Object Oriented WEB Programming Using Ruby

Day 7: 25/May/2017

Merge the design

Let us introduce Chat Module

Reference:

http://guides.rubyonrails.org/action_cable_overview.html

Method:

Step 1: make another clone for you to trace the command.

Step 2: Introduce Chat Sample / gem update

Confirm that every member can run the server.

Step 3: Use Okane’s sample to merge design.

How to introduce?

Step 1: make another clone

Purpose:

When you clone the project master, you have no chance to modify the rails project by yourself. So, make another clone to try the rails command on the copy.

```
mkdir trial
cd trial
git clone https://github.com/webdbhosei/ScopsOwl.git
```

Gemfile update

Last week, I had added the following

```
# gem 'i18n-tasks'
```

This had caused another gem’s installation problem, so I had commented out.

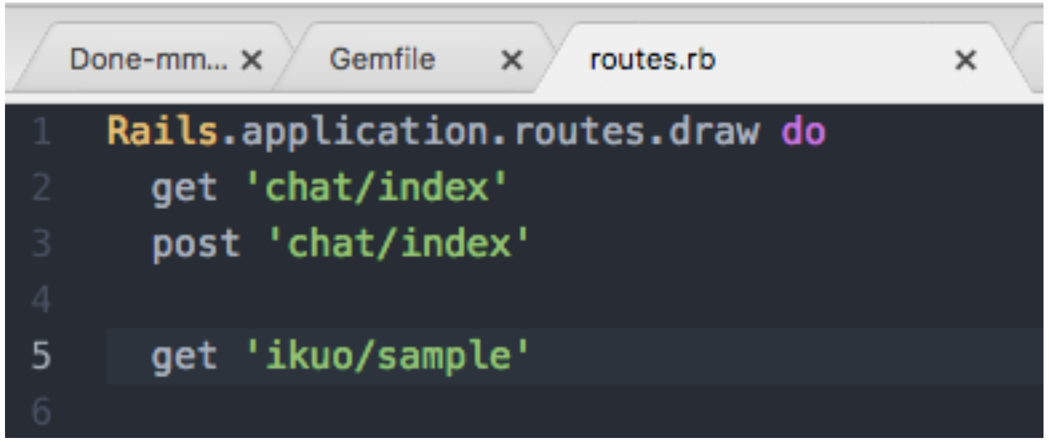
Step 2 : Chat introduction

Generate the chat controller

```
rails g controller chat index
rails g channel chat
```

