Object Oriented WEB Programming Using Ruby

Day 10: 21/June/2012

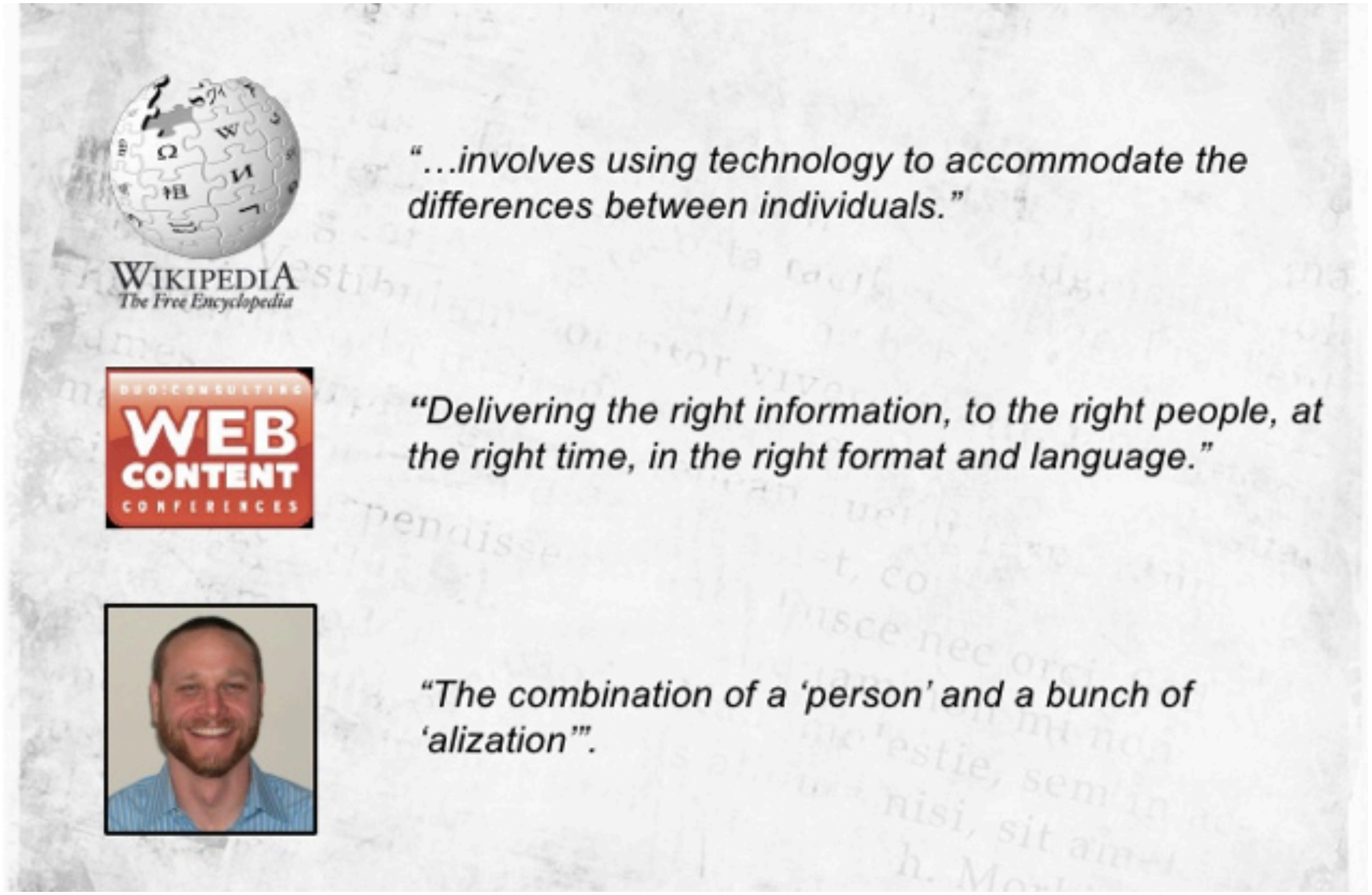
Personalization

Today’s Goal

- This topics was not included in the original syllabus. It was arranged because of the following two reasons;
  - To fit the higher requirement level of the course attendants’ potentials.
  - To complete the installation of the ‘project.’

- When Database application runs on the WEB, the screen view is often arranged for each user. We learn this arrangement as personalization.

What is Personalization?


"…involves using technology to accommodate the differences between individuals."

"Delivering the right information, to the right people, at the right time, in the right format and language."

"The combination of a 'person' and a bunch of 'alization'".

http://www.slideshare.net/forakerdesign/web-content-personalization-three-case-studies

Our Installation

We support the following two points;

1. One guest can vote only once for one topic.
2. Guest can see what he voted, and can change his/her vote watching others votes.

Design policy for ‘votes’ table (1)

- Now we introduce ‘votes’ table.
  - Previously, our table design was assisted by rails’ ‘scaffolding’ support, and the generated skeleton was enough to maintain the data and enough to show the contents.
- The behavior of ‘votes’ is different.
  - One record is added when a guest votes for any topic, and this is the only way to add a record to ‘votes’ table.

Design policy for ‘votes’ table (2)

However, the scaffold structure is helpful enough to maintain the record, for the administration purpose.

So, first we scaffold the votes table, and then, add ‘vote buttons’ to the view, add an action method to ‘vote’ to the model, and add ‘indicator’ to show which side the guest voted.

Votes table

Field Design

- Id, the primary key,
- Guest_id (user_id), to record who voted,
- Vote, to record pros(1) or cons(0),
- Causes_id, to record the topic, or
- Solution_id, to record the topic.

To simplify the design, it would be better to have two 'votes' tables, 'cause_votes' and 'solution_votes.' I will adopt this 'two votes table design.'

Design Only ----

I am very sorry that the coding is on the way, and I have not finished yet…

I put all my design plan on these slides, and explain what to do with the design as description.

Probably, I will show the program code writing cold, without any rehearsal…

Please, you yourself try to complete the design.

For me, 'Time-Out.. (Game Over)' for today. But the game is not over yet. I keep on writing source codes, and show the completed version source code to the public by all means.

Scaffolding of vote

Relations to guests, causes, and solutions tables are needed.

```
rails generate scaffold causeVote guest_id:integer vote:integer
    cause_id:integer
```

```
rails generate scaffold solutionVote guest_id:integer vote:integer
    solution_id:integer
```

