Rebuilding Digg

Nate Kirby
nate@digg.com
@thenatekirby

What is this talk?

- Background.
- Why Realm?
- RealmCocoa To RealmSwift
- Lessons Learned
- Wrap Up

Wait? Digg still exists?

Digg is back.

- Purchased by News.me and Betaworks
- Rebuilt the website in 6 weeks
- Also launched an iOS app.
    - The app hasn’t aged well: missing platform features, performance issues & crashes.

Causes of Crashes

- Three main categories of crashes:
- Lack of Safety: wrong value types, unexpected nulls, bad JSON parsing
- WebKit/WebCore
- Core Data: mysterious issues.
    - [NSSQLRow newObjectIDForToOne:]

Let’s fix it.

- Type safety issues? Swift to the rescue.
- WebKit? WKWebView and a dash of hope.
- Core Data isses? New Core Data stack? SQLite? FMDB? Realm?

A Brand New Digg

Why a new app?

- Focuses change. Digg is changing.
- SDKs, platform features and devices don’t stand still for any developer.
- A lot has been learned about Mobile UI / UX.
- Crashes have piled up over time.

“Crashes Suck.”

–Someone Wise

The Great DB Debate

Requirements

- Strong performance without needing a database expert.
- Must be reliable. Reliability matters.
- Performs well with a lot of data.
- Syntax that worked in our codebase.

Testing

- We ran dozens of performance / stress tests.
- This included:
  - Many small writes & reads versus large writes and reads.
  - Large objects with many relationships.
  - Deletes & “cleanup”

Results

- Realm & SQLite eventually performed fairly evenly.
  - Initially Realm out performed SQLite by up to 4x, but SQLite query improvements brought them in line.
- FMDB performed well, but slowed under scale.
- Core Data didn’t fare as well.

Ultimately?

We chose Realm.

Why not Core Data?

- Core Data takes a lot of setup to get right.
- Core Data is opaque.
- Core Data issues don’t surface immediately.
- The cost of “comfort” isn’t free.

Why Realm?

- Realm is fast, predictable and stable.
- Realm has a lightweight syntax. It’s just like using NSObject.
  - No managed object contexts, no persistent store coordinators. Just Objects and Realms.
- Realm is open source*

Why Realm?

- The Realm Browser is fantastic.
- The Realm team responds to feedback and questions.
- Installing is as simple as pod ‘RealmSwift’...
  - ...or however you choose to bundle your libraries.

From Cocoa To Swift

Our Transition

- Took a few hours, mostly because we have a TON of Realm objects.

- The result? 45% less “Realm” code. What’s left is more readable, and far more “Swifty".

- Our setup prevented a gradual transition. It was an all or nothing affair. It might not be for you.

Pain Point

- public properties of List<Object> generate errors in the generated header file.
- @property(nonatomic) /* List<MediaImage> */ images;
    - Declare them as private, make a wrapper function to return them.

Lessons Learned

Threads

- If you’re not careful, you’ll see “Accessed on the wrong thread” errors.
    - This can be challenging to debug in multi-threaded apps.
- Solution? Don’t pass Realm Objects.
    - Make it easy & make it cheap to re-query the object.
- Better yet: copy Realm Object properties into structs.

Immutable Models

- All of Digg's UIViews are built by injecting structs containing just the information they need.
- Values are copied from Realm, dictionaries, API model responses, or mock objects.
- The view models are thread safe and static.
- If Realm hadn't worked out, the view models would abstract the UI from the data storage mechanism.

Performance

- Realm can be very fast...
- ...but if you misuse it, it won't be. It isn't immune to bad code.
- Prefer larger writes to multiple writes.
- Use instrumentation if you see slowness.
    - Our slowness issues have been tangential to Realm, not caused by it.

Querying Results<>

- No NSFetchedResultsController.
  - Results<Object> has ALL objects matching the query. You take it from there.
- Only a subset of NSPredicate (though almost everything we've needed has been there).
- You can't query into relationships.
  - "content.title = %@" = NOPE.

Other Issues

- Database migrations are manual, and the syntax isn’t the best part of Realm.
- No cascading delete. You’re going to have to do that yourself.
- Don’t forget dynamic. Forgetting dynamic on your properties leads to fun debugging.

Finally

- Realm is still pre 1.0. There’s always the chance for change.
- For anything as crucial as a database, make your decisions carefully.
- Realm works for us, but it’s not a magic bullet.

Wrap up

- Swift & Realm has forced us to write simple, logical, safe and fast code.
- If you’re using Realm in Swift, use RealmSwift.
- Don’t hesitate. Give it a go. Ask questions if you have them.

Thank You!

Questions?

Nate Kirby
nate@digg.com
@thenatekirby