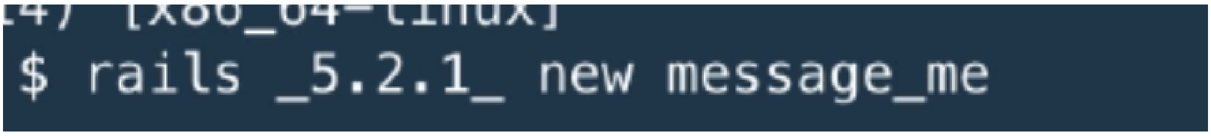
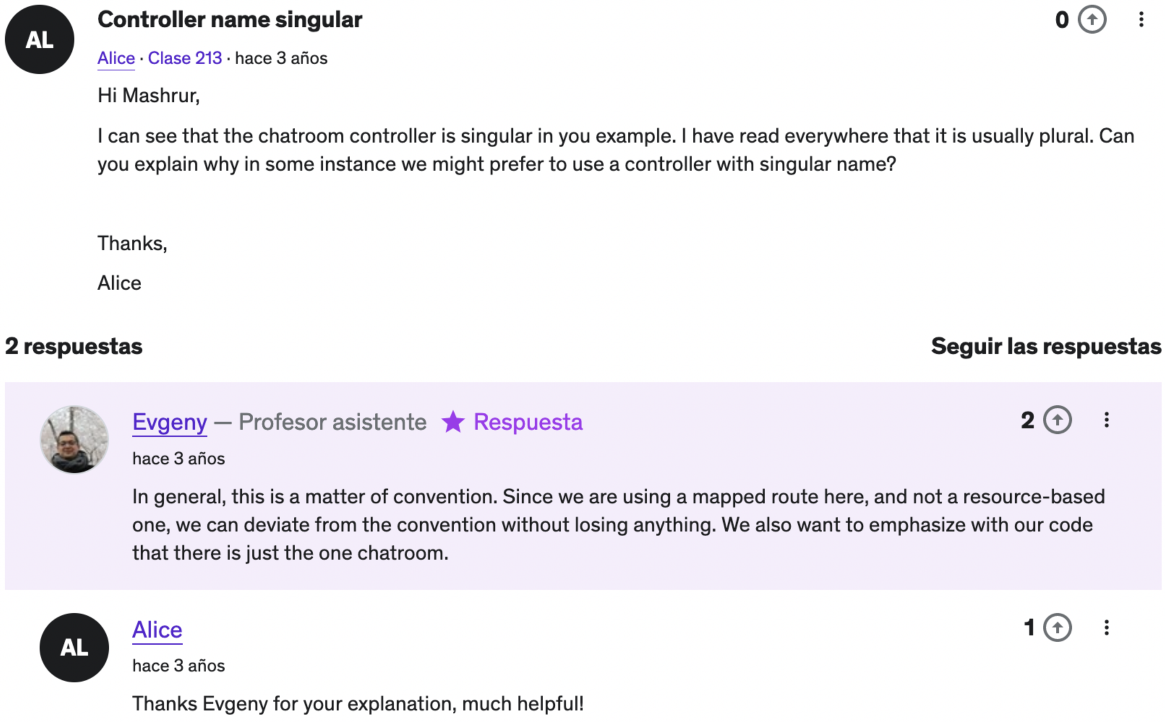


```
$ rails _5.2.1_ new message_me
```

You can create an app on that way also, specifying the rails version.

# 213. Task 2: Root and Login routes

## Routes names



**Controller name singular**

Alice · Clase 213 · hace 3 años

Hi Mashrur,

I can see that the chatroom controller is singular in you example. I have read everywhere that it is usually plural. Can you explain why in some instance we might prefer to use a controller with singular name?

Thanks,

Alice

**2 respuestas** **Seguir las respuestas**

Evgeny — Profesor asistente ★ Respuesta

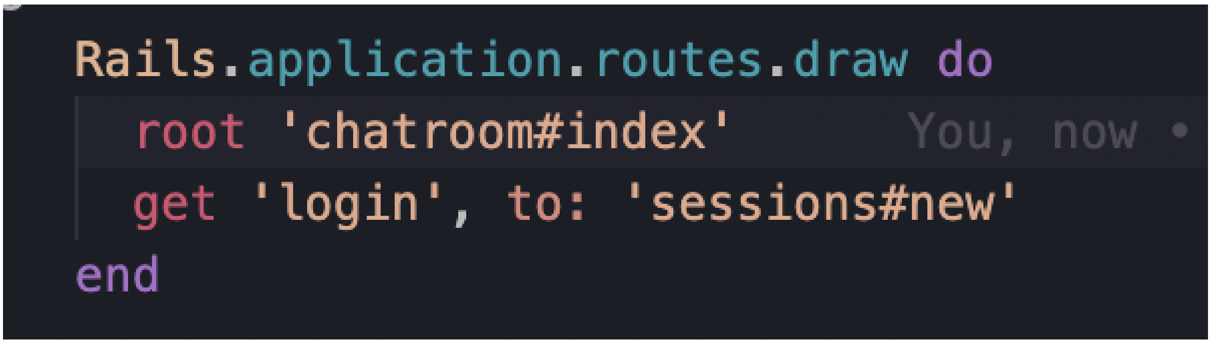
hace 3 años

In general, this is a matter of convention. Since we are using a mapped route here, and not a resource-based one, we can deviate from the convention without losing anything. We also want to emphasize with our code that there is just the one chatroom.

Alice

hace 3 años

Thanks Evgeny for your explanation, much helpful!