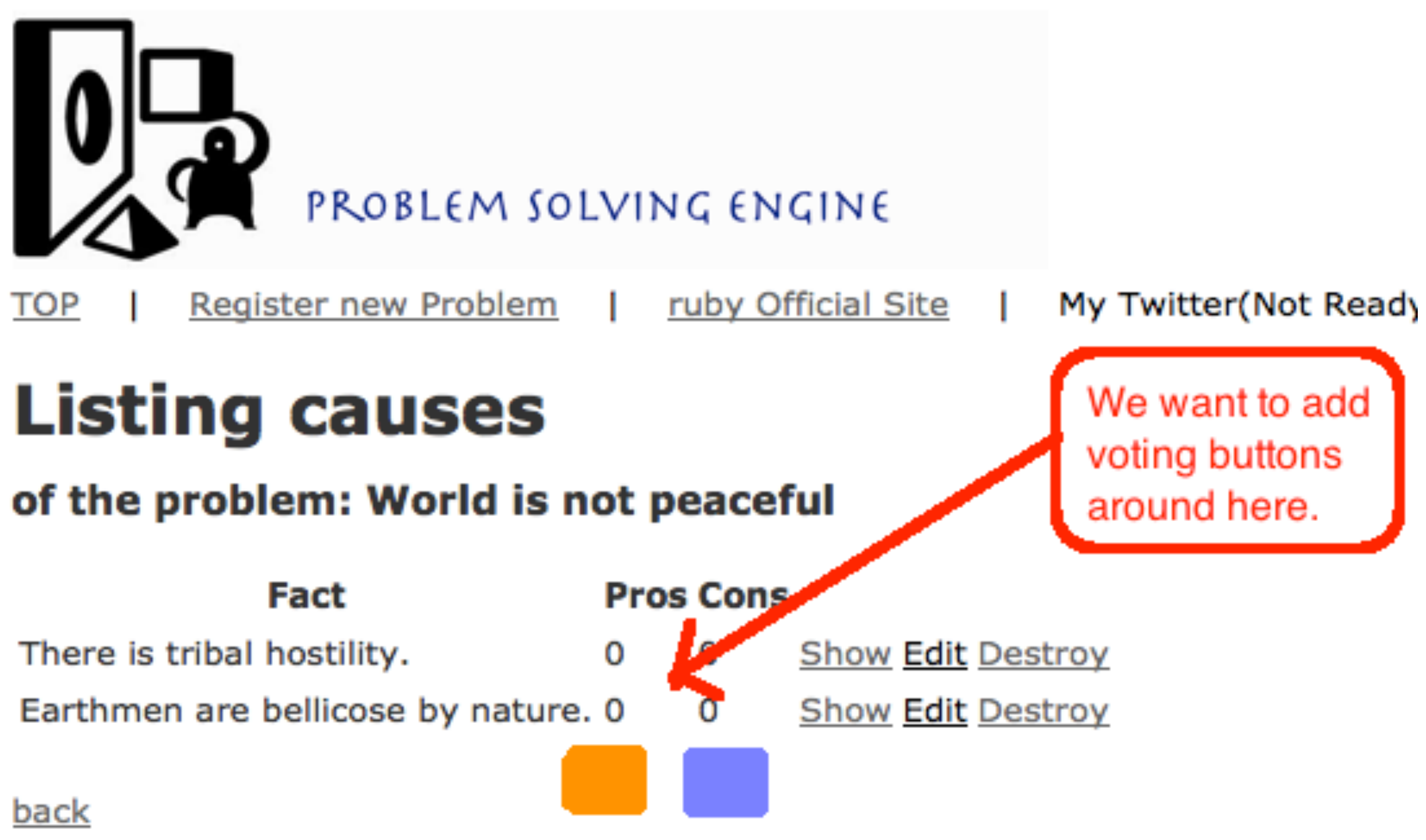
And then, migrate

```
rake db:migrate
```

Vote button design

- Put ‘Pro’ and ‘Con’ button on the list(index) screen of the Causes/Problems views.
- We should write ‘form_tag’ block to both index.html.erb file.
- When either button is pressed, send a method(POST) to the causes_controller.rb or solutions_controller.rb, with a parameter either ‘Pro’ or ‘Con.’

Voting buttons arrangement



PROBLEM SOLVING ENGINE

TOP | Register new Problem | ruby Official Site | My Twitter(Not Ready

Listing causes

of the problem: World is not peaceful

Fact	Pros	Cons	
There is tribal hostility.	0	0	Show Edit Destroy
Earthmen are bellicose by nature.	0	0	Show Edit Destroy

back

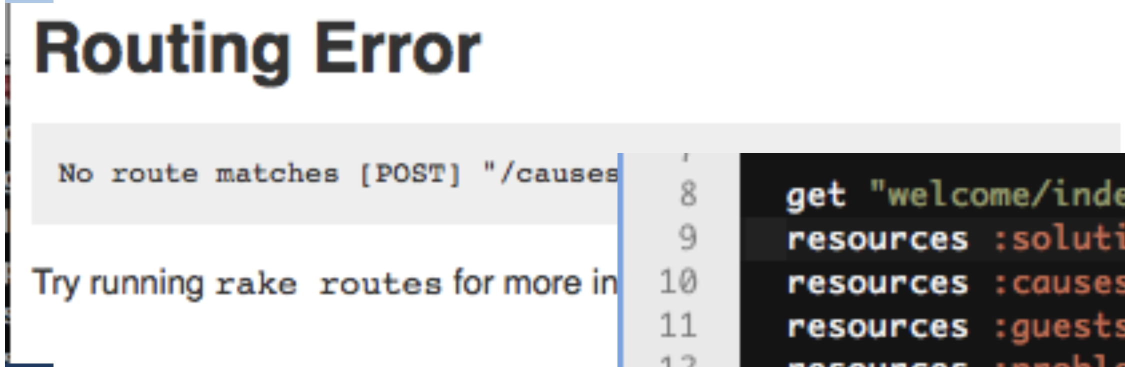
We want to add voting buttons around here.

config/routes.rb

Add one line below to config/routes.rb.

match 'causes/vote(/:cause_id)' => 'causes#vote'

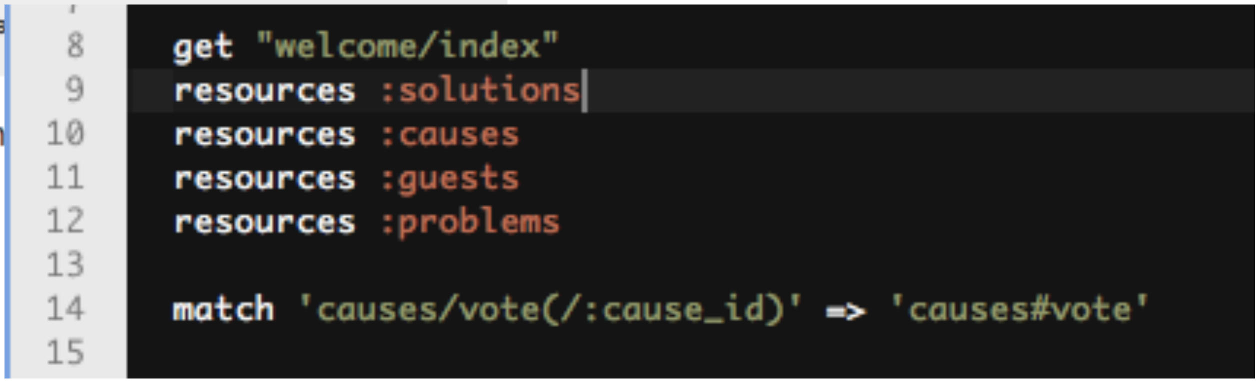
To avoid the following Routing Error, and then add 'vote' method to the causes_controller.rb.



