Ruby on Rails
web application dev

lesson 3

about me

thomas mak

thomas@cpttm.org.mo

agenda

job board example

managing gems

view helpers

model association

managing gems

gems

static pages

high_voltage

assuming now we need to add quite a lot of static pages.

gems

static pages

high_voltage

add the about / contact / faq to PagesController.

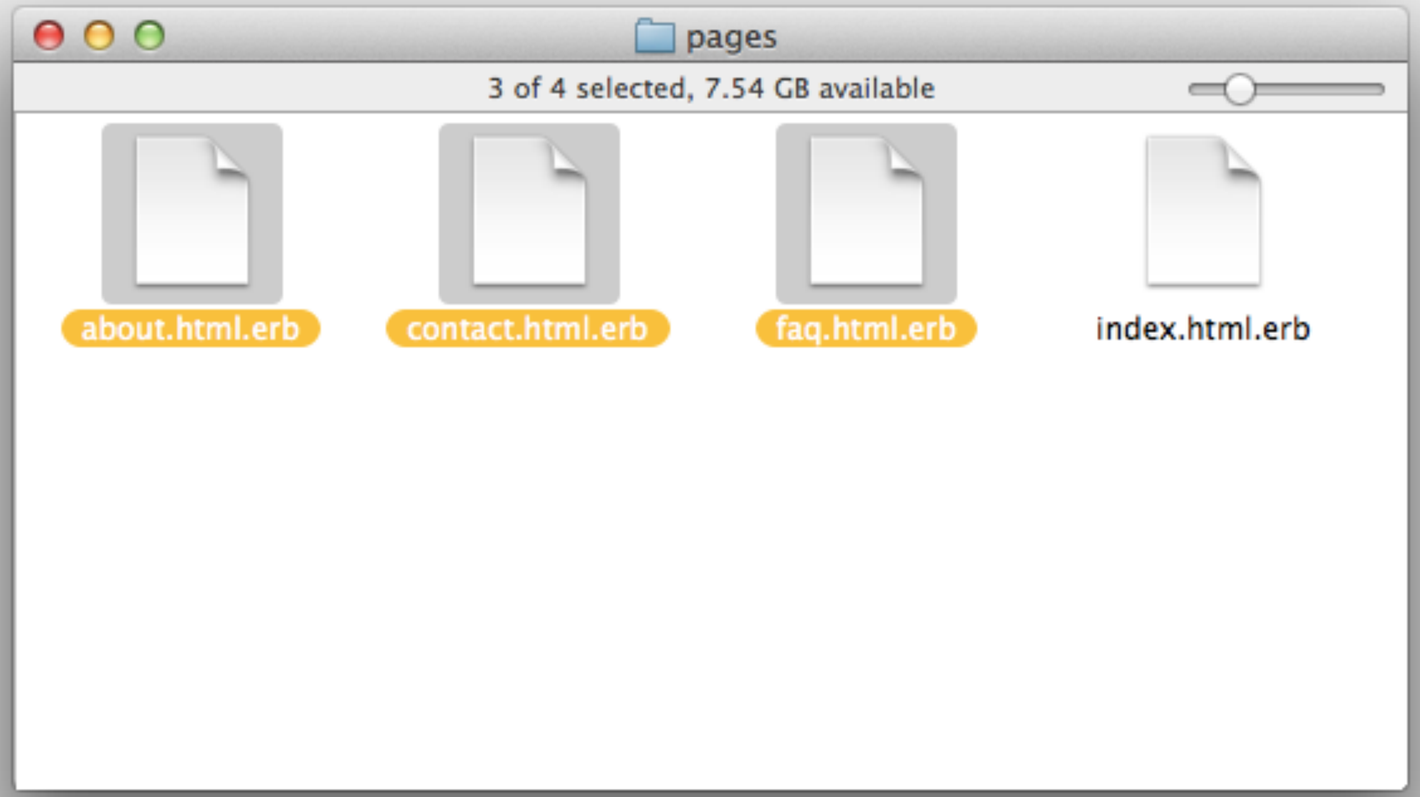
```ruby
class PagesController < ApplicationController
  def index
    @jobs = Job.order("id desc").limit(3)
  end

  def about
  end

  def contact
  end

  def faq
  end
end
```

gems

static pages

high_voltage

and then add the related views.



