My Experience Using Turbo in Production

Florin Lipan, 2021

About

- Florin Lipan
- Staff Engineer @Chatterbug (We're hiring!)
- Turbo
    - Turbo Drive
    - Turbo Frames & Turbo Streams
    - Turbo Streams over Websockets
- Some things to keep in mind
- Writing tests
- Scaling

Turbo

- A collection of tools for building interactive websites
- Mostly based on server-side rendered HTML
- Getting started:

```
gem install turbo-rails
yarn add @hotwired/turbo-rails
# NOTE: The package called @hotwired/turbo doesn't support WebSockets!
```

- *The only Javascript you'll ever need* (quotation needed):

```
import { Turbo, cable } from '@hotwired/turbo-rails'
```

Turbo Drive

- Turbolinks 2.0
- Turns every link click or form submission into an XHR call
- Replaces the `<body>` tag, preserves the `<head>`, adds a loader
- Can also extract Turbo frames (if any) and match them correctly
- Requires special attention when integrated with other tools (React, jQuery)

Demo

Turbo Frames

- `<turbo-frame id="my-frame">...</turbo-frame>`
- `<%= turbo_frame_tag("my-frame") do %>...<% end %>`
- Defines the boundaries of **a component to be updated on request**
- Should be **unique -** `dom_id(...)` is your friend
- Can be loaded **lazily** (`loading="lazy"`)

Turbo Streams (1)

- A **simple HTTP-based protocol** to provide targeted updates for Turbo frames
- Supports **append/prepend/update/remove/before/after**

```
<turbo-stream action="replace" target="my-frame">
  <template>
    <turbo-frame id="my-frame">
      <div>hello world</div>
    </turbo-frame>
  </template>
</turbo-stream>
```

Turbo Streams (2)

The protocol is completely abstracted away by Rails

```
<!-- app/views/posts/_post.html.erb -->
<turbo-frame id="my-frame">
  <div>The time is: <%= Time.now %></div>
</turbo-frame>
```

```
# app/controllers/posts_controller.rb
class PostsController < ApplicationController
  def update
    render turbo_stream: turbo_stream.replace("my-frame", partial: "posts/post")
  end
end
```

Demo

Turbo Streams over Websockets

- Rails can provide **updates to Turbo frames via Websockets** (ActionCable)
- **Subscribe** a client to events and define the Turbo frame:

```
<%= turbo_stream_from :my_events %>
<turbo-frame id="my-frame"></turbo-frame>
```

- **Trigger** an event:

```
Turbo::StreamsChannel.broadcast_replace_to(
  :my_events,
  target: "my-frame",
  partial: "posts/post"
)
```

- ...or **trigger** an event **in the background**: `#broadcast_replace_later_to`

Demo

Rethink your frontend with Rails

- **Components**: partials or github/view_component
- Solve problems **over the wire** rather than via Javascript
- Example: changing a button's state after submitting a form (include the change in the response)
- For everything else try **Stimulus**

Things to keep in mind (1)

- Replace `link_to` with `button_to` - Turbo Streams **works only on forms**
- Turbo Streams **doesn't work on forms submitted via GET**
- Try and forget about Rails UJS (`data-remote`)
- When **rendering with errors** from a controller, set a status that's **not 2xx**: `render status: :unprocessable_entity`
- **Nesting a** `<turbo-frame>` element **inside a** `<table>` will break the layout: instead Turbo actually allows referencing any element's ID
- **Devise** needs a few configuration tweaks (or errors are not reported properly)

Things to keep in mind (2)

- **Updating multiple frames** at once:

```
render turbo_stream: turbo_stream.replace(:one) + turbo_stream.append(:two) + ...
```

- **Disabling Turbo** for particular pages or elements:
  - `<div data-turbo="false">...</div>`
  - Default to opt-in rather than opt-out: `Turbo.session.drive = false`
  - `<meta name="turbo-root" content="/my-subdomain">` (doesn't seem to cover forms though?!)

Things to keep in mind (3)

- Keep your components small, **never reload more than you need**
- Avoid using **instance variables inside your partials**, prefer *locals*
- Avoid using **controller-specific variables in your broadcasted partials** - e.g. `current_user`
- You can **namespace** broadcasted messages:

```
<!-- Customer UI -->
<%= turbo_stream_from post, :customer %>

<!-- Admin UI -->
<%= turbo_stream_from post, :admin %>
```

```
Turbo::StreamsChannel.broadcast_replace_to(post, :customer, target: "...", partial: "...")
Turbo::StreamsChannel.broadcast_replace_to(post, :admin, target: "...", partial: "...")
```

How do I test this thing? (1)

- System tests (slow) vs. testing in isolation / by contract
- View tests

```
assert_select "turbo-frame#my-frame", { text: "..." }
```

- Controller tests

```
post(users_path, as: :turbo_stream)
assert_turbo_stream(action: :replace, target: "my-frame")
```

- You can also access the frame contents

```
assert_turbo_stream(action: :replace, target: "my-frame") do |selected|
  assert_match /something/, selected.to_html
end
```

How do I test this thing? (2)

- For Websockets, use `ActionCable::Channel::TestCase`

```
assert_broadcasts :my_events, 5
assert_no_broadcasts :my_events
```

- When the stream key is an object

```
# Turbo::StreamsChannel.broadcast_replace_to(some_object, :namespace)

stream_name = ["namespace", some_object.to_gid_param].join(":")
assert_broadcasts stream_name, 1
```

- For events triggered in the background, you'll need `ActiveJob::TestHelper`:

```
perform_enqueued_jobs(only: Turbo::Streams::ActionBroadcastJob) { ... }
```

Scaling

- Pretty old-school
- Scaling your servers (backend, database)
- Turbo Streams over Websockets:
  - `config.action_cable.worker_pool_size`
  - `broadcast_replace_to` vs. `broadcast_replace_later_to` - do you need it in real time?
  - Scaling Sidekiq
  - AnyCable

Further reading

- https://turbo.hotwire.dev/handbook/introduction
- https://github.com/hotwired/hotwire-rails-demo-chat
- https://github.com/hotwired/turbo-rails/blob/main/app/models/concerns/turbo/broadcastable.rb
- https://dev.to/fadrien/rails-devise-and-recaptcha-with-hotwire-turbo-and-stimulus-2hoh
- https://github.com/github/view_component
- https://github.com/anycable/anycable
- https://stimulus.hotwire.dev/handbook/introduction

Questions?

https://lipanski.com/slides/turbo