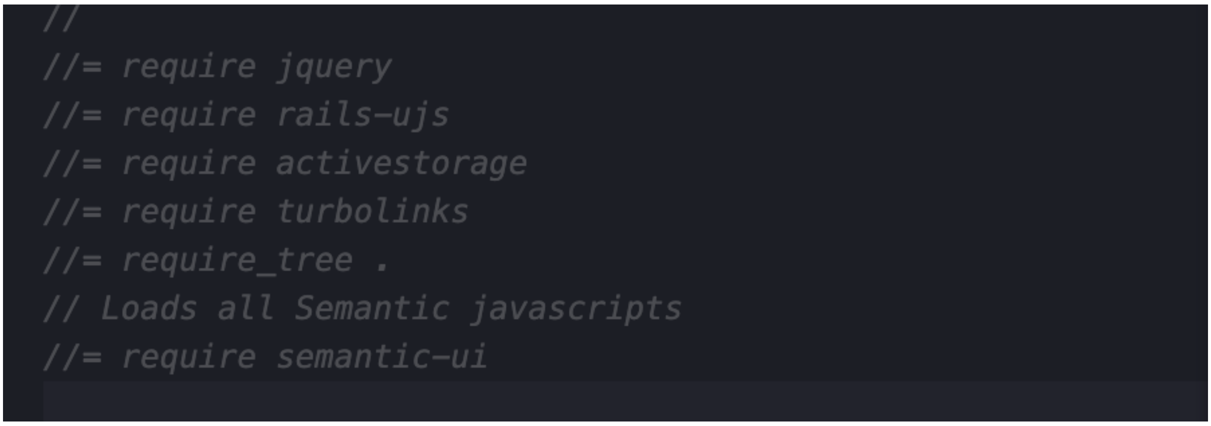


```
Rails.application.routes.draw do
  root 'chatroom#index'      You, now •
  get 'login', to: 'sessions#new'
end
```

# 214. Install Semantic-UI for front-end

We create a new custom css on styles_directory and import semantic.

Add to application.js

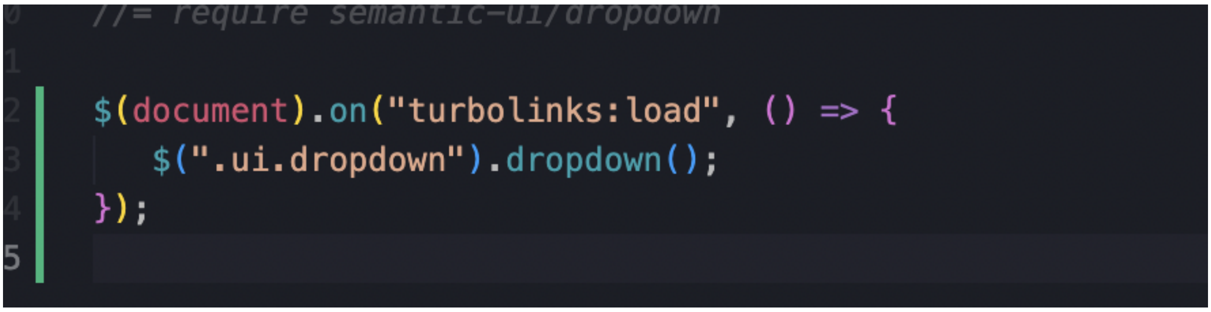


```
//
//= require jquery
//= require rails-ujs
//= require activestorage
//= require turbolinks
//= require_tree .
// Loads all Semantic javascripts
//= require semantic-ui
```

Add to application.css
-You don't need to add anything to application.css

## 216. Enable dropdown functionality and create nav partial

Semantic UI needs to initialize dropdowns trhought code.
On Application.js

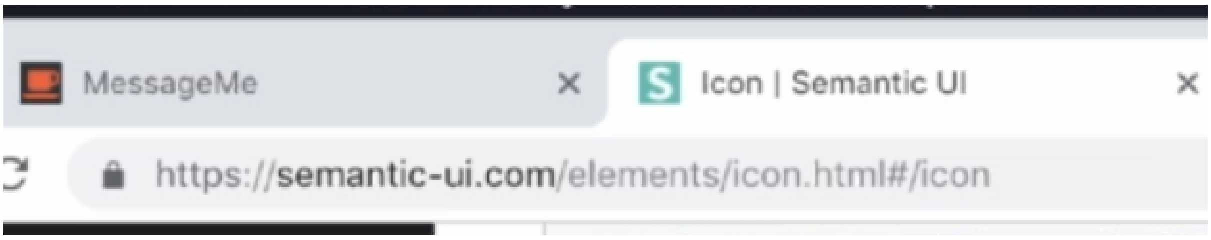


```
//= require semantic-ui/dropdown

$(document).on("turbolinks:load", () => {
    $(".ui.dropdown").dropdown();
});
```

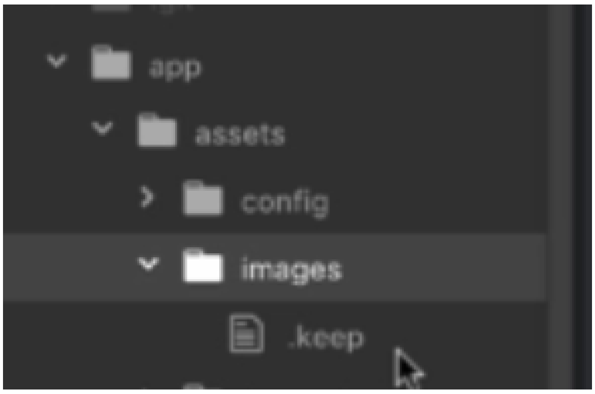
Normalmente quieres esperar a que el DOM termine de cargar, así que usamos:
$( document ).ready
Pero en este caso como rails usa turbo links y nos los vamos a desactivar entonces esperamos a los turbolinks.

