Elixir Book Club, Shared Notes For:

Ash Framework:
Create Declarative Elixir Web Apps

by Rebecca Le and Zach Daniel

https://pragprog.com/titles/ldash/ash-framework/

Book description

Ash Framework is the game-changing toolkit for Elixir developers. With modular, plug-and-play building blocks, Ash slashes development time, effort, and complexity, letting you do more with less code. Design declarative, customizable domain models that are easy to maintain and optimized for performance. Shift your focus to what to build, instead of how, using Ash's intuitive design principles. Tackle bigger challenges and build scalable, future-proof web applications with confidence. Elevate your Elixir skills and revolutionize your workflow with Ash.

Discover a dynamic new way to build web applications with Elixir and Ash Framework. Ash enables you to structure and organize the code you write in a declarative programming style so you can build functionality more quickly and answer complex questions about data more easily.

Start with a Phoenix LiveView app and grow it into a fully functional system as you master Ash's core principles. You'll build the domain model for a music database throughout the book. As you create this model, you'll learn how to define resources with ease, connect them through adaptable relationships, and enrich your models with calculated attributes and aggregates to make your data work smarter. Secure your app with seamless authentication and authorization policies that integrate cleanly across APIs and user interfaces. Skip the busywork by generating REST and GraphQL APIs from your resources, freeing you to focus on building standout features. From dynamic search and nested forms to pubsub and real-time updates, you'll explore practical, real-world scenarios at every turn.

Coauthored by the creator of Ash, this book is packed with insider knowledge, best practices, and actionable guidance to get you started fast. By the end, you'll have the skills and confidence to create applications that grow with your needs.

Welcome!

- The principles, especially in deriving the functionality, are very intriguing

- This idea seems similar to deriving materialized views from an event log as put forward in designing Data Intensive Applications
- I find myself wondering how well Ash will adapt if I don't want REST and GraphQL APIs to be structured the same way

Chapter 1: Building Our First Resource

- Why does Ash add Sourceror to our dependencies instead of putting it on its own?
- I initially assumed that igniter was just an installer and was surprised to see it turned back up for generation
- I'm also a little fuzzy on the line between igniter and codegen
- I suspect a lot of this will clear up as we go, but it might have been nice to have a touch more into on something like "The Components of Ash"
- I was a little surprised when the formatter reordered my actions
- The code examples were nice and smooth, which I really appreciated
- Zorn
    - Some of the things I like:
        - Having consistency for things like authentication, authorization, pagination, filters, and more across projects with different teams seems really powerful.
    - Some of the things I worry about:
        - When I get an error, will I know where in the stack to fix it?

Chapter 2: Extending Resources with Business Logic

- The migration to add the index oddly reset the foreign key to the same value: https://github.com/JEG2/tunez/commit/e19a90596efe3e9c605bba0c6f66531138002192#diff-38665c31e93a41d7132c1bff9af04ecb4c757bfd15758d22a5839e64d6d24752
- The unique name validation seemed to only be checked on submit
    - This is true, but configurable: https://hexdocs.pm/ash/identities.html#eager-checking
- It does seem worth it to browse the lists of types, validations, and change stages (?) that the book footnotes
    - https://hexdocs.pm/ash/Ash.Type.html#module-built-in-types
    - https://hexdocs.pm/ash/Ash.Resource.Validation.Builtins.html
    - https://hexdocs.pm/ash/Ash.Resource.Change.Builtins.html
- I'm still getting my head around which parts of a relationship definition go in their own section versus which parts need to be in the Postgres section
- This chapter makes a mention of Spark, but doesn't really clear up what that is or how it plays into things: https://hexdocs.pm/spark/get-started-with-spark.html
- So far, it feels like I am learning a DSL for managing Ecto and forms, which is fine, but I'm not yet clear on how much this is saving me (generated migrations, interface functions, and more declarative validations maybe?)
    - Zorn: I think much of the benefits are in the second half of the book.

Chapter 3: Creating a Better Search UI

- JEG2's random thoughts so far
    - Ash can help me do a lot of things I am forced to do in every project
        - Build migrations, schemas, and changesets
        - Wrap derived queries and calculations in API's
        - Define contexts and enforce boundaries
        - Filter, sort, and paginate UI's based on parsed query params
        - Looking ahead: offer external API's, support authentication, validate authorization, and more
    - The advantages for letting Ash help
        - Many common needs are pre-solved (example: aggregating related fields while ignoring nils)
        - It still helps with some other needs that I must manage (example: choosing what relationships and calculations to load at the UI, context, and CRUD layers)
        - I benefit from their collective optimization and bug bashing efforts
    - The downsides of leaning on Ash
        - You must learn a significantly large whole other system
            - Sad insight: the book doesn't really solve this problem
                - It succeeds more as an overview and inspiration
                - It is not a reference
                - It's not the best at even clarifying which components are involved (as Mike would say: "I would have appreciated more diagrams.")
                - The Ash docs do seem pretty thorough though
        - Debugging will probably be a little tougher with the added layer of unknowns
        - You will inevitably need to work around some of their bugs and limitations
- A good discussion of the tradeoffs: https://elixirforum.com/t/marketing-ash-why-you-should-use-ash/71487/10

Chapter 4: Generating APIs Without Writing Code

- I didn't see where the book remind us to "extend" albums with the JSON API
    - On p. 92, in the first sentence under "Adding Albums to the API" (wennefer)
        - You are totally correct!
- This is a pretty fantastic way to wire up API's, as far as I am concerned
- These tools seem to significantly lower the costs of switching API's or supporting multiple API's

Chapter 5: Authentication: Who Are You?

- Notes TBD

Chapter 6: Authorization: What Can You Do?

- Notes TBD

Chapter 7: Testing Your Application

- Notes TBD

Chapter 8: Having Fun With Nested Forms

- Notes TBD

Chapter 9: Following Your Favorite Artist

- Notes TBD

Chapter 10: PubSub and Real-Time Notifications

- Notes TBD