Corralling Microservices with Functional Programming

@benkyrlach

Coordinating Microservices

- Architecture consists of many “crud” microservices
- Some of which write to disparate databases
- Needed a way to coordinate a few complex operations

About The M-word

- Monads get a bad rap
- Barrier to understanding due to extreme abstract nature
- Don’t need to understand Category Theory to use!!!

Monads

- Abstract away software concerns from business concerns (handling errors vs. computing an interest rate)
- Allow the application of mathematics to real world problems

Monad Operations

- Pure -> Lift a value into a specific problem domain
- Map -> Lift a function into a specific problem domain
- FlatMap -> Compose operations in a specific problem domain

Problem Statement

- Create service coordinator for complex operations
- Easy to compose crud operations into more complex ones
- Each action associated with an “undo” in case of a failure
- Undo actions performed in reverse order

IO

- Domain of side-effects
- Abstracts away “what” is being done from “when” it is done
- Allows for equational reasoning about impure code
- Side effects can now be manipulated

Either

- Domain of errors
- Abstracts away error handling from business logic
- Allows for equational reasoning about computations that might error
- Puts you back in control of when/how errors are resolved

State

- Domain of stateful computations
- Abstracts away global state
- Allows for equational reasoning about functions that share state
- Puts you back in control of what state is available

`

- Combines ideas from IO, Either, and State
- Abstracts away a side effect combined with a list of instructions to undo it
- Composing undoable combines undo trails and side effects
- Run or undo generates a program to execute the side effect, and run the undo script if it fails

?