Modify the routing Path

```
get 'chat/index'
post 'chat/index'
```

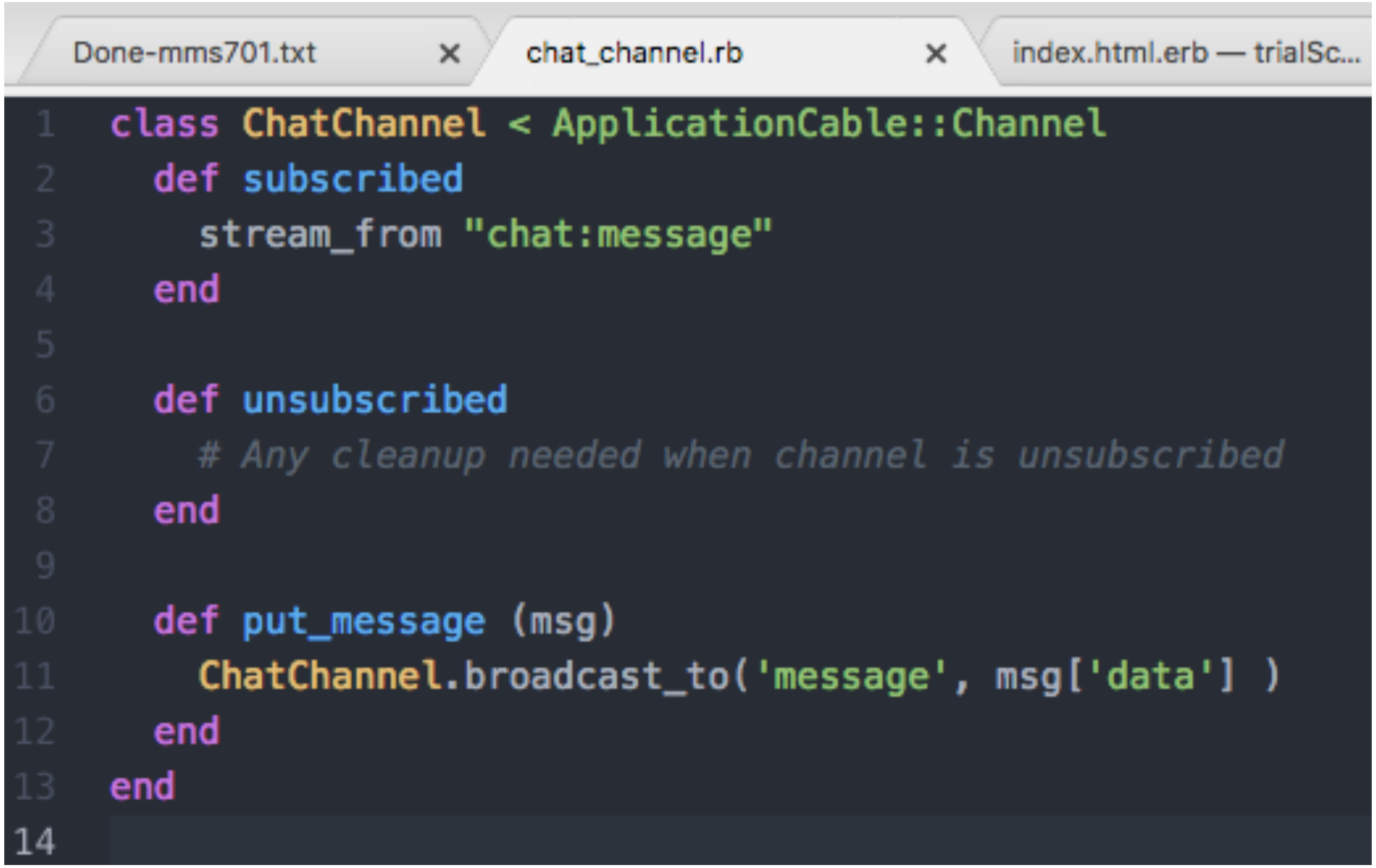


```ruby
Rails.application.routes.draw do
  get 'chat/index'
  post 'chat/index'

  get 'ikuo/sample'

```

app/channels/chat_channel.rb

```
class ChatChannel < ApplicationCable::Channel
  def subscribed
    stream_from "chat:message"
  end

  def unsubscribed
    # Any cleanup needed when channel is unsubscribed
  end

  def put_message (msg)
    ChatChannel.broadcast_to('message', msg['data'] )
  end
end
```