Routing Error

```
No route matches [POST] "/causes
```

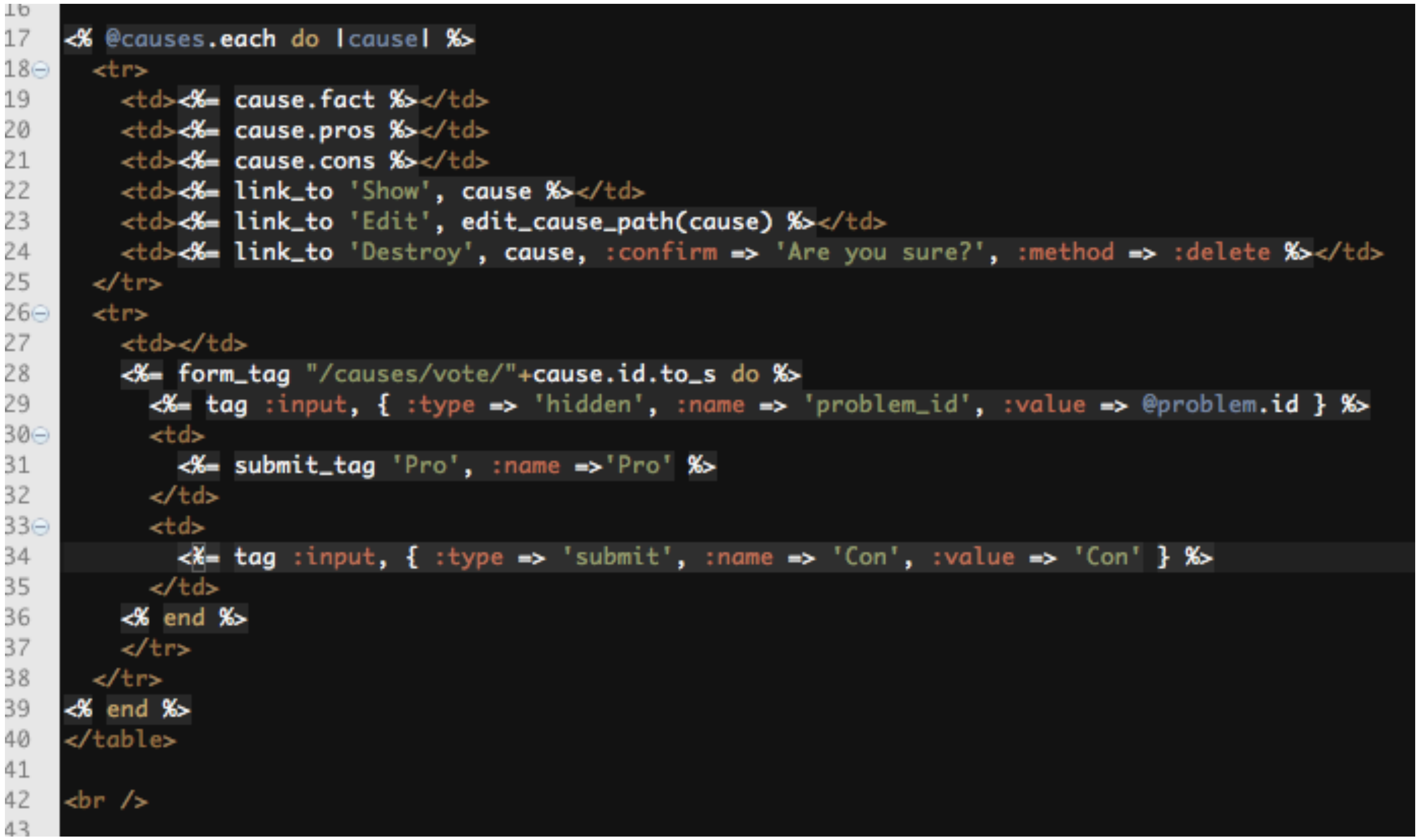
Try running `rake routes` for more in



```
get "welcome/index"
resources :solutions
resources :causes
resources :guests
resources :problems

match 'causes/vote(/:cause_id)' => 'causes#vote'
```

views/causes/index.html.erb



```erb
<% @causes.each do |cause| %>
  <tr>
    <td><%= cause.fact %></td>
    <td><%= cause.pros %></td>
    <td><%= cause.cons %></td>
    <td><%= link_to 'Show', cause %></td>
    <td><%= link_to 'Edit', edit_cause_path(cause) %></td>
    <td><%= link_to 'Destroy', cause, :confirm => 'Are you sure?', :method => :delete %></td>
  </tr>
  <tr>
    <td></td>
    <%= form_tag "/causes/vote/"+cause.id.to_s do %>
      <%= tag :input, { :type => 'hidden', :name => 'problem_id', :value => @problem.id } %>
      <td>
        <%= submit_tag 'Pro', :name =>'Pro' %>
      </td>
      <td>
        <%= tag :input, { :type => 'submit', :name => 'Con', :value => 'Con' } %>
      </td>
    <% end %>
    </tr>
  </tr>
<% end %>
</table>

<br />
```

Index.html.erb (text)

```
<% @causes.each do |cause| %>
  <tr>
    <td><%= cause.fact %></td>
    <td><%= cause.pros %></td>
    <td><%= cause.cons %></td>
    <td><%= link_to 'Show', cause %></td>
    <td><%= link_to 'Edit', edit_cause_path(cause) %></td>
    <td><%= link_to 'Destroy', cause, :confirm => 'Are you sure?', :method => :delete %></td>
  </tr>
  <tr>
      <td></td>
    <%= form_tag "/causes/vote/"+cause.id.to_s do %>
      <%= tag :input, { :type => 'hidden', :name => 'problem_id', :value => @problem.id } %>
        <td>
        <%= submit_tag 'Pro', :name =>'Pro' %>
      </td>
      <td>
        <%= tag :input, { :type => 'submit', :name => 'Con', :value => 'Con' } %>
      </td>
    <% end %>
    </tr>
  </tr>
<% end %>
```

Single Quotation

Please note, that if you copy text from my slides, sometimes, the PowerPoint converts single quotation to special ‘’ letters, and if you leave them, it may cause errors.

Submit Button

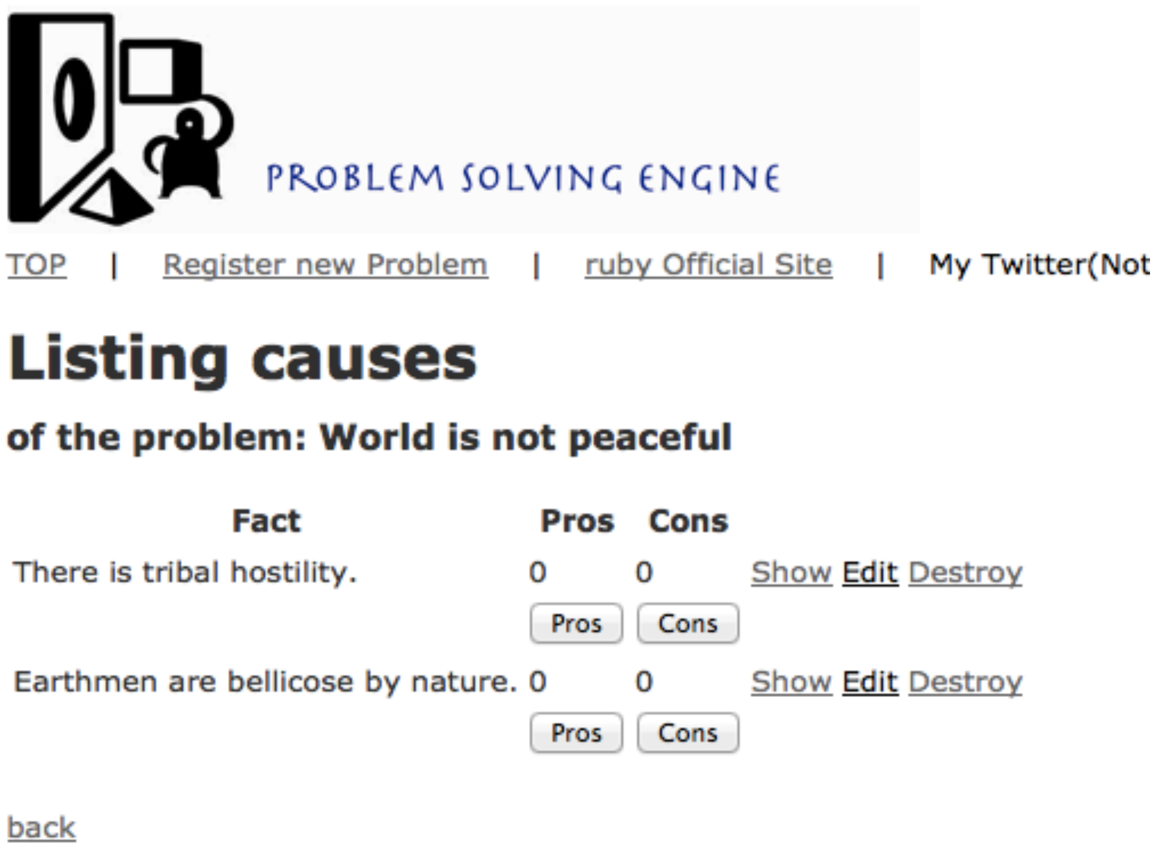
I have added two ways to install button.

