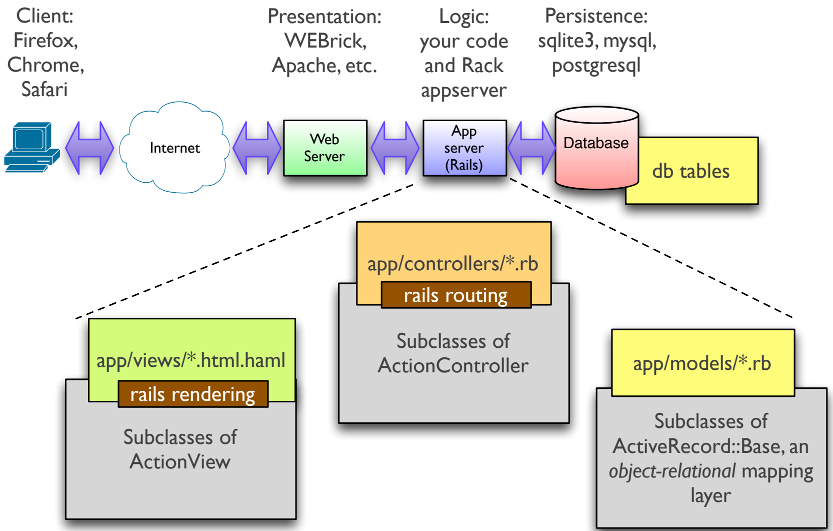
Rails overview and walkthrough

COSC 480
19 February 2018

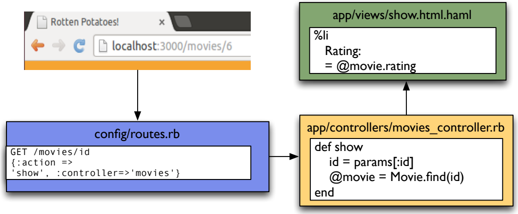
Joel Sommers
jsommers@colgate.edu
Colgate University

Rails as an MVC Framework


Client: Firefox, Chrome, Safari
Presentation: WEBrick, Apache, etc.
Logic: your code and Rack appserver
Persistence: sqlite3, mysql, postgresql
Internet
Web Server
App server (Rails)
Database
db tables
app/controllers/*.rb
rails routing
Subclasses of ActionController
app/views/*.html.haml
rails rendering
Subclasses of ActionView
app/models/*.rb
Subclasses of ActiveRecord::Base, an *object-relational* mapping layer

A trip through a Rails app

1. Routes (in `routes.rb` map incoming URLs to controller actions and extract optional parameters
    - A route's wildcard parameters (e.g., `:id`), plus any stuff after `?` in URL are put into `params[]` hash, accessible in controller actions
2. Controller actions set instance variables, which are visible to views
    - Subdirectories and filenames of `views/` match controller and action names
3. Controller eventually renders a view



```
GET /movies/id
{:action =>
'show', :controller=>'movies'}
```

```
def show
  id = params[:id]
  @movie = Movie.find(id)
end
```

```
%li
  Rating:
  = @movie.rating
```

Rails Philosophy

- Convention over configuration
  - If naming follows certain conventions, no need for config files
    - Very much unlike other frameworks, e.g., Django
- Don't Repeat Yourself (DRY)
  - Mechanisms to extract common functionality
- Both rely heavily on Ruby features
  - Introspection and metaprogramming
  - Blocks (closures)
  - Modules (mix-ins)

Rails app walkthrough

- Open a terminal, cd to some location in which you want to make a new rails app

```
# create a new app container; don't install "old" unittest stuff,
# don't automatically install all the gem dependencies, don't use
# turbolinks
$ rails new testapp --skip-turbolinks -T -B
```

- Change directories into the new app folder (testapp)
- In `Gemfile` add the following:

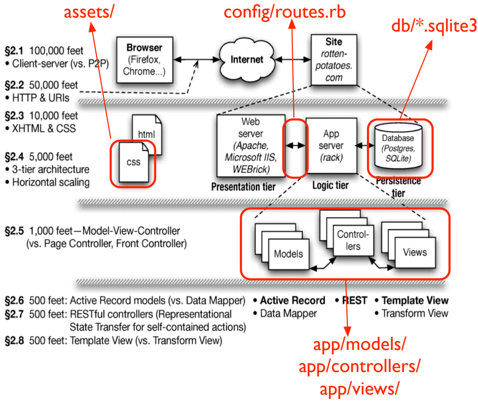
```
gem 'haml', '~> 5.0.4'
gem 'haml-rails'
```

- Note: for heroku, you need `pg` and `rails_12factor` in a `:production` group
- Now install all gems (reads your Gemfile and installs all the gems listed):

```
$ bundle install --without production
```

Connecting Architectural Concepts to Rails Apps

```
# top-level of rails app
Gemfile
Rakefile
app/
  models/, views/, controllers/
  helpers/
  assets/
config/routes.rb
db/
  development.sqlite3, test.sqlite3
  migrate/
log/
  development.log, test.log
```



assets/
config/routes.rb
db/*.sqlite3
§2.1 100,000 feet
• Client-server (vs. P2P)
Browser (Firefox, Chrome...)
Internet
Site rotten-potatoes.com
§2.2 50,000 feet
• HTTP & URIs
§2.3 10,000 feet
• XHTML & CSS
html
css
Web server (Apache, Microsoft IIS, WEBrick)
App server (rack)
Database (Postgres, SQLite)
§2.4 5,000 feet
• 3-tier architecture
• Horizontal scaling
Presentation tier
Logic tier
Persistence tier
§2.5 1,000 feet—Model-View-Controller (vs. Page Controller, Front Controller)
Models
Controllers
Views
§2.6 500 feet: Active Record models (vs. Data Mapper)
§2.7 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
§2.8 500 feet: Template View (vs. Transform View)
• Active Record
• Data Mapper
• REST
• Template View
• Transform View
app/models/
app/controllers/
app/views/

Start me up

- Start up your new rails app

  ```
  $ rails server
  ```

- Open a browser, go to `http://localhost:3000`
- Keep the server running, open a new terminal in the top-level folder of the app

Adding routes

- Type `rails routes` to see what "routes" your app knows about
  - Routes map an incoming URI to a controller and method
  - We don't have any controllers yet (or models, or views, for that matter)
  - (We're going to ignore the model for now)
- Create a new controller:

```
$ rails generate controller  # will dump out some help
$ rails generate controller RentalProperties
```

- Add the following lines to `config/routes.rb` (inside the do..end block)

```
  resources :rental_properties
  root 'rental_properties#index'
```

- Now run `rails routes` again
- Reload your browser
- Also type the following URL in your browser:
  http://localhost:3000/rental_properties/3
  - Look at the output of `rails routes` to see how this works...

Adding to the controller

- Add the following method to `RentalPropertiesController` class in `app/controllers/rental_properties_controller.rb`:

```
def index
end
```

Adding a view template

- Add the following to the file `app/views/rental_properties/index.html.haml`:

```
%h1 It works!
```

Passing information from controller to view

- Add the following line to `RentalPropertiesController#index`
- Data are passed to the view by assigning to instance variables in the controller
- By default, a view template corresponding to the controller method is rendered, but you can call the `render` method to explicitly render any template

```
@message = "Hello, rails!"
render 'index'
```

- Add the following line to the rental properties index view:

```
%p= "Here's the message: #{@message}"
```

Debugging rails apps

- Debugging SaaS applications can be very hard
  - Quite a bit of complexity
  - Many various components that are involved in handling a single request
  - Distributed systems are hard to debug, period
- Three key debugging techniques with rails:
  - logging
    - There's no printf-style debugging in Rails: use the logger
    - `logger.debug(string)`, or `logger.info(string)`, or `logger.fatal(string)`
    - Messages go to `log` directory
  - interactive debugger
    - Add `byebug` anywhere in your app to immediately go into a debugger console
    - Do this in `RentalPropertiesController#index`, print out `params`, `request.method`, `request.[port,host,url,...]`
    - Type `cont` to let the request continue
- http://guides.rubyonrails.org/debugging_rails_applications.html