app/channels/chat_channel.rb

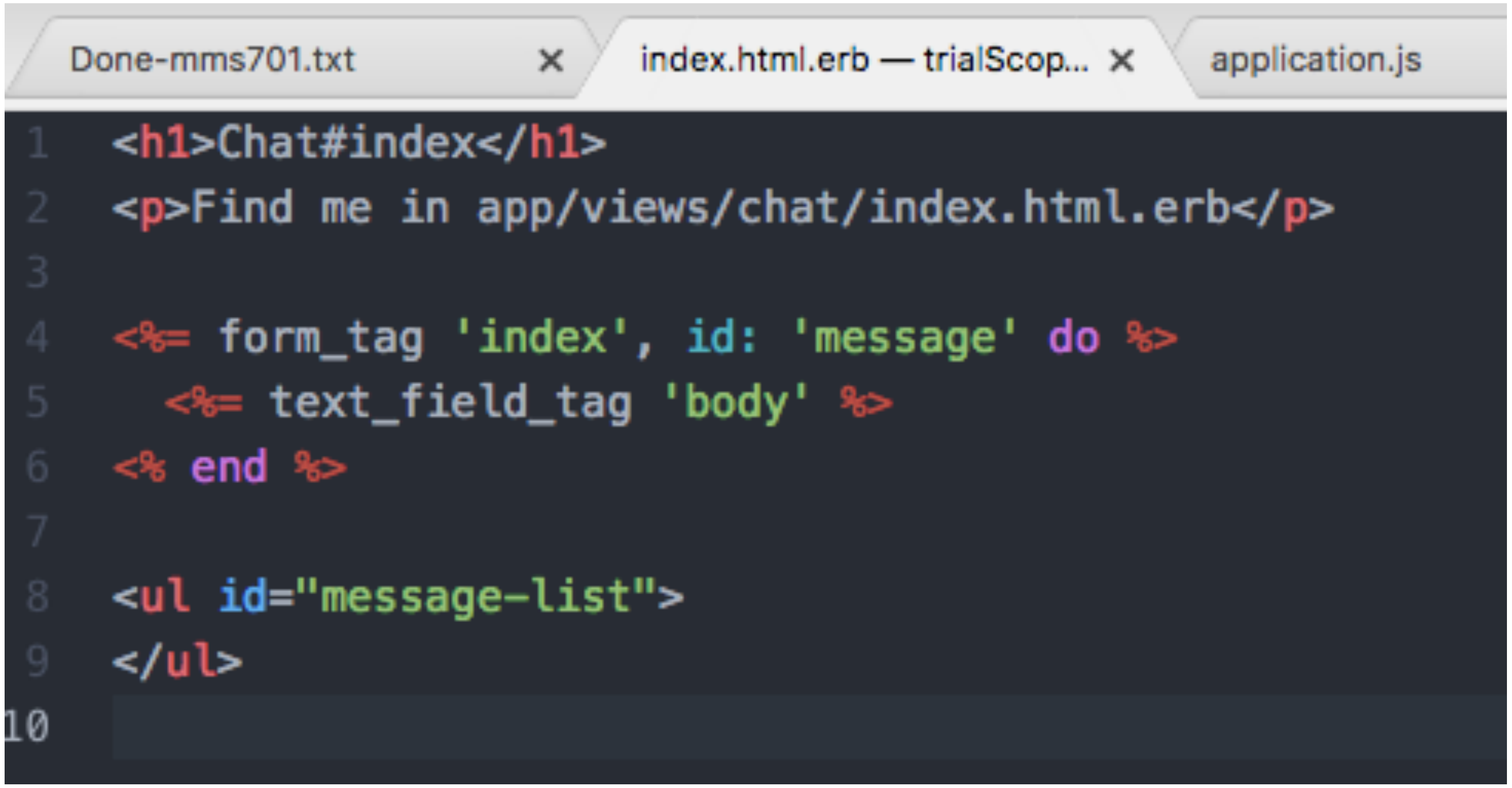


```ruby
class ChatChannel < ApplicationCable::Channel
  def subscribed
    stream_from "chat:message"
  end

  def unsubscribed
    # Any cleanup needed when channel is unsubscribed
  end

  def put_message (msg)
    ChatChannel.broadcast_to('message', msg['data'] )
  end
end

```

app/views/chat/index.html.erb

```
<h1>Chat#index</h1>
<p>Find me in app/views/chat/index.html.erb</p>

<%= form_tag 'index', id: 'message' do %>
  <%= text_field_tag 'body' %>
<% end %>

<ul id="message-list">
</ul>
```