You can try either way, (because, it is very annoying if there are two different coding styles mixed.)

Also, the html is handed a parameter ‘Problem’ Class instance, and we may need to hand those parameters back to the controllers.

(The other way is to set it to ‘global’ variable, but it may cause some trouble when it runs on multi-thread environment.)

views/causes/index.html.erb



TOP | Register new Problem | ruby Official Site | My Twitter(Not

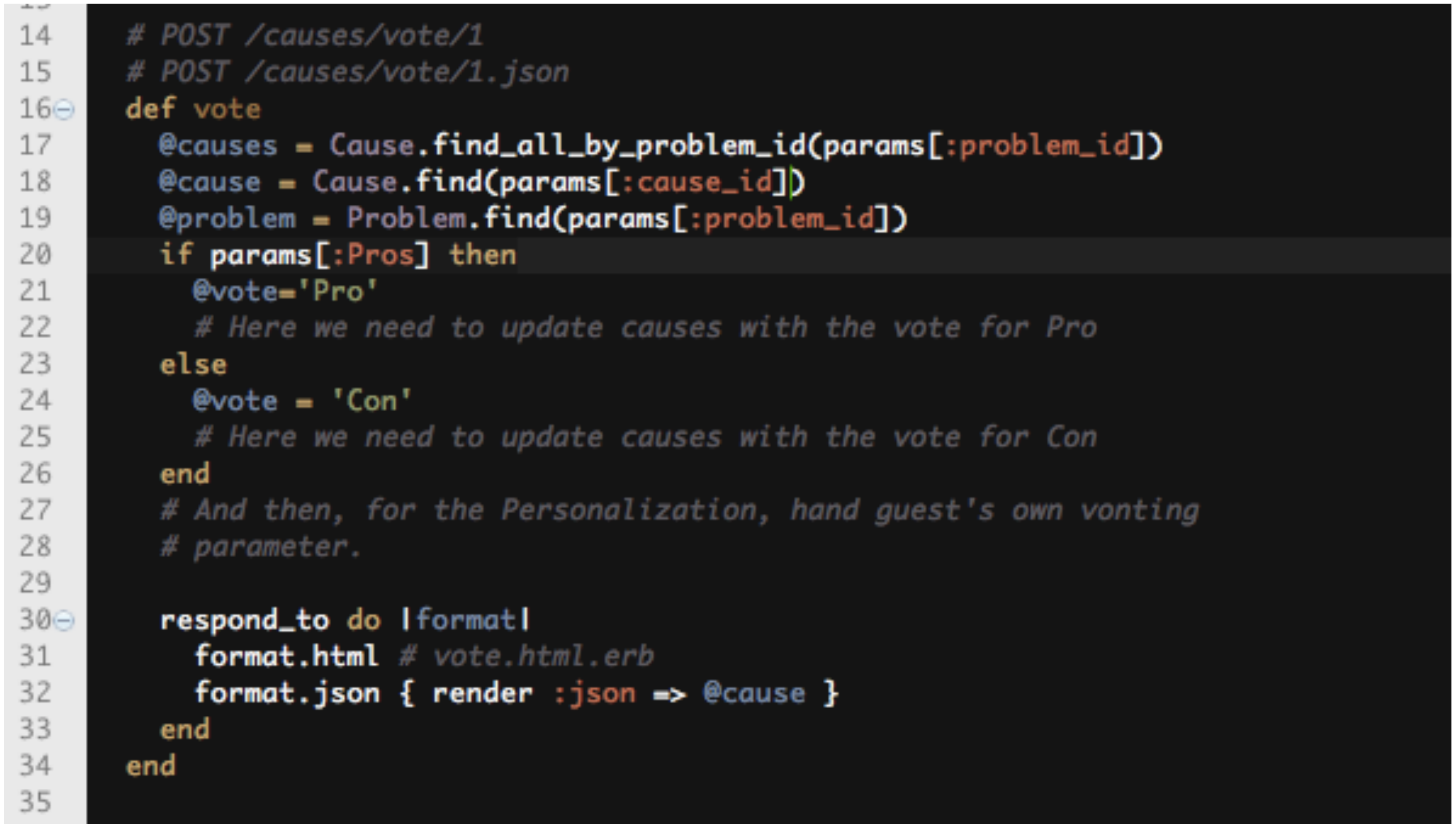
Listing causes

of the problem: World is not peaceful

Fact	Pros	Cons	
There is tribal hostility.	0	0	Show Edit Destroy
	Pros	Cons	
Earthmen are bellicose by nature.	0	0	Show Edit Destroy
	Pros	Cons	

back

Causes_controller.rb

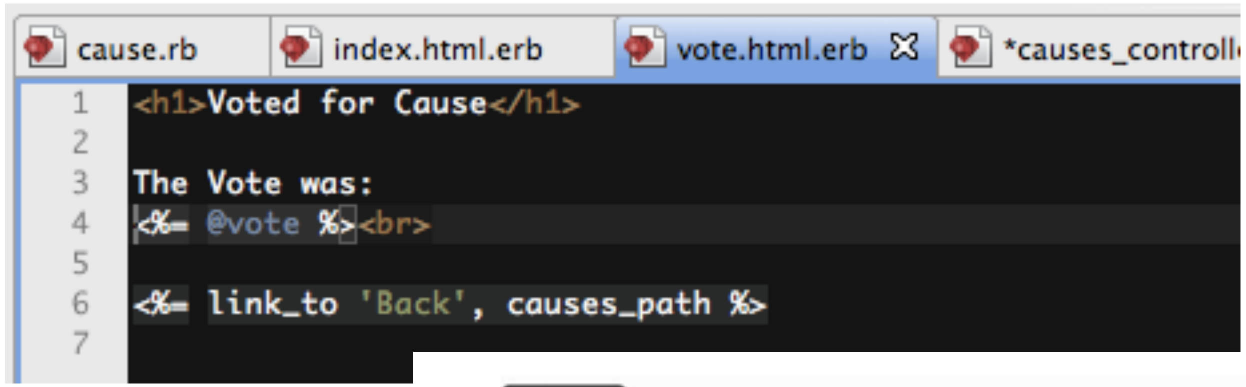


```ruby
  # POST /causes/vote/1
  # POST /causes/vote/1.json
  def vote
    @causes = Cause.find_all_by_problem_id(params[:problem_id])
    @cause = Cause.find(params[:cause_id])
    @problem = Problem.find(params[:problem_id])
    if params[:Pros] then
      @vote='Pro'
      # Here we need to update causes with the vote for Pro
    else
      @vote = 'Con'
      # Here we need to update causes with the vote for Con
    end
    # And then, for the Personalization, hand guest's own vonting
    # parameter.

    respond_to do |format|
      format.html # vote.html.erb
      format.json { render :json => @cause }
    end
  end