gems

static pages

high_voltage

then we add the routes to the routes.rb file.

```
get 'pages/about'
get 'pages/contact'
get 'pages/faq'
```

gems

static pages

high_voltage

adding so many code for static pages is such a slow process and repetitive.

gems

static pages

high_voltage

so we can refractor the common code into library – gem.

gems

static pages

high_voltage

high_voltage

https://github.com/thoughtbot/high_voltage/

thoughtbot/high_voltage

https github.com/thoughtbot/high_voltage Reader

High Voltage build passing

Rails engine for static pages.

... but be careful. Danger!

Static pages?

Yeah, like "About us", "Directions", marketing pages, etc.

Installation

```
$ gem install high_voltage
```

Include in your Gemfile:

```
gem 'high_voltage'
```

high_voltage gem

gems

static pages

high_voltage

add the following lines to **Gemfile** file.

```
# static pages
gem 'high_voltage'
```

gems

static pages

high_voltage

let bundler install the added gems.

```
$ bundle install
```

gems

static pages

high_voltage

remove the about / contact / faq from PagesController.

```
class PagesController < ApplicationController
  def index
    @jobs = Job.order("id desc").limit(3)
  end
end
```

gems

static pages

high_voltage

then we undo the routes in the routes.rb file.

```
get 'pages/about'
get 'pages/contact'
get 'pages/faq'
```

bundle install

gems

static pages

high_voltage

bundle update gem_name

bundle

gems

static pages

high_voltage

bundle install

install / remove gems

bundle update gem_name

update gems to latest allowed version

bundle

use gems listed in Gemfile.lock

view helpers

view helpers

link_to

form_fields

add the following lines to views/pages/index.html.erb

```
<%= link_to "About", "/pages/about" %>
<%= link_to "Contact", page_path('contact') %>
<%= link_to "FAQ", page_path('faq') %>
```

view helpers

link_to

form_fields

usage of high_voltage linking.

Usage

Write your static pages and put them in the RAILS_ROOT/app/views/pages directory.

```
$ mkdir app/views/pages
$ touch app/views/pages/about.html.erb
```

After putting something interesting there, you can link to it from anywhere in your app with

```
link_to 'About', page_path('about')
```

You can nest pages in a directory structure, if that makes sense from a URL perspective fo

```
link_to 'Q4 Reports', page_path('about/corporate/policies/HR/en_US/biz/sales/Quart
```

