Make a nicer looking blog with *Action Text* and *embeds*

(or, how I'm vaguely retelling how I made a blogging feature for a side project I did)

Who am I?

- I'm from Penang
- First and current place of work is Fave, where I started as a software engineer and now work as an associate manager
- Wouldn't call myself a Ruby-ist, but definitely love using it in general

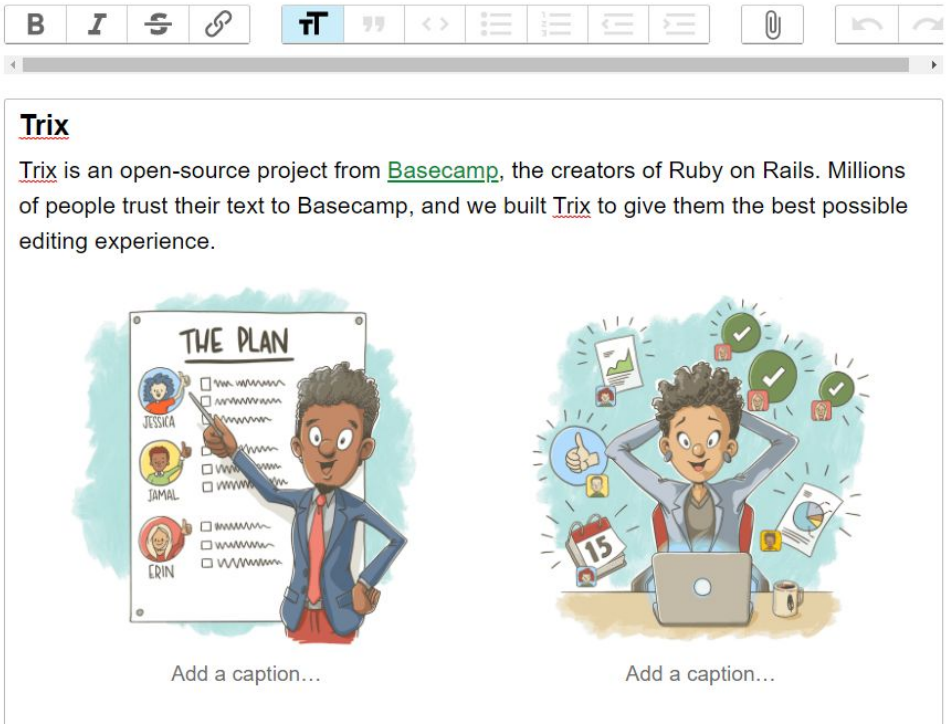
Why do I want to make a nicer looking blog post?

1. Maybe you want to relive the days of having a blogspot (why?)
2. Maybe whoever is paying you wants to have their blog look nicer and have more interesting content than just text
3. Or, maybe you don't and you'll post all about your exploits with just plain text (ala Barney Stinson) but no one reads it in the end

Whatever it is, it's 2022. Just sign up for Medium.

What is *ActionText*?

- Brings rich text content and editing using Trix to Rails
- Why is this great?
  - Built-in: no need to worry about it being maintained
  - Customisable: you can build your own functions



How does it work?

```ruby
class Post < ApplicationRecord
  has_rich_text :body
end
```

```erb
<div class="col-8 mb-2">
  <div class="form-group my-1">
    <%= form.label :body %>
    <%= form.rich_text_area :body, class: "form-control" %>
  </div>
</div>
```

So what about embeds?

- What kind of embeds can I use with Action Text?
  - Action Text already supports drop-in for images, video, pdf etc.
  - How about YouTube, Spotify, Soundcloud? So I can feel like I'm using Friendster / Yaooagain
- This is where the customisation of the Trix editor comes in

Customising your Trix editor to support embeds

Disclaimers

1. I’m assuming you’ve already installed Action Text
2. Most of this I implemented 6 - 8 months ago so there *will* definitely be some gaps in my memory
3. Most obviously, I’m assuming you’re using Rails 6 and above

Step 1
Install Stimulus js

Refer here for more information

1. Run *yarn install stimulus*
2. Under *app/javascript/controllers* create the *index.js* file which will hold the below starter code
3. Lastly, import the controllers in your *application.js* file

```
// Load all the controllers within this directory and all subdirectories.
// Controller files must be named *_controller.js.

import { Application } from "stimulus"
import { definitionsFromContext } from "stimulus/webpack-helpers"

const application = Application.start()
const context = require.context("controllers", true, /_controller\.js$/)
application.load(definitionsFromContext(context))
```

```
require("@rails/actiontext");
import "controllers";
```

Step 2
Create the *Embed* class

```ruby
class Embed
  include ActiveModel::Model
  include ActiveModel::Attributes
  include GlobalID::Identification
  include ActionText::Attachable
  YT_EMBED = "youtube.com/embed".freeze
  YT_WATCH = "youtube.com/watch".freeze
  SPOTIFY_TRACK = "spotify.com/track".freeze
  SPOTIFY_ALBUM = "spotify.com/album".freeze
  SPOTIFY_PLAYLIST = "spotify.com/playlist".freeze
  SOUNDCLOUD_TRACK = "soundcloud.com/tracks".freeze
  SOUNDCLOUD_PLAYLIST = "soundcloud.com/playlists".freeze
  VIMEO = "vimeo.com".freeze

  attribute :id

  def self.find(id)
    new(id: id)
  end

  def source
    case
    when id.include?(YT_EMBED)
      id
    when id.include?(YT_WATCH)
      params = Rack::Utils.parse_query(URI(id).query)
      "https://www.youtube.com/embed/#{params["v"]}"
    when id.include?(VIMEO)
      param = id.split('/').last
      "https://player.vimeo.com/video/#{param}"
    when id.include?(SPOTIFY_TRACK)
      param = URI(id).path.split('/').last
      "https://open.spotify.com/embed/track/#{param}"
    when id.include?(SPOTIFY_ALBUM)
      param = URI(id).path.split('/').last
      "https://open.spotify.com/embed/album/#{param}"
    when id.include?(SPOTIFY_PLAYLIST)
      param = URI(id).path.split('/').last
      "https://open.spotify.com/embed/playlist/#{param}"
    else
      id
    end
  end
```

