Intro to Hyperloop / HyperStack

by @jimmynguyc

What is Hyperloop?

- Previously React.rb
- Full-stack modern web tooling for SPA
- “The Complete Isomorphic Ruby Framework”
- Renaming to Hyperstack

Not to be confused with …

Virgin hyperloop one

COMPS

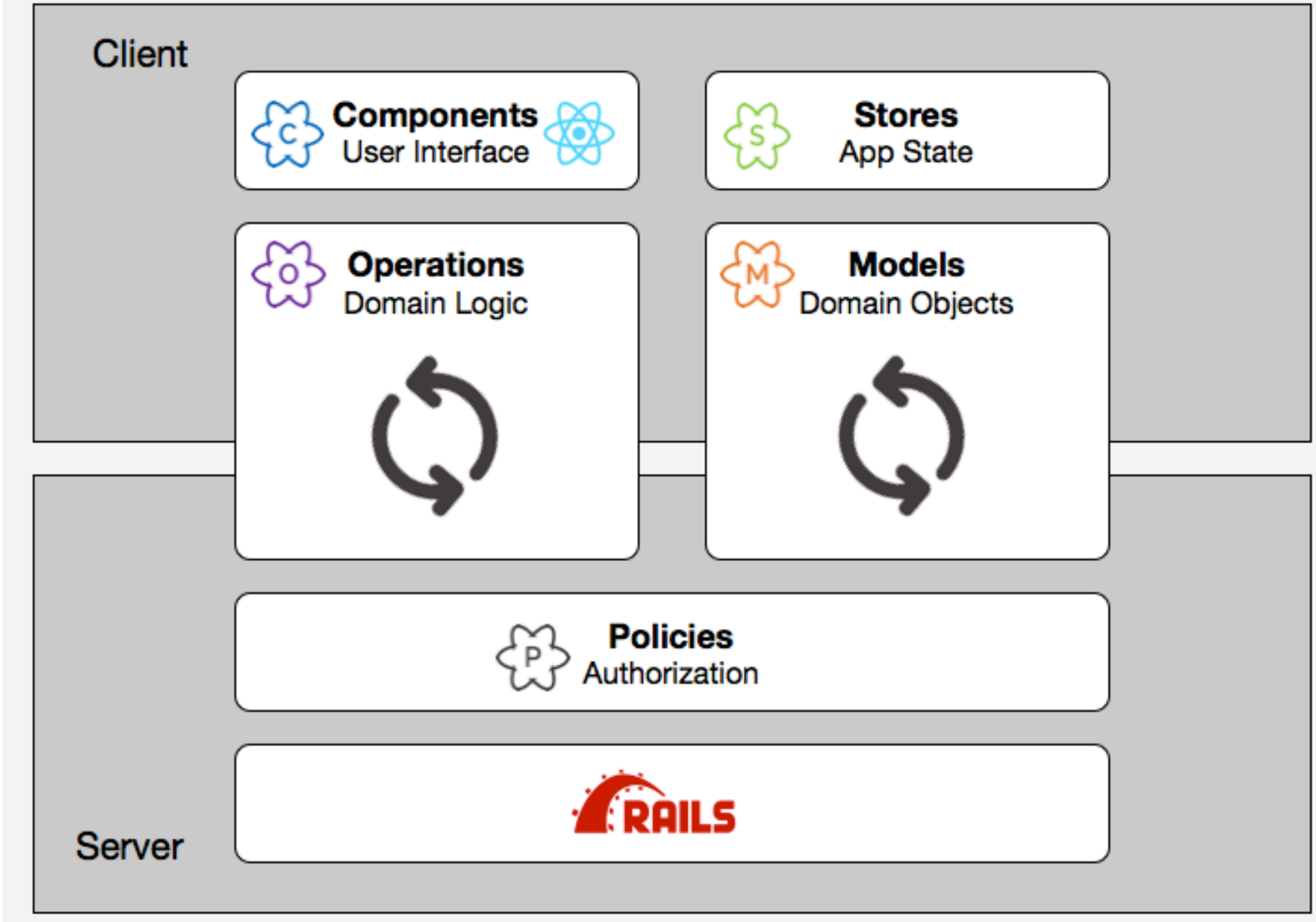
Components describe how the UI will display the current application state and how it will handle user actions. Using React, Components automatically rerender parts of the display as state changes due to local or remote activities.