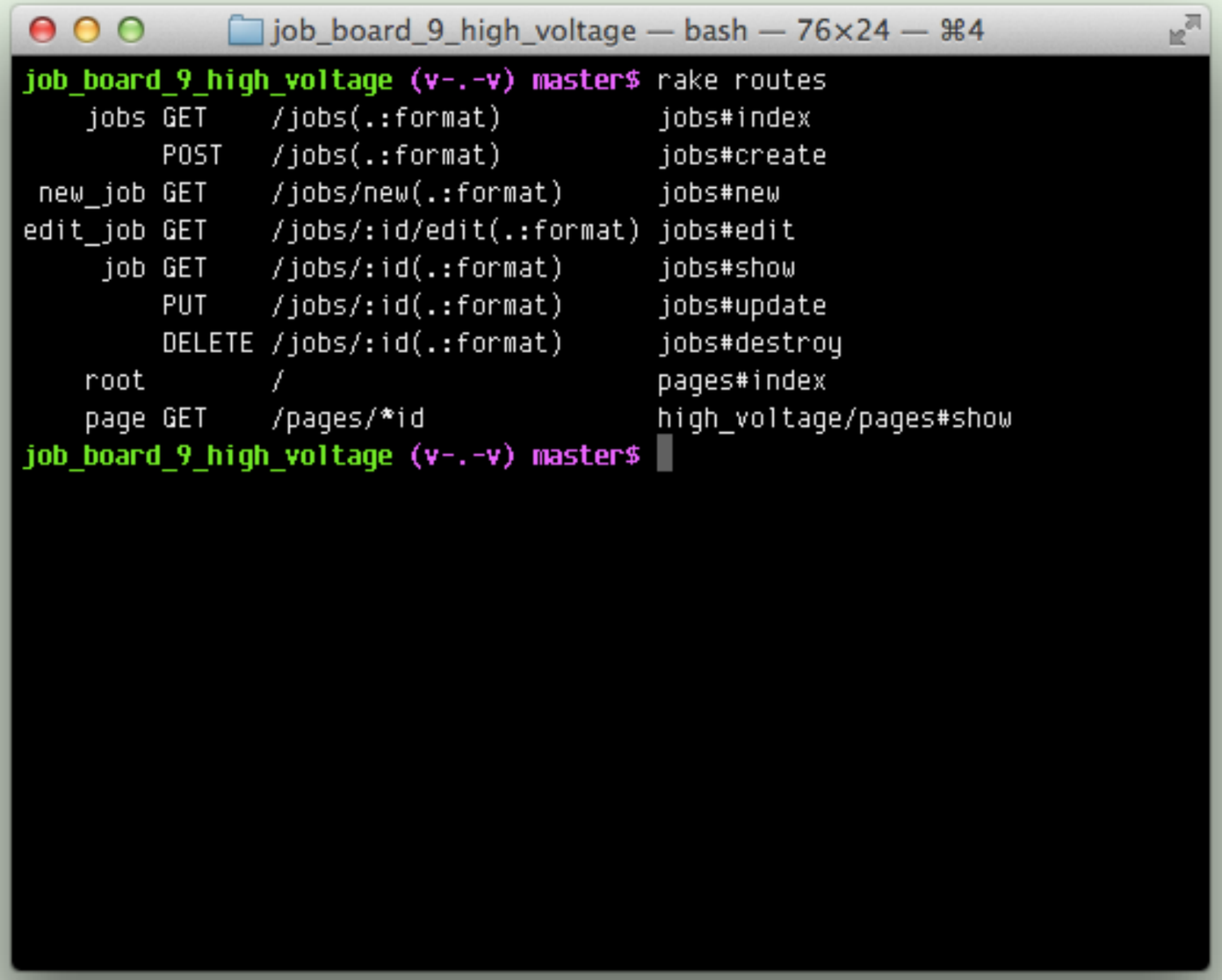
Bam.

view helpers

link_to

form_fields

fetching all the available paths by **rake routes**.



view helpers

link_to

form_fields

```
form_for obj do |f|
    f.text_field
end
```

view helpers

link_to

form_fields

text_field
password_field
email_field
hidden_field
text_area

new in rails 4

date_field
time_field

view helpers

link_to

form_fields

the generated html code from form helper.