## 217. Add favicon

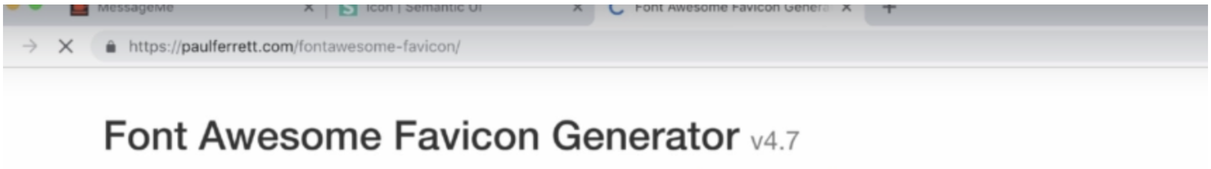
Semantic icons, uses font-awesome icons



Favicon should be added into this route



Use this page to tranform font awesome icons into favicons.

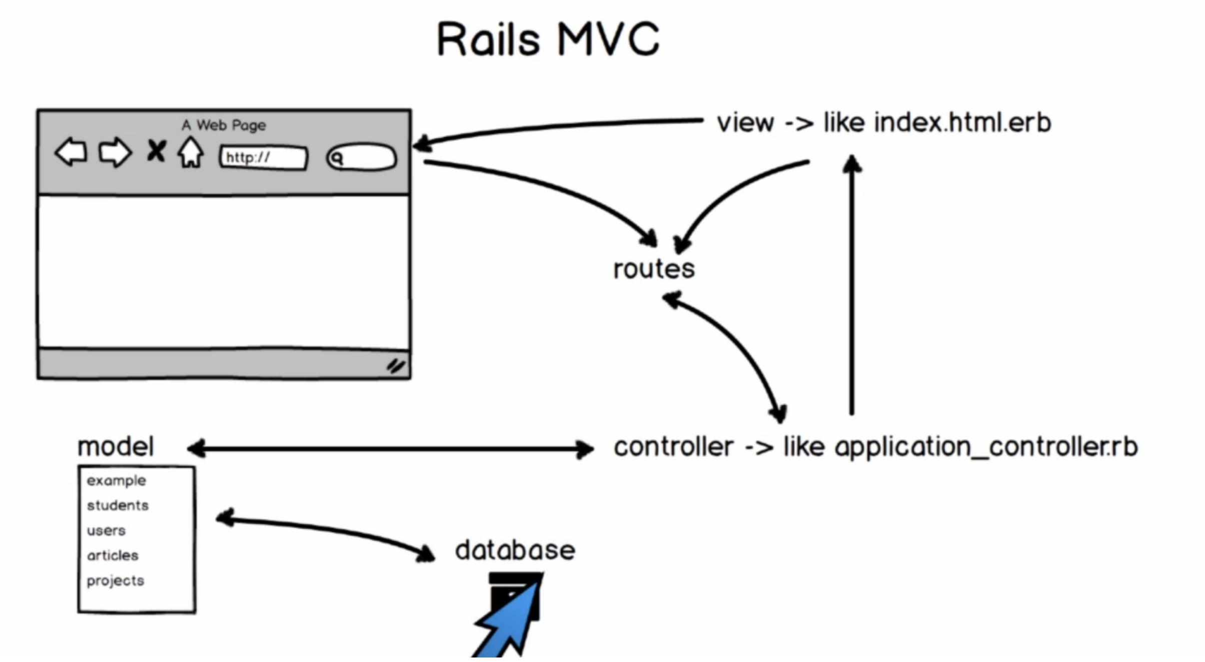


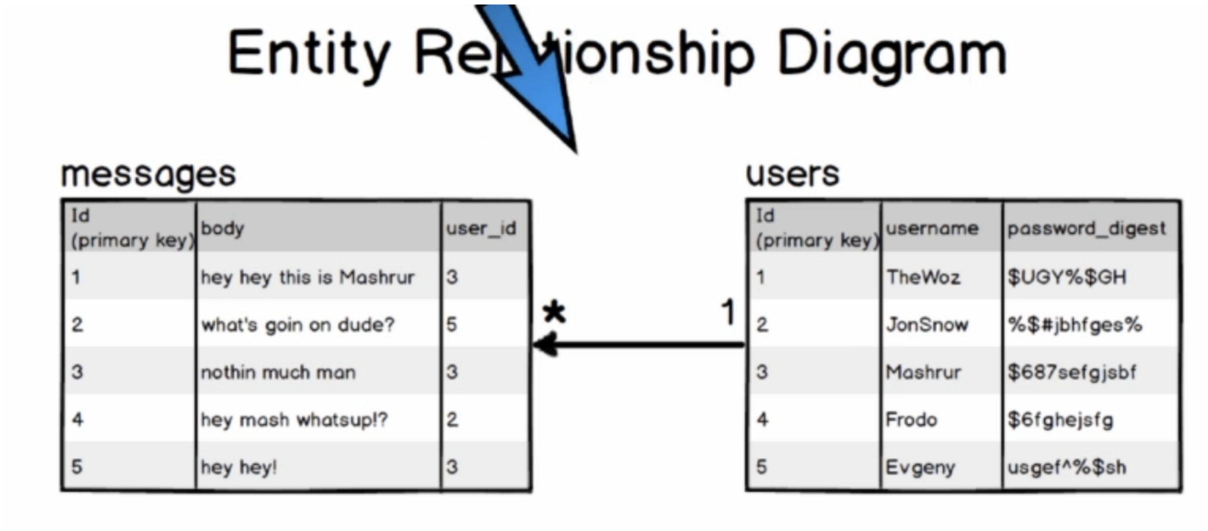
https://paulferrett.com/fontawesome-favicon/

Font Awesome Favicon Generator v4.7

To add the icon, you write: <%= favicon_link_tag %> on application.html.erb head tag.