Step 3
Create a Rails controller and view for the embeds

This will come into use later

1. Create a PATCH method which returns a JSON with the content of the embed, as well as the sgid of the embed
2. Make sure to add html sanitization in your initializers as well

```
class EmbedsController < ApplicationController
  def update
    embed = Embed.new(id: params[:content])
    content = ApplicationController.render(partial: 'embeds/embed',
                                           locals: { embed: embed },
                                           formats: :html)
    render json: { content: content, sgid: embed.attachable_sgid }
  end
end
```

```
<% if embed.video? || embed.spotify_album_or_playlist? || embed.soundcloud_playlist? %>
  <%= content_tag :iframe, nil, width: "100%", height: 420, src: embed.source %>
<% elsif embed.spotify_track? || embed.soundcloud_track? %>
  <%= content_tag :iframe, nil, width: "100%", height: 200, src: embed.source %>
<% else %>
  <%= content_tag :iframe, nil, width: "100%", height: 200, src: embed.source %>
<% end %>
```

```
Rails::Html::WhiteListSanitizer.allowed_tags << "iframe"
Rails::Html::WhiteListSanitizer.allowed_tags << "video"
Rails::Html::WhiteListSanitizer.allowed_tags << "source"
Rails::Html::WhiteListSanitizer.allowed_tags << "audio"
```

Step 4
Create the Stimulus controller for adding embeds in Trix

This will take a few slides

Under the *app/javascript/controllers* create your stimulus controller. Name it something obvious, because you will be adding it to your view later on, e.g. *trix_embed_controller.js*

```
import { Controller } from "stimulus"
import Trix from "trix"
import Rails from "@rails/ujs"

You, 7 months ago | 1 author (You)
export default class extends Controller {…
}
```

```
connect() {
  this.addTrixButton()
  this.addTrixDialog()
  this.eventListenerForMediaButton()
  this.eventListenerForAddButton()
}
```

```
addTrixButton() {
  const buttonHTML = `<button
                        type="button"
                        class="trix-button"
                        data-trix-attribute="embed"
                        data-trix-action="embed"
                        title="Embed"
                        tab-index="-1">
                        Media
                      </button>`
  const buttonGroup = this.element.toolbarElement.querySelector(".trix-button-group--block-tools")
  buttonGroup.insertAdjacentHTML("beforeend", buttonHTML)
}
```

```
addTrixDialog() {
  const dialogHTML = `<div
                        class="trix-dialog trix-dialog--link"
                        data-trix-dialog="embed"
                        data-trix-dialog-attribute="embed">
                        <div class="trix-dialog__link-fields">
                          <input
                            type="text"
                            name="embed"
                            class="trix-input trix-input--dialog"
                            placeholder="Paste your URL"
                            aria-label="embed code"
                            required=""
                            data-trix-input=""
                            disabled="disabled">
                          <div class="trix-button-group">
                            <input
                              type="button"
                              class="trix-button trix-button--dialog"
                              data-trix-custom="add-embed"
                              value="Add">
                          </div>
                        </div>
                      </div>`
  const dialogGroup = this.element.toolbarElement.querySelector(".trix-dialogs")
  dialogGroup.insertAdjacentHTML("beforeend", dialogHTML)
}
```

```
eventListenerForMediaButton() {
  document.querySelector('[data-trix-action="embed"]').addEventListener("click", event => {
    const dialog = document.querySelector('[data-trix-dialog="embed"]')
    const embedInput = document.querySelector('[name="embed"]')
    if (event.target.classList.contains("trix-active")) {
      event.target.classList.remove("trix-active");
      dialog.classList.remove("trix-active");
      delete dialog.dataset.trixActive;
      embedInput.setAttribute("disabled", "disabled");
    } else {
      event.target.classList.add("trix-active");
      dialog.classList.add("trix-active");
      dialog.dataset.trixActive = "";
      embedInput.removeAttribute("disabled");
      embedInput.focus();
    }
  })
}
```

```
eventListenerForAddButton() {
  document.querySelector('[data-trix-custom="add-embed"]').addEventListener("click", event => {
    const content = document.querySelector("[name='embed']").value
    if (content) {
      let _this = this
      let formData = new FormData()
      formData.append("content", content)
      Rails.ajax({
        type: 'PATCH',
        url: '/embed.json',
        data: formData,
        success: ({content, sgid}) => {
          const attachment = new Trix.Attachment({content, sgid})
          _this.element.editor.insertAttachment(attachment)
          _this.element.editor.insertLineBreak()
        }
      })
    }
  })
}
```

Step 5
Add the embed to your existing Trix editor

```
<div class="col-8 mb-2">
  <div class="form-group my-1">
    <%= form.label :body %>
    <%= form.rich_text_area :body, class: "form-control", data: { controller: "trix-embed" } %>
  </div>
</div>
```

Thank you for listening!

How to get in touch with me

1. LinkedIn: Pengiran Nazrin
2. Github: Nazisagit