```

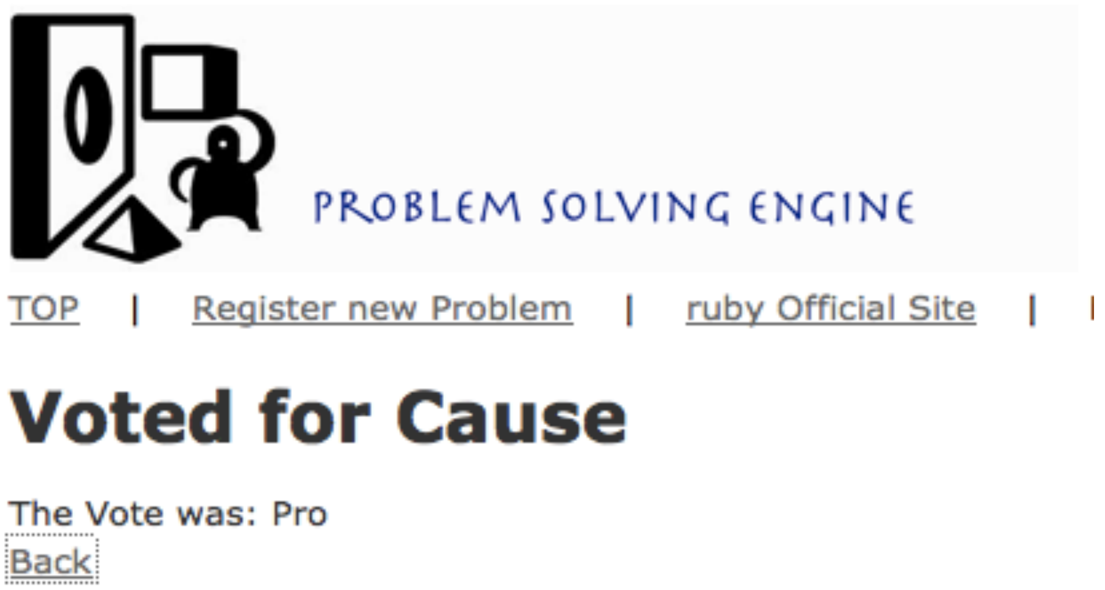
Views/causes/vote.html.erb



```
<h1>Voted for Cause</h1>

The Vote was:
<%= @vote %><br>

<%= link_to 'Back', causes_path %>
```

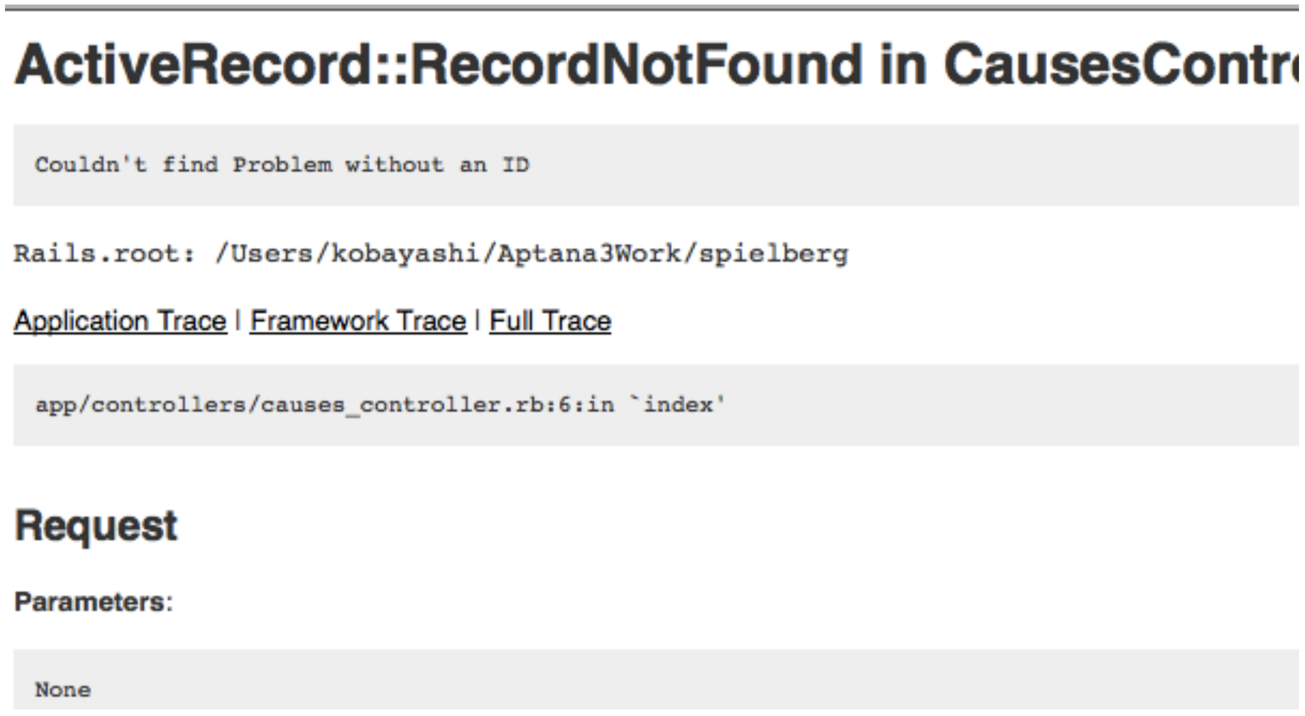


Voted for Cause

The Vote was: Pro
Back

Parameter failed

We did not hand the Problem_id to causes/index, so when we click 'BACK' button, the following error occurred.



ActiveRecord::RecordNotFound in CausesContr

```
Couldn't find Problem without an ID
```

```
Rails.root: /Users/kobayashi/Aptana3Work/spielberg
```

Application Trace | Framework Trace | Full Trace

```
app/controllers/causes_controller.rb:6:in `index'
```

Request

Parameters:

```
None
```

Two ways to clear the error.

Voting result screen has been handed three variables, @causes(hash), @cause(Cause class instance), and @problem(Problem class instance).

We should rewrite the following line

```
<%= link_to 'Back', causes_path %>
```

to hand @problem.id, to the causes/index path.

1) Use link_to, and rewrite URL,

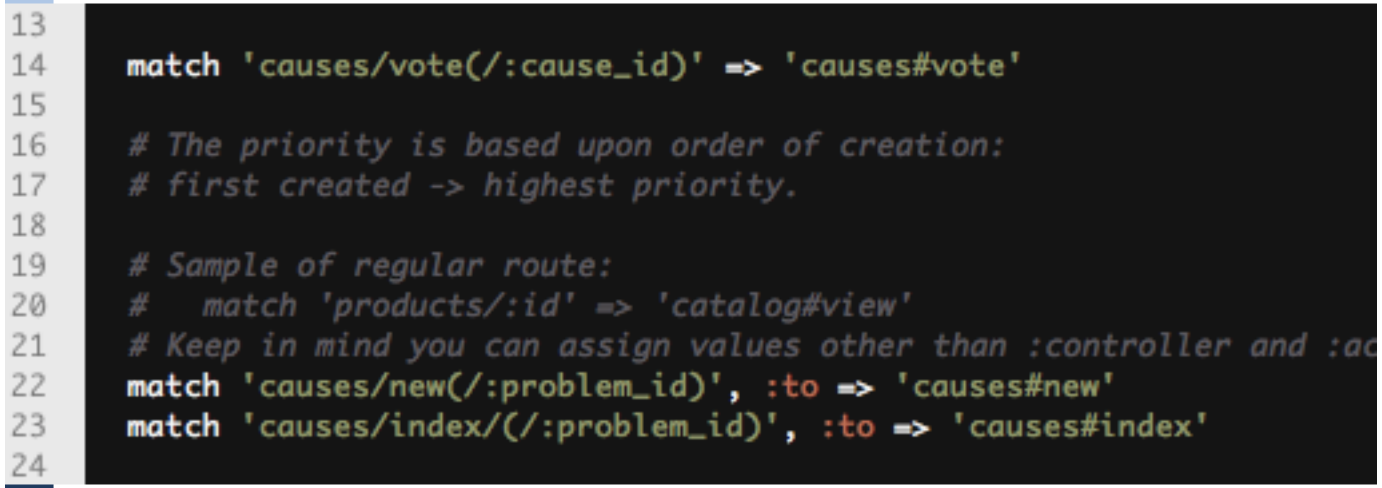
```
<%= link_to 'Back', '/causes/index/'+@problem.id.to_s %>
```

2) Use Button to hand parameter as a hidden input.

When you change links

Just remember, you modified routes.rb.

