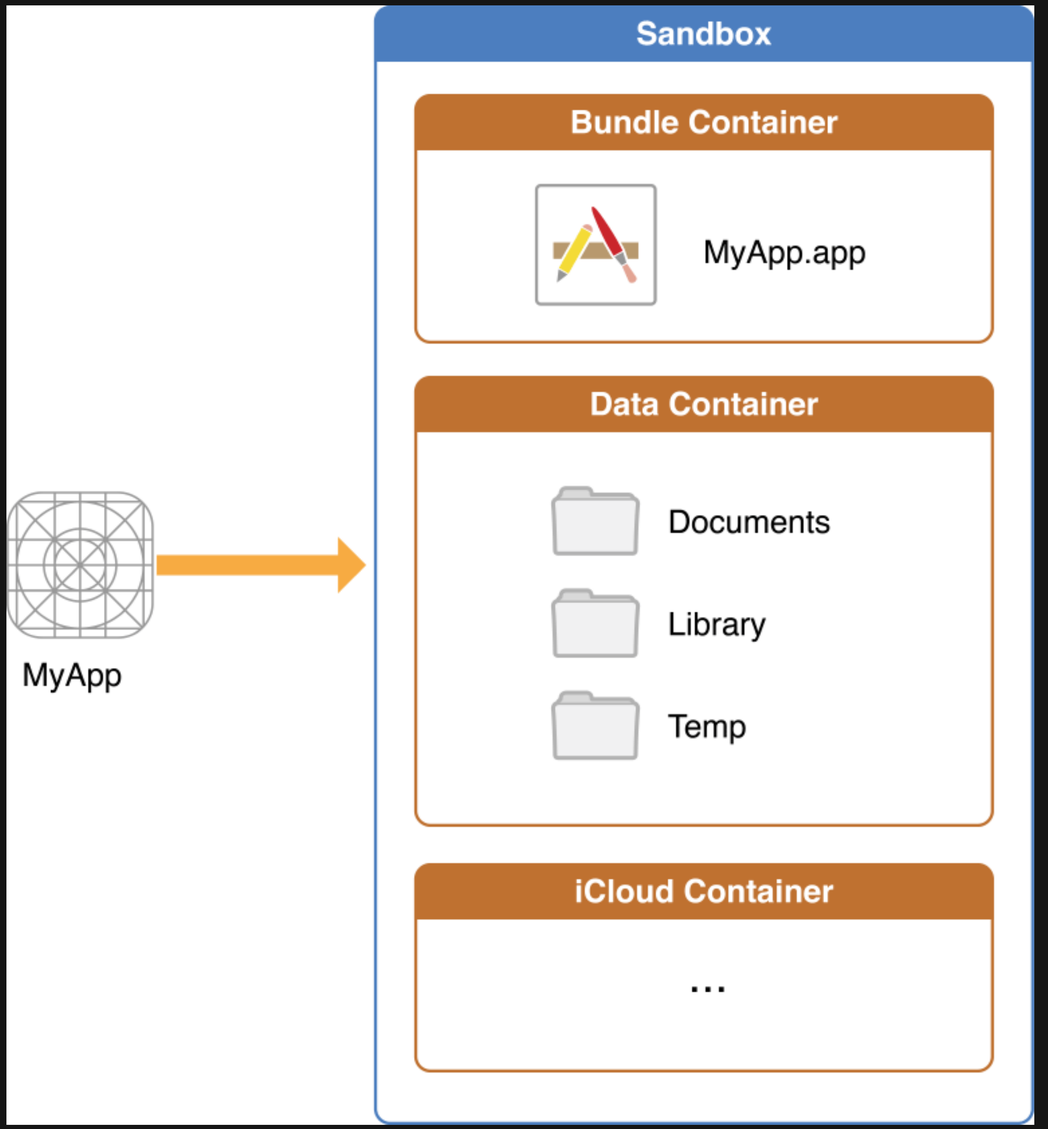
Accès au système de fichiers

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: Data) {

    let fm = FileManager.default
    guard let baseURL = fm.urls(for: .documentDirectory, in:.userDomainMask).first else { return }
    let fullURL = baseURL.appendingPathComponent("myFile")
    try? data.write(to: fullURL)
}
```

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() -> Data? {

    let fm = FileManager.default
    guard let baseURL = fm.urls(for: .documentDirectory, in:.userDomainMask).first else { return nil }
    let fullURL = baseURL.appendingPathComponent("myFile")

    if fm.fileExists(atPath: fullURL.path) {

        let data = try? Data(contentsOf: fullURL)
        return data
    }
   return nil
}
```

- Pour plein d’autres possibilités sur le système de fichier, la documentation de FileManager 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 par défaut
- Mono utilisateur
- Fonctionnalités avancées (prédicat, schéma de migration…)

Pour aller plus loin…

- UserDefaults Documentation
- Encoding and Decoding Custom Types