app/views/chat/index.html.erb

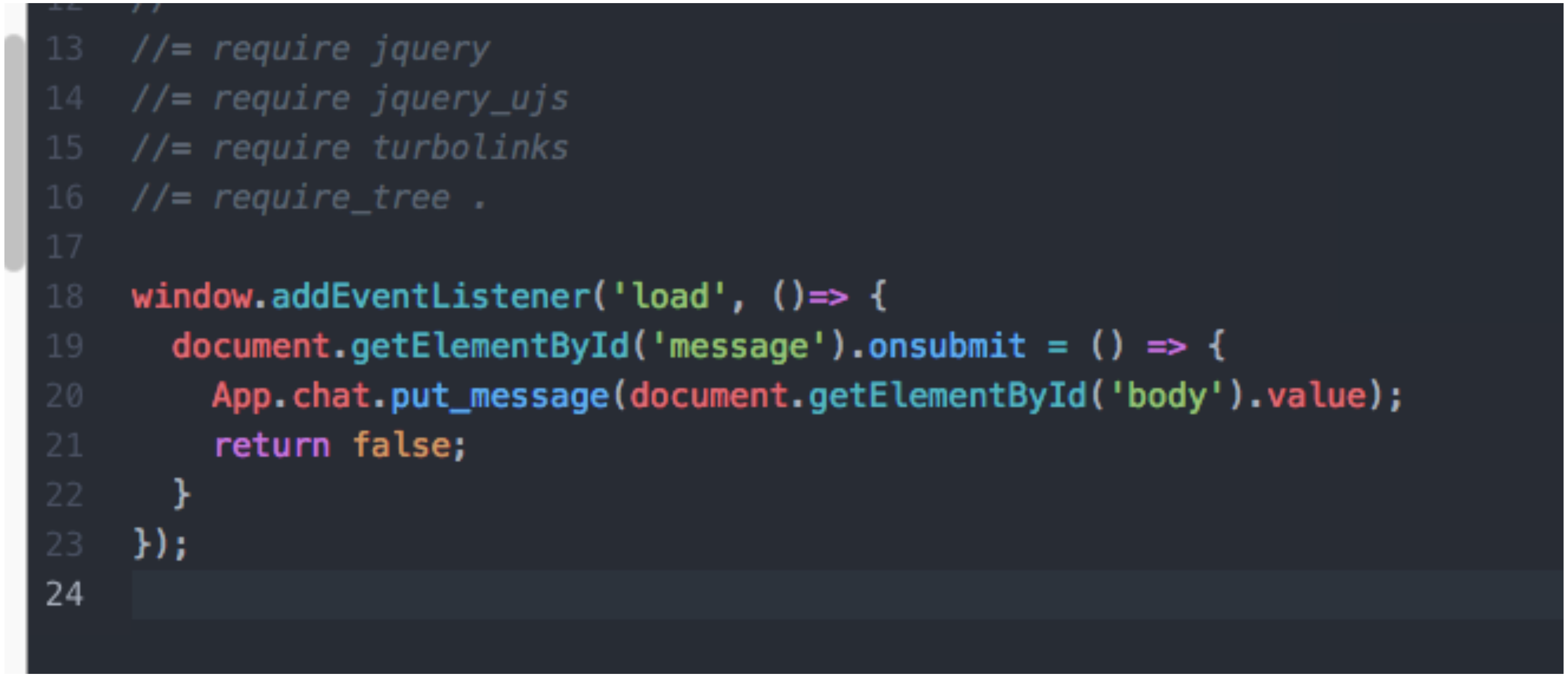


```
<h1>Chat#index</h1>
<p>Find me in app/views/chat/index.html.erb</p>

<%= form_tag 'index', id: 'message' do %>
  <%= text_field_tag 'body' %>
<% end %>

<ul id="message-list">
</ul>

```

app/assets/javascripts/application.js

```
window.addEventListener('load', ()=> {
  document.getElementById('message').onsubmit = () => {
    App.chat.put_message(document.getElementById('body').value);
    return false;
  }
});
```

