GENERAL ASSEMBLY

MOBILE DEVELOPMENT

FILES & PERSISTENCE: PART 1

William Martin
VP of Product, Floored

Angel X. Moreno
EIR, Developer

LEARNING OBJECTIVES

- Recognize the different types of persistence and their pros/cons.
- Implement user defaults, but recognize that storage here should be limited.
- Create property lists and save/read data from property lists.
- Discover the iOS folder structure and where we should store certain types of files.
- Use folder search within our applications to read and write files.
- Create, read, and write to flat files.

INTRO TO PERSISTENCE

WHY PERSIST?

- Saving data across sessions:
  - App settings
  - User credentials
  - Game high scores
  - etc.
- Sharing data between screens.

HOW TO PERSIST?

- There are many, many ways to persist data.
- There is no ‘right’ way.
- Choosing how to persist depends on several things:
  - What kind of data am I writing?
  - What kind of data am I reading?
  - Am I storing relations between things?
  - How persistent does my data need to be?

HOW TO PERSIST?

- There are several built-in options for persisting data in iOS:
  - This session, we'll discuss:
    - Flat files
    - Property lists
    - User defaults
  - Next session, we'll introduce:
    - Core Data
    - SQL

USER DEFAULTS

USER DEFAULTS

- A key-value store for storing app settings and other small, independent bits of data.
- Like a persistent Dictionary.
- Apple reference here.

USER DEFAULTS

- What kind of data am I writing?
  - Small bits of data and an associated key, stored one at a time
  - e.g. A String, a Number, a Boolean, a Dictionary, an Array.
- What kind of data am I reading?
  - Same as above, retrieved one at a time.
- Am I storing relations between things?
  - No.
- How persistent does my data need to be?
  - Persistent across app sessions, deleted when the app is deleted.

USER DEFAULTS

- Good for:
  - App settings
  - App state
  - Native integration with iOS App Preferences.
- Not good for:
  - Large data sets
  - Complex relations
  - Sensitive data
  - Caches

NSUSERDEFAULTS : STORING

```
let defaultsMgr = NSUserDefaults.standardUserDefaults()
defaultsMgr.setBool(true, forKey: “mySetting”)
```

NSUSERDEFAULTS : RETRIEVING

```
let defaultsMgr = NSUserDefaults.standardUserDefaults()
defaultsMgr.boolForKey(“mySetting”)  // == true
```

NSUSERDEFAULTS CODE ALONG

GROUP ASSIGNMENT

- When your application starts, create an array of athlete names, store this array using iOS user defaults.
- In the view controller, retrieve this list of names and print them out.
- Bonus 1: Print these names in a table view.
- Bonus 2: Add ability to add a persisted name to the list.

PROPERTY LISTS

PROPERTY LISTS

- Another form of key-value store.
- A bundle of text data (formatted as XML) stored in a text file.
- *.plist files.

PROPERTY LISTS

- What kind of data am I writing?
  - Small-ish sets of data (a few hundred KB) retrieved file-at-a-time.
  - e.g. A String, a Number, a Boolean, a Dictionary, an Array.
- What kind of data am I reading?
  - Same as above, retrieved file-at-time.
- Am I storing relations between things?
  - Not complicated ones.
- How persistent does my data need to be?
  - Persistent across app sessions, deleted when the app is deleted.

PROPERTY LISTS

- Good for:
  - Persisting lists
  - Persisting collections of properties
- Not good for:
  - Very data sets
  - Complex relations
  - Sensitive data

PROPERTY LISTS : STORING

Note how the array type is `NSArray`.

```
let myFavoriteNumbers: NSArray = [1.0, 1.0, 2.0, 3.0, 5.0]
// If you have a String containing a URL:
myFavoriteNumbers.writeToFile(fileUrlString, atomically: true)
// If you have an NSURL instance:
myFavoriteNumbers.writeToURL(fileUrl, atomically: true)
```

PROPERTY LISTS : RETRIEVING

```
// If you have a String containing a URL:
let myArray = NSArray(contentsOfFile: fileUrlString)

// If you have an NSURL instance:
let myArray = NSArray(contentsOfURL: fileUrl)
```

PROPERTY LISTS : DETAILS

- NSUserDefaults is a fancy wrapper around an app-specific plist file
- plists can store:
  - Strings
  - Numbers
  - Date
  - Data
  - Dictionary
  - Array
- Or any combination of the above

PLIST CODE-ALONG

GROUP ASSIGNMENT

- Extend your app to add a plist for coaches with at least 2 coaches. Coaches should have name, years of experience, coach title.
- Print the name of each coach and their title.
- Bonus: Allow user to add coaches through user interface.
- Bonus 2: Display all players and coaches in their own respective table view.

FLAT FILES AND DIRECTORY STRUCTURES

DIRECTORY

- Your app bundle contains a lot of useful information:
  - Your app binary, and all of its supporting files.
  - Your app's documents.
  - Settings.
  - Temporary files.
  - These are stored in various directories on the device's "hard drive."

DIRECTORY

- **App/Documents:** User created content. Backed up.
- **App/Documents/Inbox:** for accessing outside entities such as opening email attachments. Backed up.
- **App/YourApp.app:** The app itself and its branding supporting documents. Ie: app icons, app binary, different branding images, fonts, sounds. Not backed up.
- **App/Library:** Most other files that are not user files. This folder itself usually does not contain files, but contains sub folders where content is stored. Backed up.

DIRECTORY, CONT'D

- **App/Library/Caches:** Temporary data that our app needs to re-create later. Don't place any important application files in this folder, as system puts least priority on these files and clears out the folder if out of space. Not backed up.
- **App/Library/Preferences:** Preferences that app remembers between launches. Backed up.
- **App/tmp:** Temporary files that app creates and downloads. Used for storing downloaded files to increase app performance. The system may purge these files when app is not in use. Not backed up.

DIRECTORY, CONT'D

We can get at these directories using `NSFileManager`.

```
// Gets the documents directory (NSURL instance).
let mgr = NSFileManager.defaultManager()
let urls = mgr.URLsForDirectory(
    .DocumentDirectory,
    inDomains: .UserDomainMask
)
if let documentPath = urls.first as? NSURL {
    /* my code */
}
```

DIRECTORY, CONT'D

```
// This is for appending a file path onto the directory.
let filePath = documentPath.URLByAppendingPathComponent(
    "file.plist",
    isDirectory: false
)
```

FLAT FILES CODE ALONG

GROUP ASSIGNMENT

- Create a new file in the documents folder to store player notes.
- The signatures will be in the form of an Array of Strings.
- Append a new note to the end of the list (through user input from single text box) and save it to the signatures file.
- Notes are global, not associated with a specific player.
- Bonus: List all players, when one is clicked, give the ability to add notes to that player specifically. Those notes should be persisted and printed out when the ‘notes’ screen is navigated to.
- Bonus 2: Display and remove player notes through a table view.