# 221. Explore the back-end design





messages

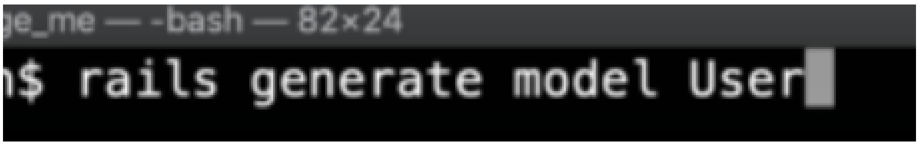
| Id (primary key) | body | user_id |
|---|---|---|
| 1 | hey hey this is Mashrur | 3 |
| 2 | what's goin on dude? | 5 |
| 3 | nothin much man | 3 |
| 4 | hey mash whatsup!? | 2 |
| 5 | hey hey! | 3 |

users

| Id (primary key) | username | password_digest |
|---|---|---|
| 1 | TheWoz | $UGY%$GH |
| 2 | JonSnow | %$#jbhfges% |
| 3 | Mashrur | $687sefgjsbf |
| 4 | Frodo | $6fghejsfg |
| 5 | Evgeny | usgef^%$sh |

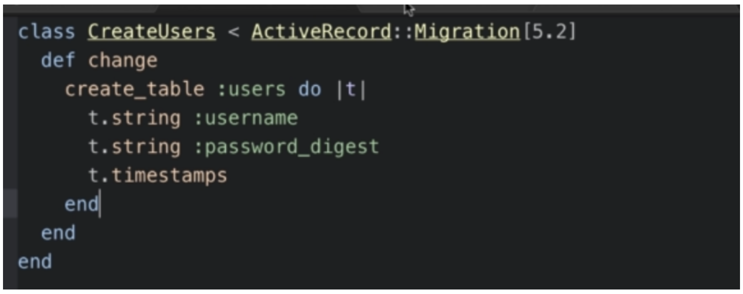
## 222. Task 4: Build User resource

Shortcut to generate user model and migration file



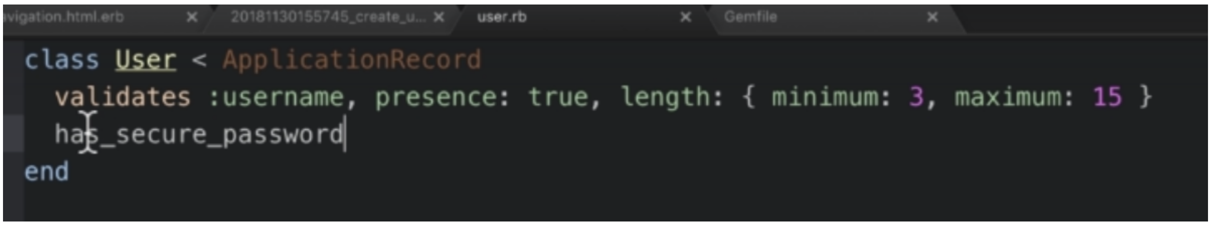
```
n$ rails generate model User
```

**On model we need to use password digest as a column name so the automatic hash can work**



```
class CreateUsers < ActiveRecord::Migration[5.2]
  def change
    create_table :users do |t|
      t.string :username
      t.string :password_digest
      t.timestamps
    end
  end
end
```

**On user model we have to add property has_secure_password so this automatic hash can work**



```
class User < ApplicationRecord
  validates :username, presence: true, length: { minimum: 3, maximum: 15 }
  has_secure_password
end
```

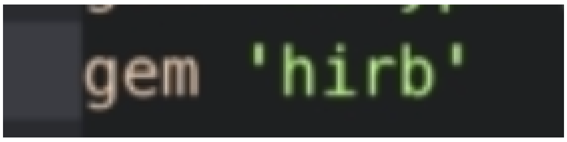
has_secure_password is used to encrypt and authenticate passwords using BCrypt . It assumes the model has a column named `password_digest`.

https://www.bigbinary.com/blog/rails-6-allows-configurable-attribute-name-on-has_secure_password

## Seed the database

You can add default values to the database using the seed file under db.
**To execute it you use:**
rails db:seed

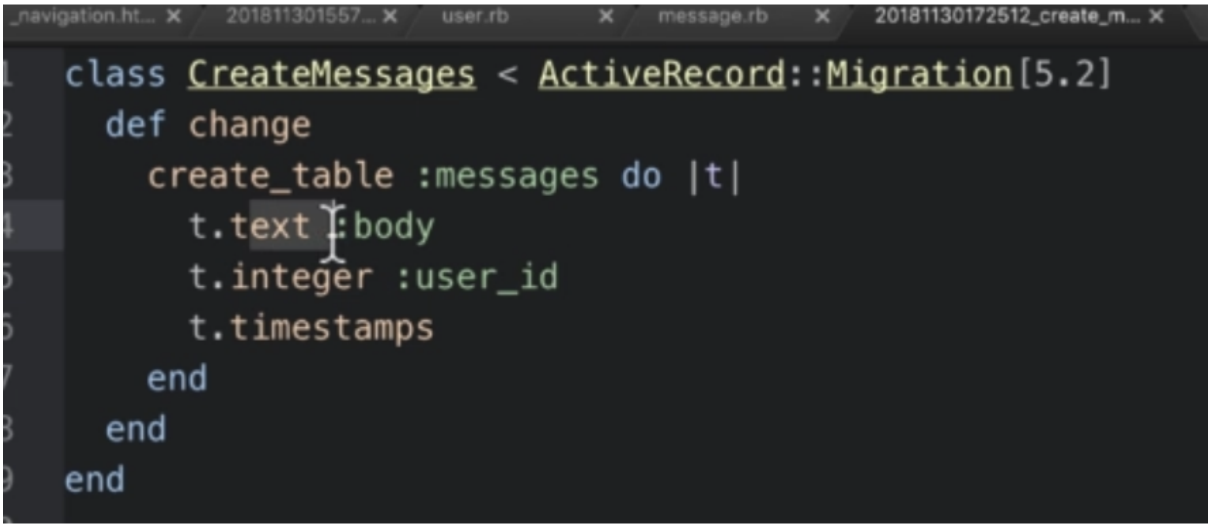
## Gem to see prettier queries on console



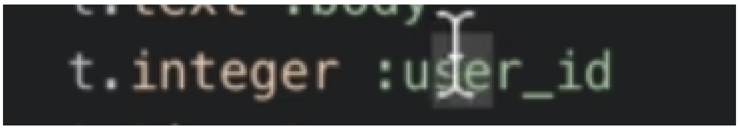
```
gem 'hirb'
```

You type on console :
Hirb.enable
Then you can try the queries.