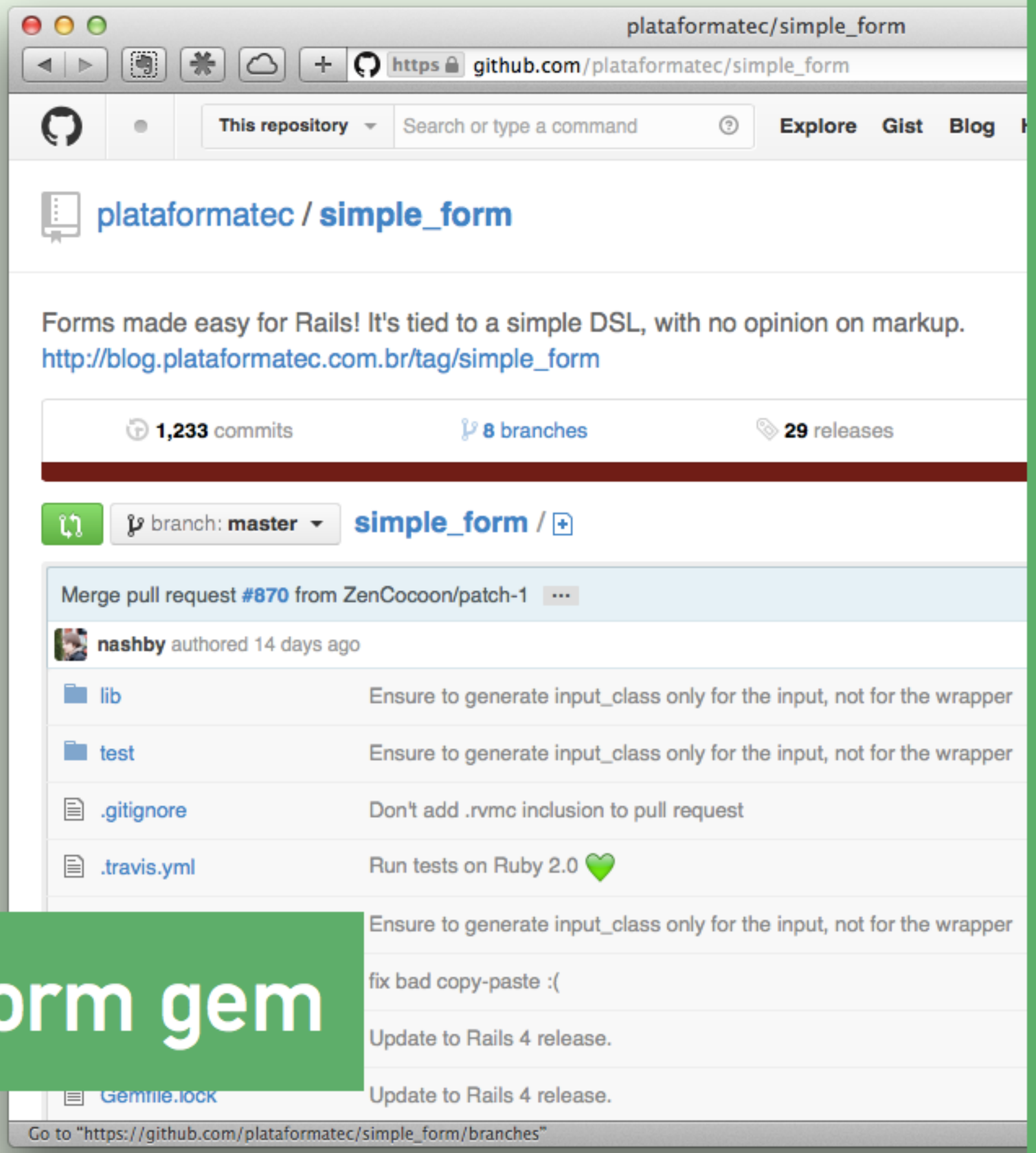
```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body style>
    ▶<nav>…</nav>
     <h1>New job</h1>
    ▼<form accept-charset="UTF-8" action="/jobs" class="new_job" id="new_job" method="post">
      ▶<div style="margin:0;padding:0;display:inline">…</div>
      ▼<div class="field">
         <label for="job_title">Title</label>
         <br>
       ▶<input id="job_title" name="job[title]" size="30" type="text">
       </div>
      ▼<div class="field">
         <label for="job_content">Content</label>
         <br>
       ▶<textarea cols="40" id="job_content" name="job[content]" rows="20">…</textarea>
       </div>
      ▼<div class="field">
         <label for="job_category">Category</label>
         <br>
       ▶<input id="job_category" name="job[category]" size="30" type="text" value="ruby">
       </div>
      ▶<div class="field">…</div>
      ▶<div class="field">…</div>
      ▶<div class="actions">…</div>
     </form>
     <a href="/jobs">Back</a>
     <script id="hiddenlpsubmitdiv" style="display: none;"></script>
   ▶<script>…</script>
  </body>
</html>
```

view helpers

link_to

form_fields

simple_form gem



view helpers

link_to

form_fields

add the following lines to **Gemfile** file.

```
# make the form simpler
gem 'simple_form'
```

view helpers

link_to

form_fields

let bundler install the added gems.
in addition, we need to run a gem specific init command.

```
$ bundle install
$ rails generate simple_form:install
```

view helpers

link_to

form_fields

here is what the simple_form does.

```
$ rails generate simple_form:install

SimpleForm 2 supports Twitter Bootstrap and Zurb
Foundation 3. If you want a configuration that is
compatible with one of these frameworks, then
please re-run this generator with --bootstrap
or --foundation as an option.
      create  config/initializers/simple_form.rb
       exist  config/locales
      create  config/locales/simple_form.en.yml
      create  lib/templates/erb/scaffold/_form.html.erb
```

view helpers

link_to

form_fields

now we can change the old form code,

```
<%= f.label :title %><br />
<%= f.text_field :title %>
```

to using the simple_form format:

```
<%= f.input :title %>
```

view helpers

link_to

form_fields

simple_form detects the field type from the model table in database.