Operations encapsulate business logic. In a traditional MVC architecture, Operations end up either in Controllers, Models or some other secondary construct such as service objects, helpers, or concerns. Here they are first class objects. Operations orchestrate the interactions between Components, external services and Stores.

Your ActiveRecord **Models** are available in your Isomorphic code. Components, Operations, and Stores have CRUD access to your server side Models, using the standard ActiveRecord API. Amazingly, we automatically synchronize data between connected clients.

Policies keep authorization logic out of Models, and Operations, and also allows the isomorphic transport mechanism to know what and when to communicate between client and server.

Stores hold the local application state. Stores are Ruby classes that keep the dynamic parts of the state in special state variables. We use Stores to share state between Components.


Client
Components
User Interface
Stores
App State
Operations
Domain Logic
Models
Domain Objects
Policies
Authorization
RAILS
Server

Hyperloop::Component

```
class BookList < Hyperloop::Component
  # Display each book in our catalog unless it's already in the cart basket.
  # When the user clicks on a book, add it to the Basket.
  render(UL) do
    Book.all.each do |book|
      LI { "Add #{book.name}" }.on(:click) do
        AddBookToBasket(book: book)
      end unless acting_user.basket.include? book
    end
  end
end
```

Hyperloop::Operation

```ruby
class AddToActingUsersWatchList < Hyperloop::Operation
  # Add a book to the current acting_user's watch list, and
  # send an initial email about the book.

  param :book, type: Book
  # Operations have access to the current 'acting_user'
  # so we do not need to pass it as a parameter.

  step do
    return if acting_user.watch_list.include? params.book
    WatchListMailer.new_book_email\
      WatchList.create(user: acting_user,  book: params.book)
  end
end
```

Hyperloop::Model

```
class Book < ApplicationRecord
  scope :in_catalog, -> { where(catalog: true) }

end
```

```
class BookList < Hyperloop::Component
  # Display each book in the catalog
  render(UL) do
    Book.in_catalog.each do |book|
      LI { book.name }
    end
  end
end
```

Hyperloop::Policy

```ruby
class BookPolicy
  regulate_broadcast do |policy|
    # allow the entire application to see all book attributes
    # except the 'unit_cost'.
    policy.send_all_but(:unit_cost).to(Application)
  end
  # but only acting_user's who are admins can make changes to Books
  allow_change(on: [:create, :update]) { acting_user.admin? }
end

class OperationPolicy
  # We need AddToActingUsersWatchList to execute on the server but
  # it can be invoked from the client if there is a logged in user.
  AddToActingUsersWatchList.execute_on_server { acting_user }
end
```

Hyperloop::Store

```
class Discounter < Hyperloop::Store

  state discount: 30, scope: :class, reader: true
    def self.lucky_dip!
      mutate.discount( state.discount + rand(-5..5) )
    end
  end
end
```

Hyperloop + Rails

Demo

Why Hyperloop

- Super easy to pick up (tutorials, docs, sample apps, etc)
- One language - front-end & back-end using Ruby
- Operations & models are isomorphic
- OOTB pundit style authorization & Server side rendering
- Active contributors

Why Not Hyperloop

- Fairly new + low adoption = almost non existent StackOverflow issues

- Framework lock in

- You'll need to learn React & all the accompanying tooling anyways

- Extra layer of trans-piling (Ruby -> JS) increases project built time

- Bugs?

Resources

- http://ruby-hyperloop.org/

- https://hyperstack.org/

Thanks