# 223. Task 5: Build Message resource

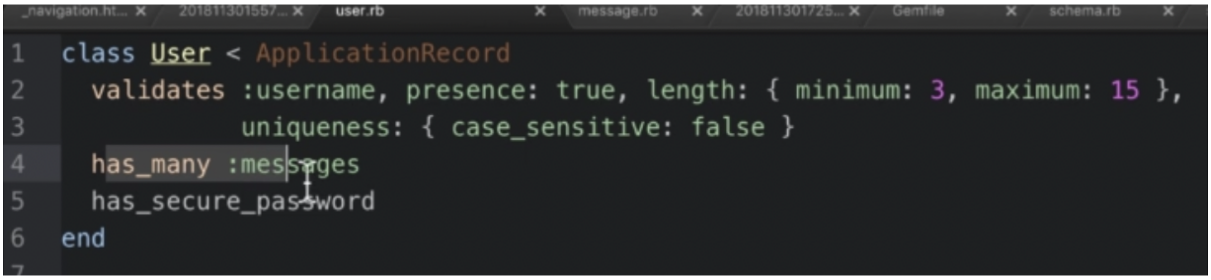


```
class CreateMessages < ActiveRecord::Migration[5.2]
  def change
    create_table :messages do |t|
      t.text :body
      t.integer :user_id
      t.timestamps
    end
  end
end
```



```
t.integer :user_id
```

Because we already have an user table, :user_id will be added automatically as a foreign key into the database.
**We still have to add the relationship into the model**

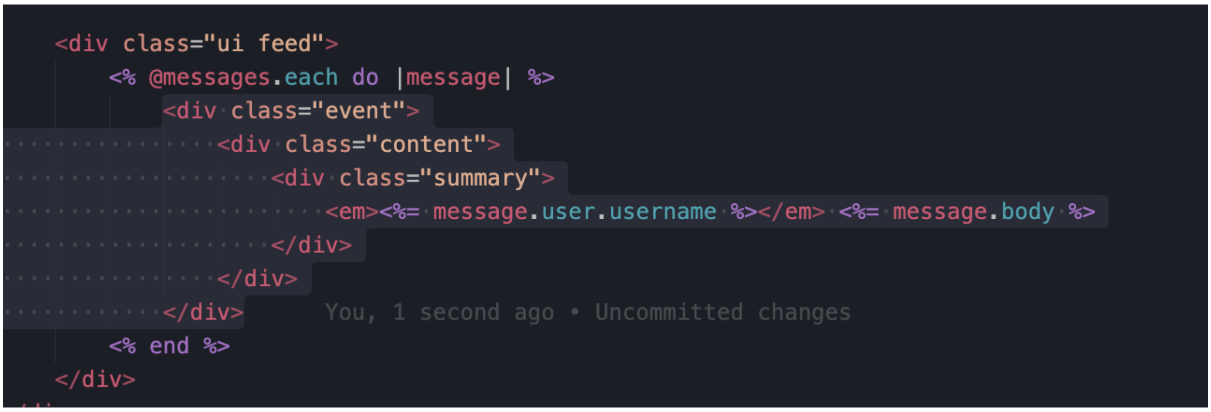


```
class User < ApplicationRecord
  validates :username, presence: true, length: { minimum: 3, maximum: 15 },
            uniqueness: { case_sensitive: false }
  has_many :messages
  has_secure_password
end
```

# 225. Add message partial and refactor some code

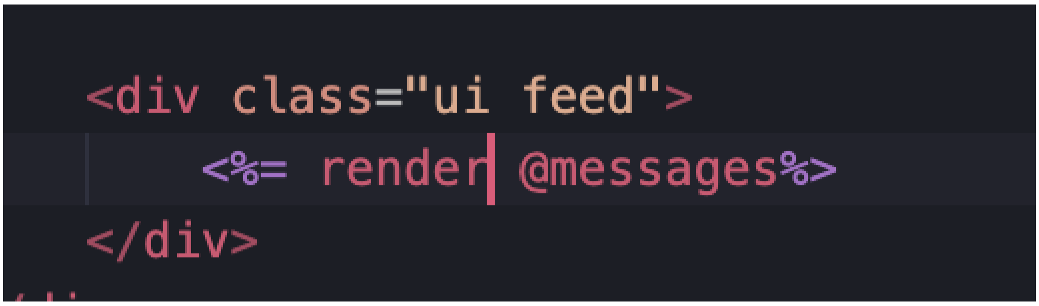
## Shortcut on foreach element.

Instead of this:



```
<div class="ui feed">
    <% @messages.each do |message| %>
        <div class="event">
            <div class="content">
                <div class="summary">
                    <em><%= message.user.username %></em> <%= message.body %>
                </div>
            </div>
        </div>      You, 1 second ago • Uncommitted changes
    <% end %>
</div>
```

You can use:



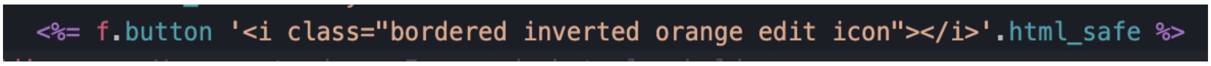
```
<div class="ui feed">
    <%= render @messages%>
</div>
```

This is because the “each” structure to render an item is used a lot, so rails developers develop this feature. If you have a model called message. Then if you render an array of these elements, rails will search for a directory in views called “messages” and search for a partial called “_message.html.erb” for example.