model association

association nested comments into job

generate a new scaffold named job.

```
$ rails generate scaffold comments
    name:string content:text
```

the new code will make use of the simple_form gem.

association

comment belongs_to job

nested comments into job

job has_many comments

association nested comments into job

actually we need one more column in job table.

```
$ rails generate migration AddJobIdToComment
    job_id:integer
```

association nested comments into job

add the belongs_to line in comment.rb

```
class Comment < ActiveRecord::Base
  attr_accessible :content, :name, :job_id

  belongs_to :job
end
```

association nested comments into job

add the has_many line in job.rb

```
class Job < ActiveRecord::Base
  attr_accessible :content, :title
  attr_accessible :category, :salary,
  attr_accessible :valid_till

  has_many :comments

  def self.min_salary(salary)
    where("salary >= ?", salary)
  end

  def self.still_valid
    where("valid_till >= ?", Date.current)
  end
end
```

association

nested

comments

into job

then in the routes.rb, we nest the resources.

```
resources :jobs do
  resources :comments
end
```

association nested comments into job

the result of rake routes.

job_board_11_comments — bash — 107×24 — ⌘4

```
job_board_11_comments (v-.-v) master$ rake routes
    job_comments GET    /jobs/:job_id/comments(.:format)          comments#index
                 POST   /jobs/:job_id/comments(.:format)          comments#create
 new_job_comment GET    /jobs/:job_id/comments/new(.:format)      comments#new
edit_job_comment GET    /jobs/:job_id/comments/:id/edit(.:format) comments#edit
     job_comment GET    /jobs/:job_id/comments/:id(.:format)      comments#show
                 PUT    /jobs/:job_id/comments/:id(.:format)      comments#update
                 DELETE /jobs/:job_id/comments/:id(.:format)      comments#destroy
            jobs GET    /jobs(.:format)                           jobs#index
                 POST   /jobs(.:format)                           jobs#create
         new_job GET    /jobs/new(.:format)                       jobs#new
        edit_job GET    /jobs/:id/edit(.:format)                  jobs#edit
             job GET    /jobs/:id(.:format)                       jobs#show
                 PUT    /jobs/:id(.:format)                       jobs#update
                 DELETE /jobs/:id(.:format)                       jobs#destroy
            root        /                                         pages#index
            page GET    /pages/*id                                high_voltage/pages#show
job_board_11_comments (v-.-v) master$
```

association nested comments into job

add this line to all methods in **comments_controller.rb**

```
@job = Job.find params[:job_id]
```

association nested comments into job

the index method in comments controller.

```ruby
def index
    @job = Job.find params[:job_id]

    @comments = @job.comments.all

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @comments }
    end
  end
```

association nested comments into job

the show method in comments controller.

```
def show
    @job = Job.find params[:job_id]
    @comment = @job.comments.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @comment }
    end
end
```

association nested comments into job

the new method in comments controller.

```
def new
    @job = Job.find params[:job_id]
    @comment = @job.comments.new

    respond_to do |format|
      format.html # new.html.erb
      format.json { render json: @comment }
    end
end
```

association nested comments into job

the edit method in comments controller.

```
def edit
    @job = Job.find params[:job_id]
    @comment = @job.comments.find(params[:id])
end
```

association nested comments into job

the show method in comments controller.

```
def show
    @job = Job.find params[:job_id]
    @comment = @job.comments.find(params[:id])

    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @comment }
    end
end
```

association

nested

comments

into job

change all 'comment' in
comment views and controller to:

[@job, comment]
job_comments_path(@job)
new_job_comment_path(@job)
job_comment_url(@job)

recources ruby-toolbox.com