Also, check routes' names by typing 'rake routes' command.



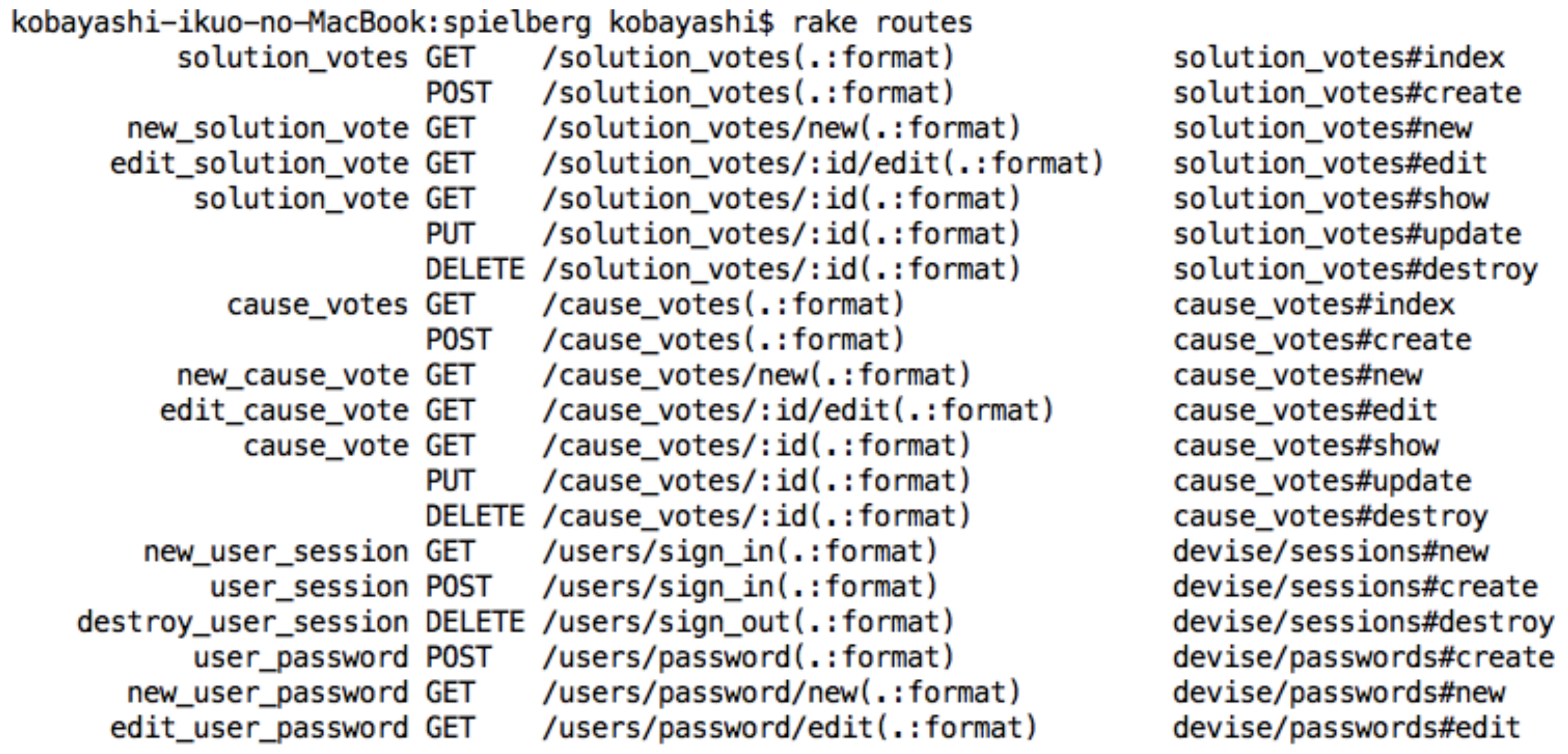
```
match 'causes/vote(/:cause_id)' => 'causes#vote'

# The priority is based upon order of creation:
# first created -> highest priority.

# Sample of regular route:
#   match 'products/:id' => 'catalog#view'
# Keep in mind you can assign values other than :controller and :ac
match 'causes/new(/:problem_id)', :to => 'causes#new'
match 'causes/index/(/:problem_id)', :to => 'causes#index'
```

Rake routes

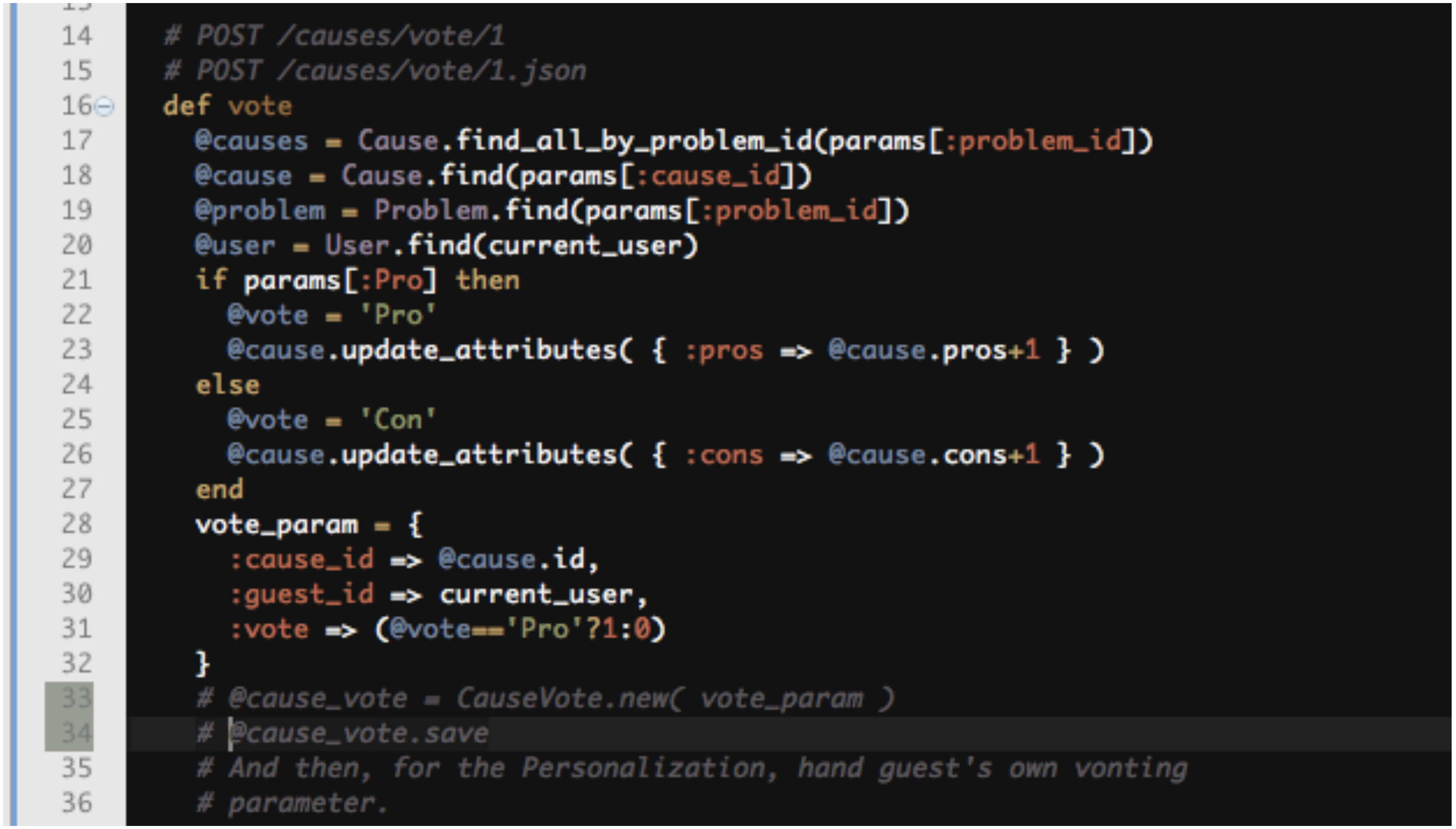
Later on, we may need to remove routes which we will not use, or which we should not allow users to access.



```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rake routes
         solution_votes GET    /solution_votes(.:format)                solution_votes#index
                        POST   /solution_votes(.:format)                solution_votes#create
      new_solution_vote GET    /solution_votes/new(.:format)            solution_votes#new
     edit_solution_vote GET    /solution_votes/:id/edit(.:format)       solution_votes#edit
          solution_vote GET    /solution_votes/:id(.:format)            solution_votes#show
                        PUT    /solution_votes/:id(.:format)            solution_votes#update
                        DELETE /solution_votes/:id(.:format)            solution_votes#destroy
            cause_votes GET    /cause_votes(.:format)                   cause_votes#index
                        POST   /cause_votes(.:format)                   cause_votes#create
         new_cause_vote GET    /cause_votes/new(.:format)               cause_votes#new
        edit_cause_vote GET    /cause_votes/:id/edit(.:format)          cause_votes#edit
             cause_vote GET    /cause_votes/:id(.:format)               cause_votes#show
                        PUT    /cause_votes/:id(.:format)               cause_votes#update
                        DELETE /cause_votes/:id(.:format)               cause_votes#destroy
       new_user_session GET    /users/sign_in(.:format)                 devise/sessions#new
           user_session POST   /users/sign_in(.:format)                 devise/sessions#create
   destroy_user_session DELETE /users/sign_out(.:format)                devise/sessions#destroy
          user_password POST   /users/password(.:format)                devise/passwords#create
      new_user_password GET    /users/password/new(.:format)            devise/passwords#new
     edit_user_password GET    /users/password/edit(.:format)           devise/passwords#edit
```