# 228. Enable flash messages display

# 230. Add messages from UI

## Add HTML to form helper

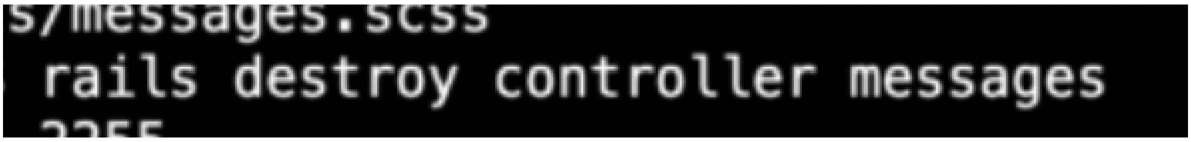


```
<%= f.button '<i class="bordered inverted orange edit icon"></i>'.html_safe %>
```

## Controller generator

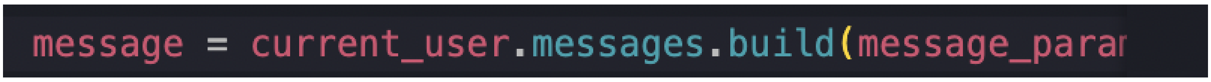
Create is a method
rails generate controller messages create

Remove controller



```
s/messages.scss
 rails destroy controller messages
```

Add message association to user automatically



```
message = current_user.messages.build(message_parar
```

# 231. Introduction to WebSocket protocol

On ruby the websocket protocol is available through something called the **action cable**. The action cable is something that ruby added and makes things more easy to implement websockets.

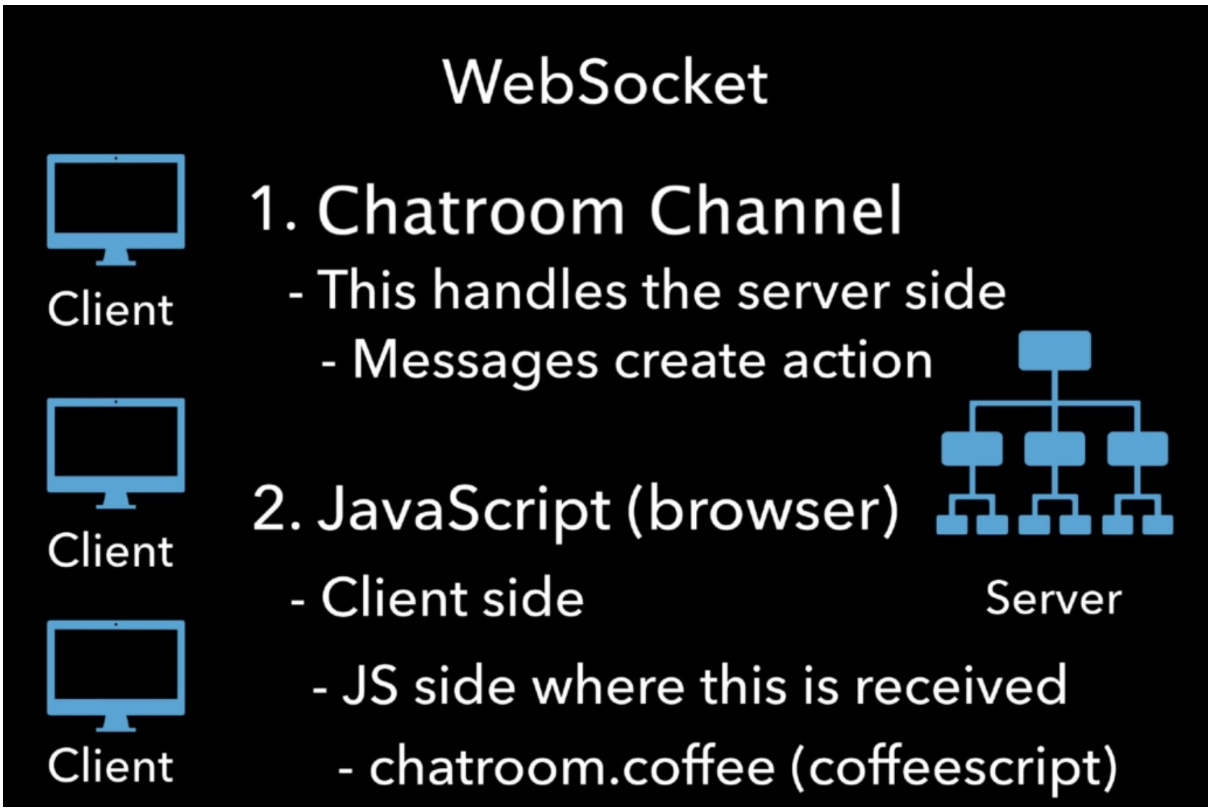
1. Clients make a connection to the server via socket request.
2. Communication becomes two duplex communication.

# 232. Implementing real-time with ActionCable overview

**Channel**
A channel is what manages this duplex communication.

We are going to create a chat room channel that is going to enable two way communication.

We can use this channel to broadcast messages to all listening users.