app/assets/javascripts/application.js

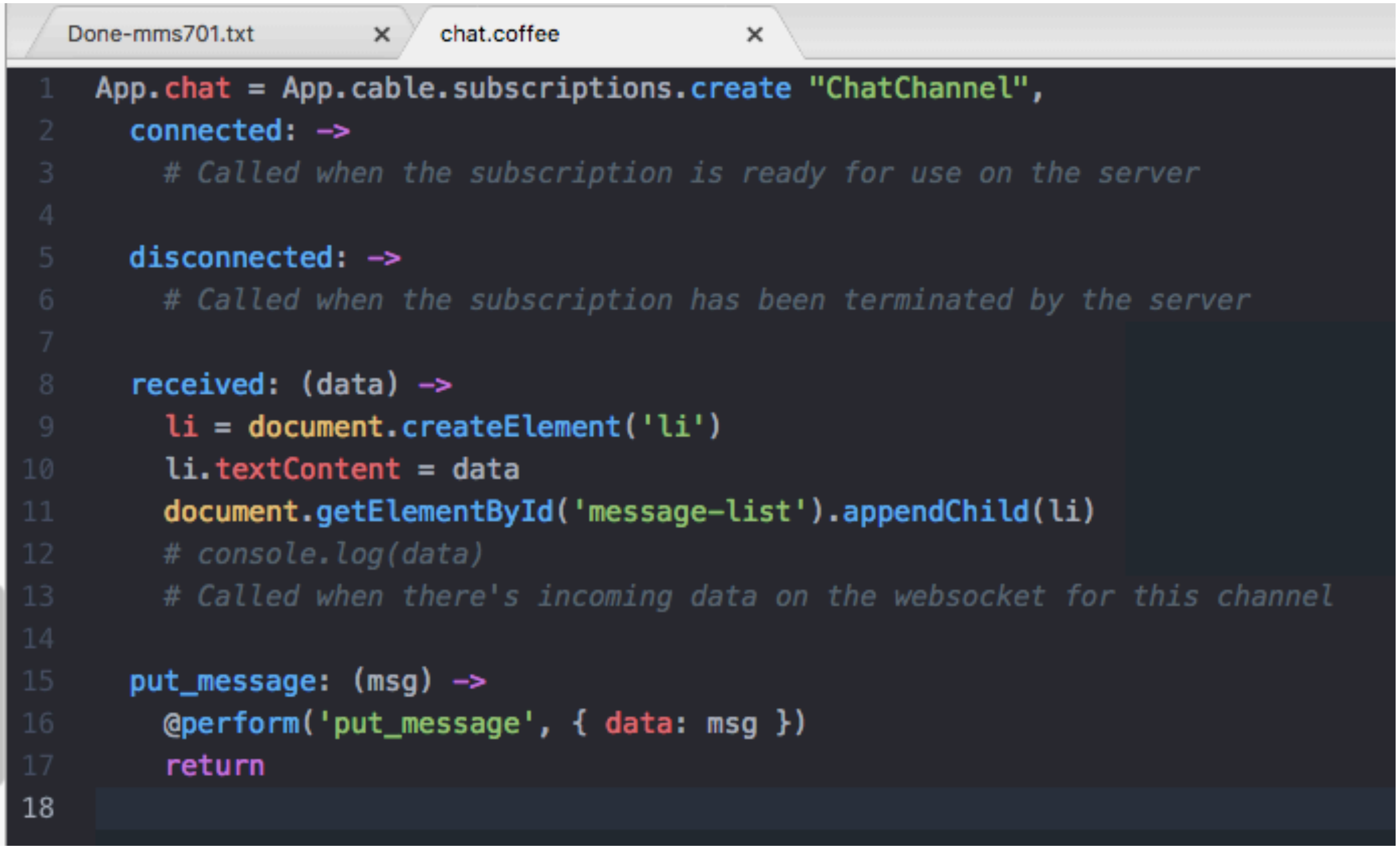


```
//= require jquery
//= require jquery_ujs
//= require turbolinks
//= require_tree .

window.addEventListener('load', ()=> {
  document.getElementById('message').onsubmit = () => {
    App.chat.put_message(document.getElementById('body').value);
    return false;
  }
});
```

