GENERAL ASSEMBLY

MOBILE DEVELOPMENT FILES AND PERSISTENCE: PART 2

Tedi Konda
Executive Director, Technology, Unison

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

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, a dictionary
- 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

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, a dictionary
- 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

FLAT FILES & 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

FILES AND PERSISTENCE: PART 2

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.
- **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

- We can get at these directories with NSFileManager
- if let documentPath = NSFileManager.defaultManager().URLsForDirectory(.DocumentDirectory, inDomains: .UserDomainMask).first as? NSURL { /* my code*/ } // This gets the documents directory
- let filePath = documentPath.URLByAppendingPathComponent("file.plist", isDirectory: false) // This is for appending a file path onto the directory

FLAT FILES CODE-ALONG

FILES AND PERSISTENCE: PART 2

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.

CORE DATA

CORE DATA

- An object persistance framework
- Very powerful, very complicated
- Lots of boilerplate

CORE DATA

- Managed object model (MOM): a file that represents the data model, essentially the database schema.
- Entity: essentially a class definition in Core Data
- Attribute: a property of an entity (a member variable)
- Relationship: link between two entities. This is where entity (table) relationships are defined.
- Managed object: an entity that we want to store in Core Data. Its instances are placed in managed object context.
- Managed object context: this is the virtual representation of our data. This instance of our data can be manipulated as we like and saved when we are ready.

CORE DATA

- We always work on the managed object context
- A series of operations are performed on the MOC (insert, fetch & update, delete), then saved when we want them to persist

CORE DATA CODE-ALONG

GROUP ASSIGNMENT

- Clone your first app, use Core Data to store note information instead of files
- Bonus: Add title to notes
- Bonus: To the best of your ability, try and duplicate the UX of the iOS notes app