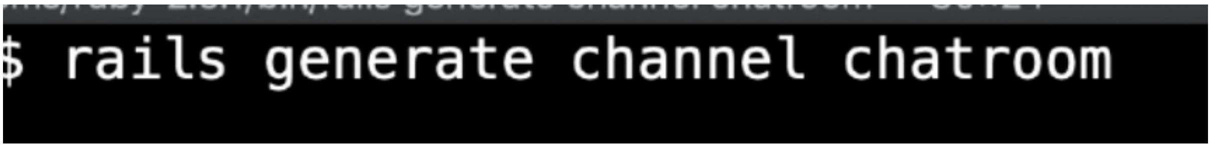


WebSocket

1. Chatroom Channel
   - This handles the server side
     - Messages create action

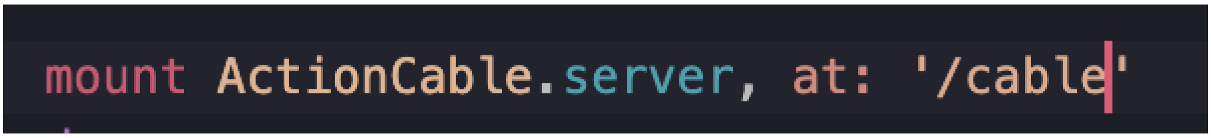
2. JavaScript (browser)
   - Client side
     - JS side where this is received
       - chatroom.coffee (coffeescript)

## 233. Generate a Chatroom channel



```
$ rails generate channel chatroom
```

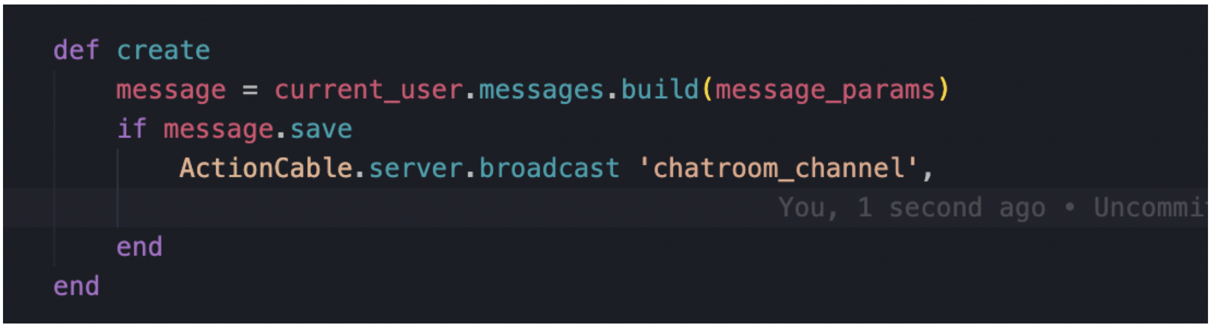
We still need a route for the websocket:



```
mount ActionCable.server, at: '/cable'
```

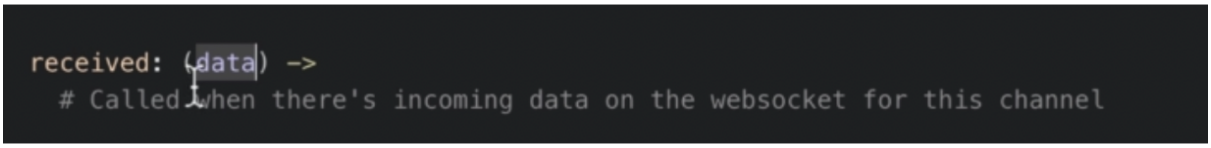
## 234. Modify and broadcast messages

All data broadcast from:



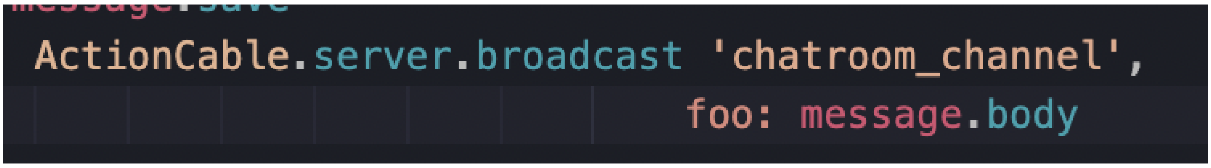
```
def create
    message = current_user.messages.build(message_params)
    if message.save
        ActionCable.server.broadcast 'chatroom_channel',

    end
end
```

Will be received on the coffee file



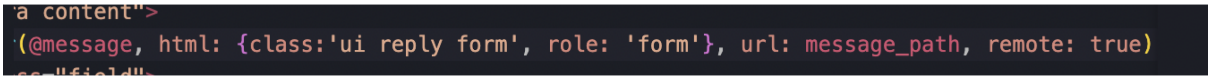
```
received: (data) ->
  # Called when there's incoming data on the websocket for this channel
```

**On the method from the action cable you can pass any key and value you want to send**



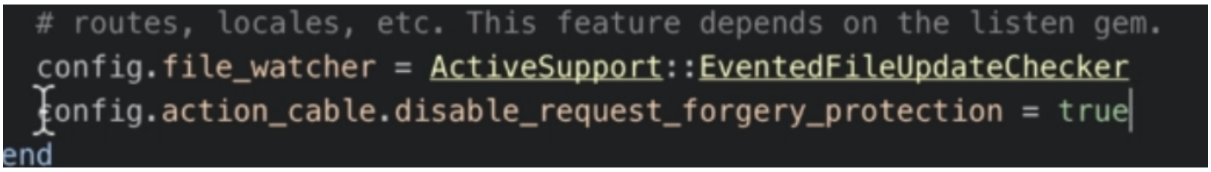
```
message.save
ActionCable.server.broadcast 'chatroom_channel',
                                        foo: message.body
```

If you want to send a message with the form remember to add remote property so it does not reload the page. Remote will submit the information using js.