app/assets/javascriopts/channels/chat.coffee

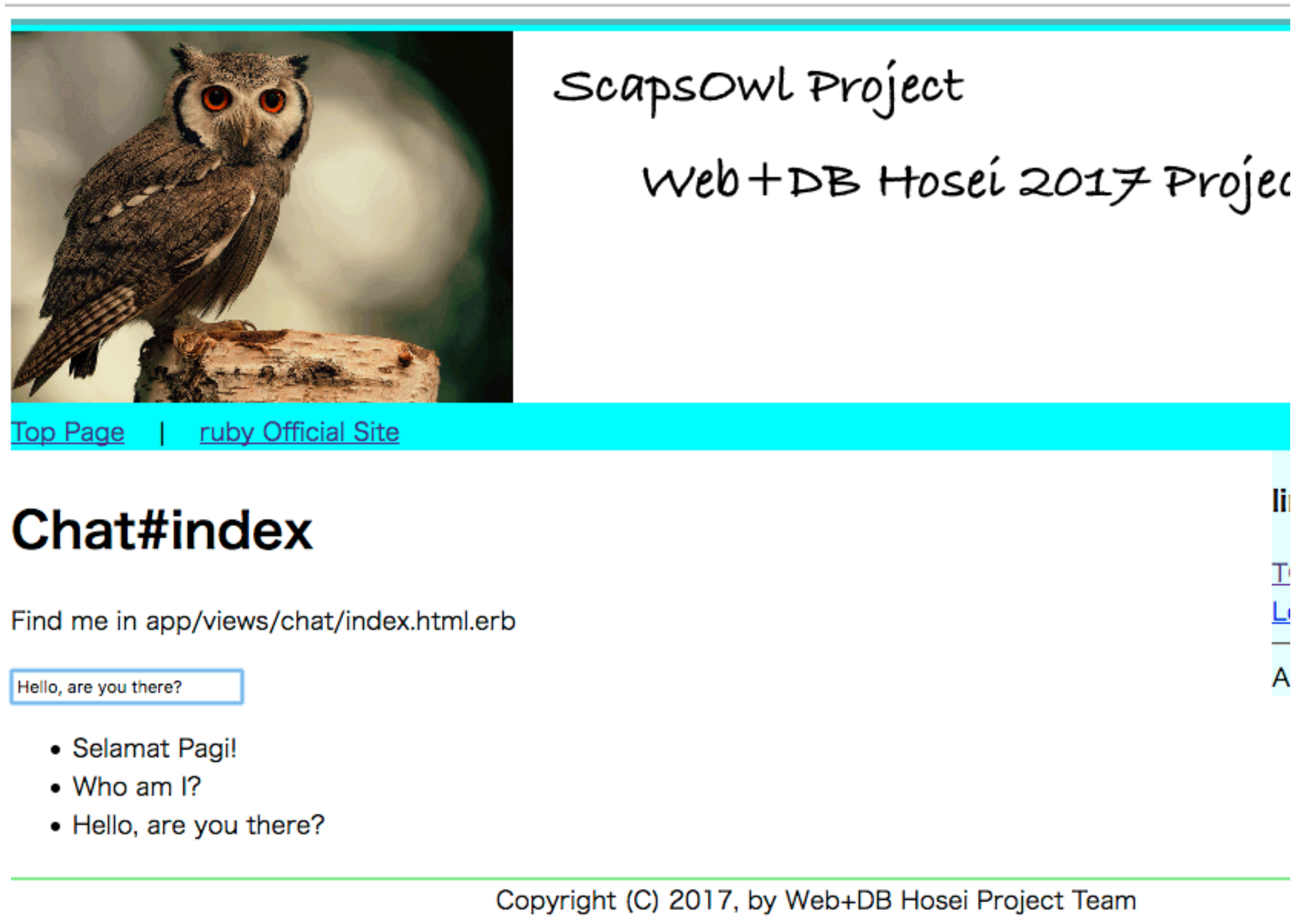
```
App.chat = App.cable.subscriptions.create "ChatChannel",
  connected: ->
    # Called when the subscription is ready for use on the server

  disconnected: ->
    # Called when the subscription has been terminated by the server

  received: (data) ->
    li = document.createElement('li')
    li.textContent = data
    document.getElementById('message-list').appendChild(li)
    # console.log(data)
    # Called when there's incoming data on the websocket for this channel

  put_message: (msg) ->
    @perform('put_message', { data: msg })
    return
```

app/assets/javascriopts/channels/chat.coffee



```coffee
App.chat = App.cable.subscriptions.create "ChatChannel",
  connected: ->
    # Called when the subscription is ready for use on the server

  disconnected: ->
    # Called when the subscription has been terminated by the server

  received: (data) ->
    li = document.createElement('li')
    li.textContent = data
    document.getElementById('message-list').appendChild(li)
    # console.log(data)
    # Called when there's incoming data on the websocket for this channel

  put_message: (msg) ->
    @perform('put_message', { data: msg })
    return

