Ruby on Rails Hotwire Project Structure

Full Hotwire stack with Turbo and Stimulus. SPA-like experience with server-rendered HTML.

#rails #ruby #hotwire #turbo #stimulus #full-stack

PNG PDF Copy Prompt

Project Directory

```
myapp/
    Gemfile
    Gemfile.lock
    package.json
    Rakefile
    config.ru
  app/
    controllers/
      application_controller.rb
      posts_controller.rb
      comments_controller.rb  Turbo stream re…
    models/
      application_record.rb
      post.rb  broadcasts_to f…
      comment.rb
    views/
      layouts/
        application.html.erb
      posts/
        index.html.erb
        show.html.erb
        _post.html.erb  Turbo Frame par…
        _form.html.erb
      comments/
        _comment.html.erb
        create.turbo_stream.erb  Stream response
    javascript/  Stimulus contro…
      application.js
      controllers/
        application.js
        index.js  Auto-load contr…
        dropdown_controller.js
        modal_controller.js
        form_controller.js
    channels/  Action Cable fo…
      application_cable/
    helpers/
    assets/
  config/
    application.rb
    routes.rb
    database.yml
    cable.yml  WebSocket config
    importmap.rb  JavaScript impo…
    environments/
    initializers/
  db/
  test/
  lib/
  public/
  bin/
```

Why This Structure?

Hotwire (HTML Over The Wire) delivers SPA-like experiences without JavaScript frameworks. Turbo handles navigation and form submissions, Turbo Streams update page fragments, and Stimulus adds behavior. Server renders HTML, client stays light.

Key Directories

app/javascript/controllers/ - Stimulus controllers for JavaScript behavior

app/views/*/*.turbo_stream.erb - Turbo Stream templates for live updates

app/channels/ - Action Cable for WebSocket broadcasts

config/importmap.rb - JavaScript dependencies without bundler

Turbo Streams Broadcast

```
# app/models/comment.rb
class Comment < ApplicationRecord
  belongs_to :post
  broadcasts_to :post
end


<%= turbo_stream_from @post %>

  <%= render @post.comments %>
```

Getting Started

1. `rails new myapp` (Hotwire included by default)
2. `rails generate scaffold Post title:string`
3. `rails db:migrate`
4. Add `broadcasts_to` to models for live updates
5. `bin/dev` (starts Rails + CSS watcher)

When To Use This

- Traditional apps wanting SPA-like feel
- Real-time features (chat, notifications, dashboards)
- Teams without JavaScript framework expertise
- When you want to keep stack simple
- Progressive enhancement of existing Rails apps

Trade-offs

Learning curve - Turbo Frames/Streams concepts take time

Complex interactions - Very dynamic UIs may need more Stimulus

WebSocket requirement - Turbo Streams broadcasting needs Action Cable

Naming Conventions

Stimulus controllers - snake_case: dropdown_controller.js

Turbo Stream templates - action.turbo_stream.erb

Turbo Frame IDs - dom_id helper: post_1, comment_5