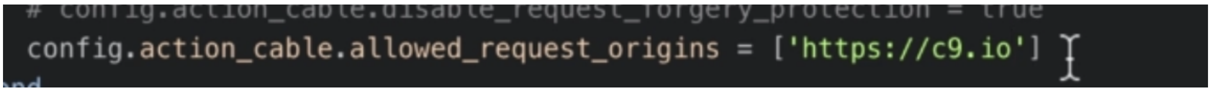
```
'a content">
(@message, html: {class:'ui reply form', role: 'form'}, url: message_path, remote: true)
ss="field">
```

If you were testing this on amazon or another web service, it wont work, you have to config some things, to me it work fast for development you can add this line on the development environment line.



```
  # routes, locales, etc. This feature depends on the listen gem.
  config.file_watcher = ActiveSupport::EventedFileUpdateChecker
  config.action_cable.disable_request_forgery_protection = true
end
```

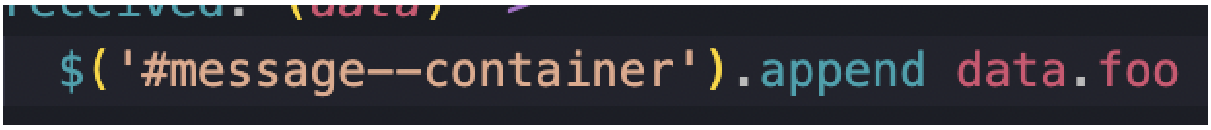
The other way, more safe, is to add the route of the server on the allowed request origins.



```
  # config.action_cable.disable_request_forgery_protection = true
  config.action_cable.allowed_request_origins = ['https://c9.io']
end
```

# 235. Display messages using partial

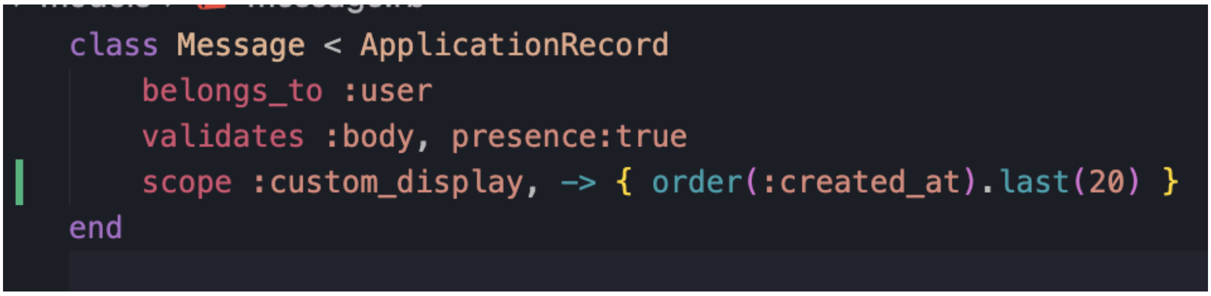
On coffee script you don't need to add parenthesis to methods, they tried to simplificate them



```
received: (data) ->
  $('#message--container').append data.foo
```

## 237. Beautify input box, functionality and create custom scope

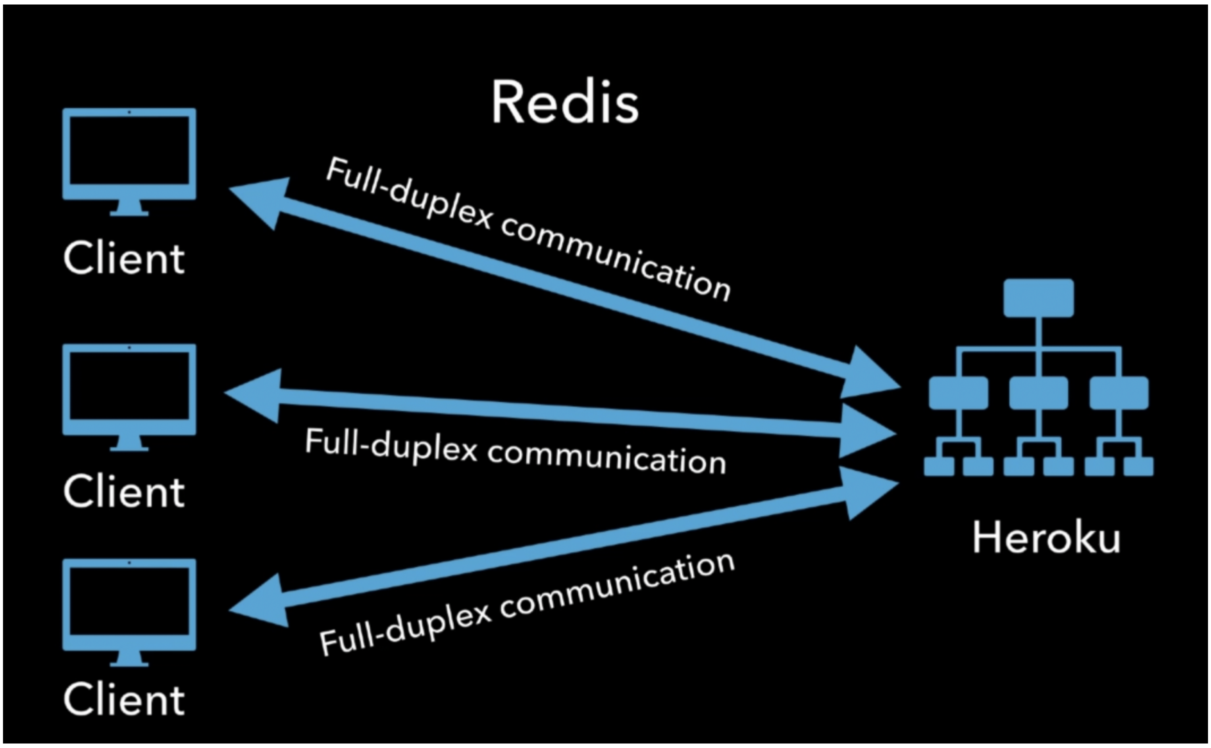
### Custom method on model



```ruby
class Message < ApplicationRecord
    belongs_to :user
    validates :body, presence:true
    scope :custom_display, -> { order(:created_at).last(20) }
end
```

Scopes are custom queries that you define inside your Rails models with the scope method.

## 238. Section Project


Redis
Client
Client
Client
Full-duplex communication
Full-duplex communication
Full-duplex communication
Heroku

On production, the communication of the websockets is managed by redis, you you will have to make some configuration related to it.

--------

The last app on the course is focused too on beginners so you can visit that if you have some gaps in your mind.