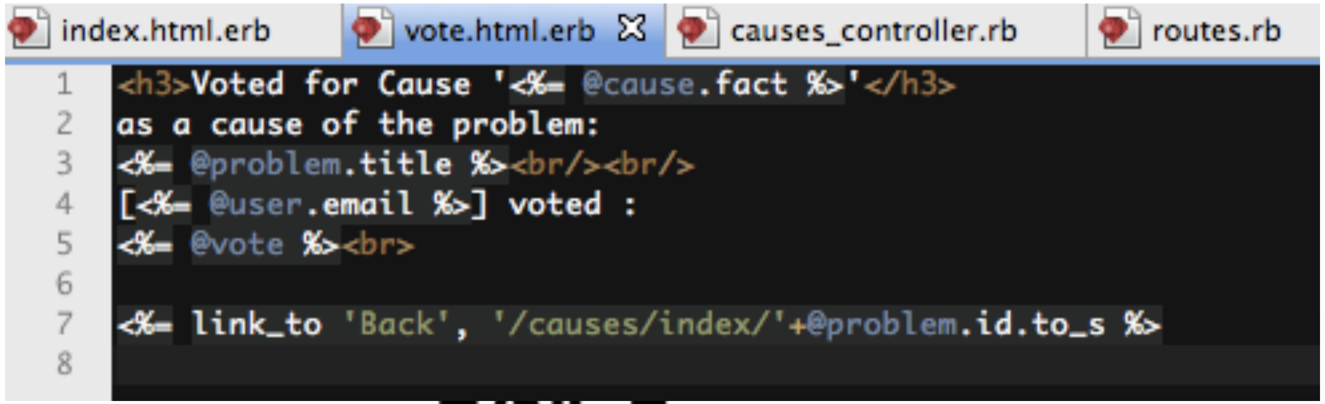
To count up votes (vote method)

app/controllers/causes_controller.rb



```ruby
# POST /causes/vote/1
# POST /causes/vote/1.json
def vote
  @causes = Cause.find_all_by_problem_id(params[:problem_id])
  @cause = Cause.find(params[:cause_id])
  @problem = Problem.find(params[:problem_id])
  @user = User.find(current_user)
  if params[:Pro] then
    @vote = 'Pro'
    @cause.update_attributes( { :pros => @cause.pros+1 } )
  else
    @vote = 'Con'
    @cause.update_attributes( { :cons => @cause.cons+1 } )
  end
  vote_param = {
    :cause_id => @cause.id,
    :guest_id => current_user,
    :vote => (@vote=='Pro'?1:0)
  }
  # @cause_vote = CauseVote.new( vote_param )
  # @cause_vote.save
  # And then, for the Personalization, hand guest's own vonting
  # parameter.
```

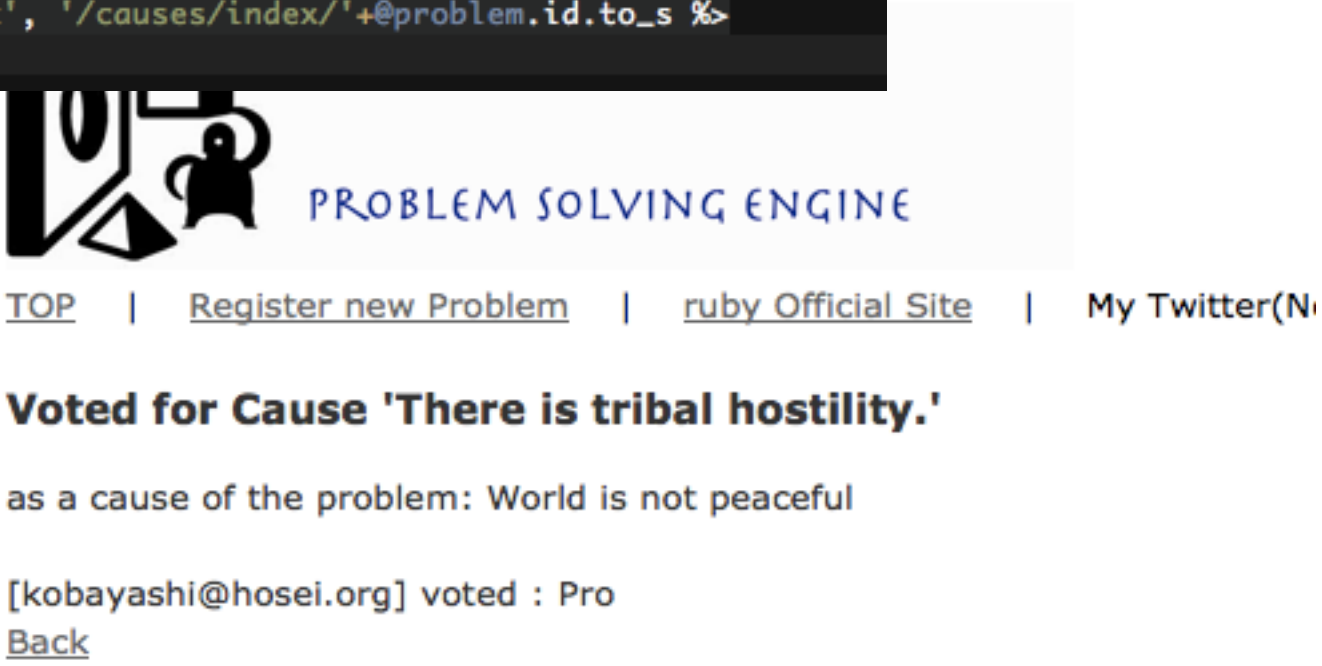
views/causes/vote.html.erb



```
<h3>Voted for Cause '<%= @cause.fact %>'</h3>
as a cause of the problem:
<%= @problem.title %><br/><br/>
[<%= @user.email %>] voted :
<%= @vote %><br>

<%= link_to 'Back', '/causes/index/'+@problem.id.to_s %>

