Basic Core Data

Just some stuff I picked up

Disclaimer

Wrote generic google photos type app using pre 2016 WWDC core data tech

“New stuff” - A lot of this is post 2016 material. Haven’t used it in complex form / solidified knowledge.

Code won’t be the cleanest / best. Aim was ease of use for t-patch prototype branch

Pros

Free faulting - uses Memento pattern to keep memory footprint down (overkill for t-patch since temp / res rates are low in memory usage)

Object graph complexity simplification - No need to load and link stuff from DB to your in memory object graph and handle high complexity faulting since Core Data does it for u (Overkill for t-patch since there are just a few simple links in our object graphs)

Free migration - no manual creating of rows / tables then writing complex unit tests for them.

Fast processing - I got my photos app to run a few percent slower than actual google photos (Overkill for t-patch, we won't be processing a ton of data)

Fast coding if you need what it provides - Got to google photo level with just two devs in about a year (We will only notice migration benefits and I doubt we'd be migrating often, it's kinda overkill)

Future proof - you can start with a simple solution and if you noticed you need more speed or complexity, it can be easy to modify / adjust your current solution.

Cons

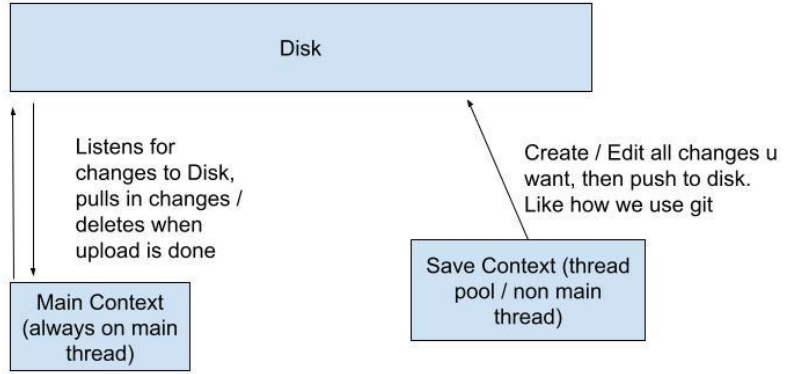
Complexity - if you use the most optimal solutions, complexity of the system can increase significantly.

Easy to crash - If your design isn’t simple and some devs don’t understand context threading, devs can easily open up crashes / bugs

High learning curve - To understand everything in core data, it takes a lot of time and effort. New core data stuff keeps coming out so your knowledge can be dated after a few years (like me, LOL)

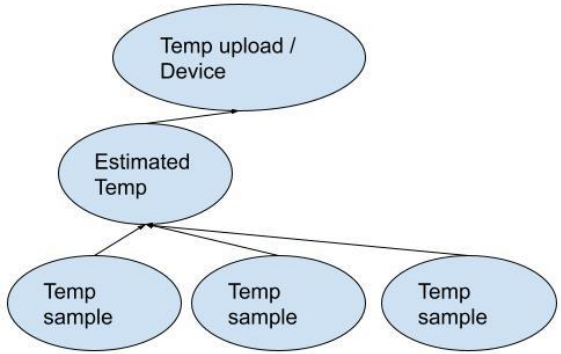
Branding - A lot of devs give up on coredata before they fully understand how to use it. This leads to high startup costs whenever someone is onboarded and possibly people not joining verily due to us using coredata.

Current R2D2 design / Seems like it's popular with auto listen now



Note: can have more than one save context, can also branch off Main context to boost efficiency but keepin it simple

Managed Objects



Example 0

Create project with core data initialized in it

Notice it creates XCData file - explain the following

- How to create entity / object
- How to add primitive data structures
- How to add one to one link - explain one to one link
- How to add many to one link - explain many to one link
- How to migrate to new version of database easily

Notice it Creates PersistenceController with container

- Get main context via “container.viewContext”
- Get save / background thread context pol via “saveContext = container.newBackgroundContext()”

Notice it makes it global for u “.environment(\.managedObjectContext, persistenceController.container.viewContext)”

Example 1

Put down Add 10 items with 1 second delay button down

Put down Remove all items button

Press add 10 items, scroll up and down. Notice UI delay

Example 2

Have main context managed objects modified on both main thread and background thread

- Notice crash (may not crash the first time, please be patient)
- Thread exec failure is usually a context thread error but other types of race conditions can occur. *run the code a few times to see different errors*

Example 3

Do creates / saves on save context and push to disk

Turn on auto listening for main context

- Notice adding to disk with no delays

How has Core Data improved “recently”?

The R2D2 model with auto merge / update is far more simple to maintain

- The other two context threads in the three context stack have been merged into one
- The save in a block has a lot of hints for devs. “Background” “saveContext”

Eliminating a lot of code

- Auto updates UI with swiftUI with little code
- Eliminating migration code
- Less unit tests / UI testing

More advanced stuff

Predicates / queries in fetch requests - Haven’t done it in the “new” stuff yet

Batch saves - This is new to me, watched some stuff in some of the more recent WWDC videos i’ll link in next slide.

How to include a core data module into a swift package -

- make sure iOS 12+,
- make sure has comment “// swift-tools-version:5.3” at top of Package.swift
- put resources:[.process(“Resources”)] inside target

Links to cool WWDC core data videos. (I don’t remember watching the last 3)

go/coredata01

go/coredata02

go/coredata03

go/coredata04

go/coredata05

Questions I still have

Does the save in save context save right to disk or does it do optimizations? For instance does it save to memory then alert main thread and only actually save to disk when it has time?

Can I squeeze more performance out of the three stack system than the R2D2 system?

How does NSPersistentContainer make the three stack system easier? (didn’t explore this because we didn’t need three stack with t-patch)

Questions from teammates

Please ask questions. I may not know the answer but I'll do my best to answer them.