Rails and SAAS

Aspect-Oriented Programming

- Cross-cutting concern ≜ logically centralized, but may appear multiple places in implemented
- Advice ≜ a specific piece of code that implements a cross-cutting concern
- Pointcut ≜ places you want to 'inject' advice at runtime
- Aspect ≜ advice + pointcut

Validations

- Validation ≜ advice in AOP sense
  - Many places in app where a model could be modified/updated
- Can be done with length, presence, regex, within range or list of values
  - Can conditionally apply a validation
- Custom validations use `validate :<function name>`
  - Functions add to `errors` to signal invalid data
- Can use `before_validation`, `before_validation_on_create`, `before_save`, `after_save` etc. to run pre and post operations
- Can be defined in an `ApplicationRecord` with `validates :<field>, <condition>` where `<condition>` can be something like `length: { in: 1..40 }`
  - Will then respond to `valid?` function and have `errors` getter
- `errors` responds to `full_messages` which converts errors into array of message strings
  - Can then call `to_sentence` to convert them all into a sentence

Controller Filters

- eg: how to ensure a user is logged in
  - Redirect them to the right spot
  - Ensure only admins can access internal pages
- Use `before_action :<function name>` to call the action before each controller action
  - Filters inherit; a filter in `ApplicationController` will apply to all controllers
  - Rails v < 5 declared via `before_filter`
- Operate like controller methods; have access to `flash` and `session`
- eg:

```ruby
before_action :set_movie, only: %i[ show edit update destory]
def set_movie
    @movie = Movie.find(params[:id])
end
```

- Often implement patterns like `set_current_user` and `require_user`
- Can skip a `before_action` with `skip_before_action :<action name>`

Validations VS Filters

	Validation	Filter
Advice (DRYness)	Check invariants on model	Check conditions for allowing controller action to run
Pointcut	AR model lifecycle hooks	before and/or after any public controller method
Can change execution flow	No	Yes
Can define advice in arbitrary function	Yes; shortcuts provided for common cases	Yes, must provide function

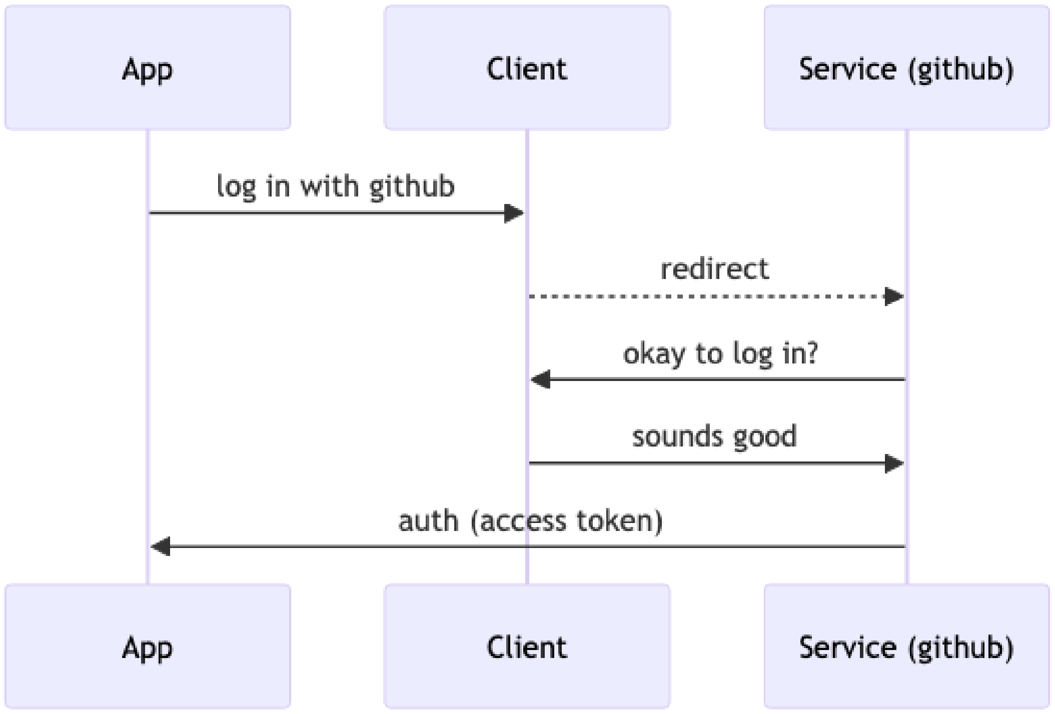
	Validation	Filter
Info about errors	Each model object has associated `errors` object	Capture in `flash`, `session`, or instance variable

Partials

- `render` inside a view allows sharing code between views
- Common examples:
  - `movies/_form`
    - `< %= render "form", moviegoer: @moviegoer %>`
  - `shared/_flash`
  - `users/_user_profile`
- Convention: names start with _
- Files live inside `app/views/` but we don't need to specify that
- Pass data to partials as a hash
- Tip: avoid using instance variables in partials b/c you don't know which instance variables will be available

SSO and Third-Party Auth

- Can share info between applications without having to reveal password to an app
- Auth: prove who you say you are
- Authorization: prove you are allowed to do what you're asking
- Most sites don't have a RESTful API, so you need to 'log in' (or simulate it)
- Doesn't work for SOA



- Usually use a `SessionsController`
- Benefit: you store only minimal data