```

Test Run



Chat#index

Find me in app/views/chat/index.html.erb

Hello, are you there?

- Selamat Pagi!
- Who am I?
- Hello, are you there?

Copyright (C) 2017, by Web+DB Hosei Project Team

Mission : Make this Chat a library

(1) Relocate this chat screen to the _right_bar if possible.

(2) Refacter this chat, to make this a library, so that other members can use.

Next week:

(1) Introduce login to identify the user.

(2) Bind the chat message to the speaker

(3) save the chat messages to database.

Step 3 : bind Okane’s sample

Everyone, generate the blank model, skeleton, and blank views in each project.

Do it one by one, to avoid the unnecessary conflict in our project.

Note: View designer will generate controller.

Hatuka-k will ‘pull’ those changes.

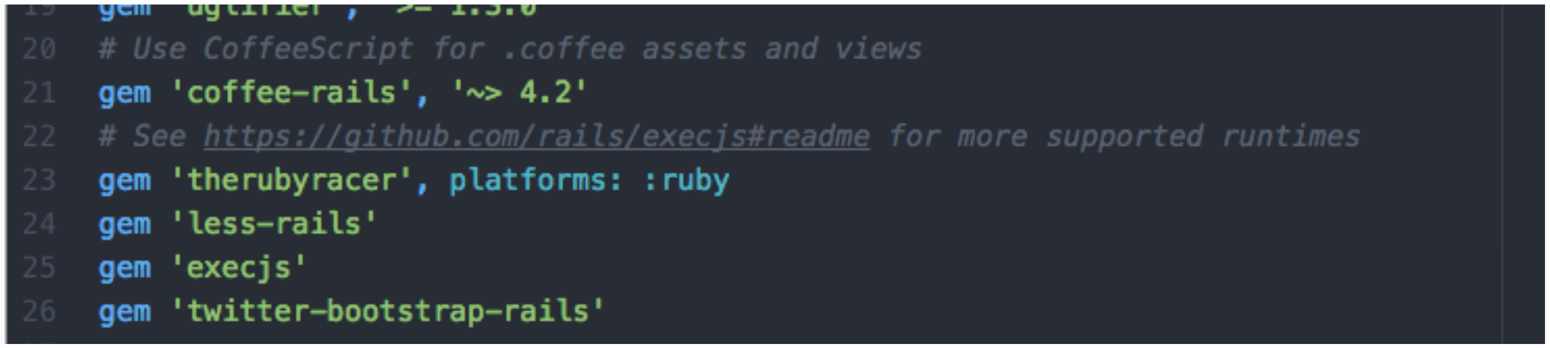
Bind bootstrap into rails

Update Gemfile

Add the following line;

```
gem 'less-rails'
gem 'execjs'
gem 'twitter-bootstrap-rails'
```

Then bundle install.



```
gem 'uglifier', '>= 1.3.0'
# Use CoffeeScript for .coffee assets and views
gem 'coffee-rails', '~> 4.2'
# See https://github.com/rails/execjs#readme for more supported runtimes
gem 'therubyracer', platforms: :ruby
gem 'less-rails'
gem 'execjs'
gem 'twitter-bootstrap-rails'
```

Install bootstrap

```
rails g bootstrap:install less
```

Now, check bootstrap classes

Also, check LESS, and SASS

Generate your own portal

```
rails g controller your_handle index
```

Then, submit pull request to me.

Let us try and see how it works.

Now I will(should) face with the routes.rb conflicts, if you successfully generated your own portal.