```



PROBLEM SOLVING ENGINE
TOP | Register new Problem | ruby Official Site | My Twitter(N

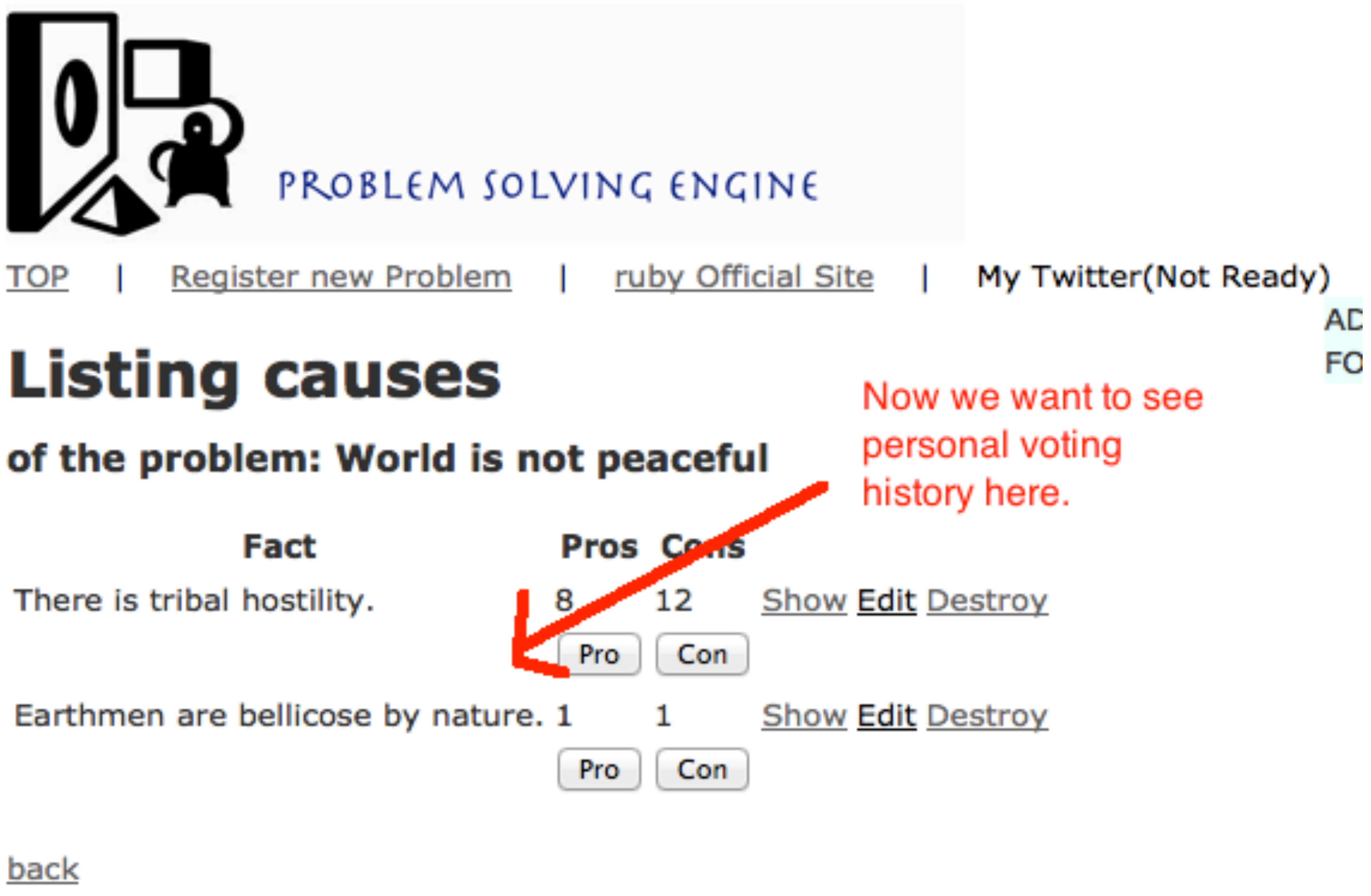
Voted for Cause 'There is tribal hostility.'

as a cause of the problem: World is not peaceful

[kobayashi@hosei.org] voted : Pro
Back

views/causes/index.html.erb

After recording the votes;



TOP | Register new Problem | ruby Official Site | My Twitter(Not Ready)

AD
FO

Listing causes

of the problem: World is not peaceful

Now we want to see personal voting history here.

Fact	Pros	Cons	
There is tribal hostility.	8	12	Show Edit Destroy
	Pro	Con	
Earthmen are bellicose by nature.	1	1	Show Edit Destroy
	Pro	Con	

back

Design of Relations

To maintain the ‘causes_id’ and/or ‘solutions_id’ of ‘votes’ table, we need the description of relations.

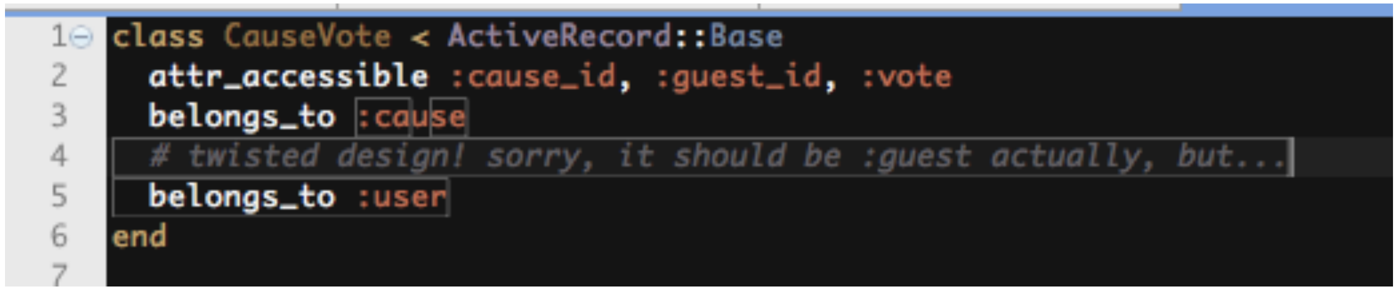
One Vote belongs to a Cause.

A Cause has many Votes.

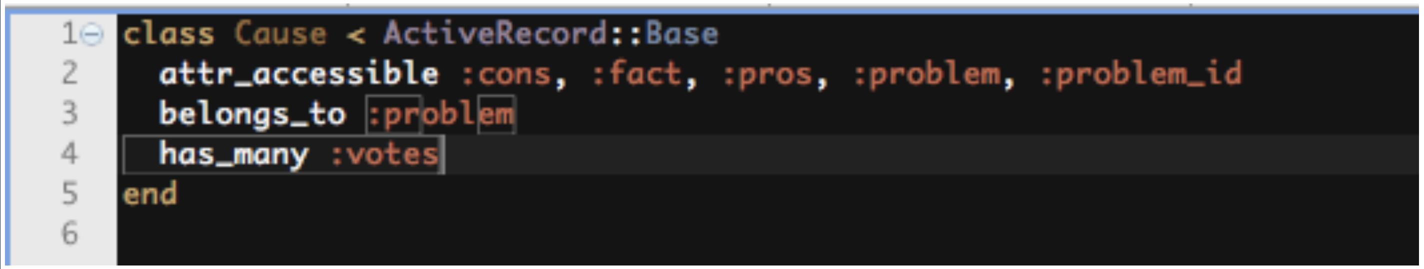
So the relationship between Vote-Cause is one to many.

Relationship

models/cause.rb, models/cause_vote.rb, and models/users.rb..



```ruby
class CauseVote < ActiveRecord::Base
  attr_accessible :cause_id, :guest_id, :vote
  belongs_to :cause
  # twisted design! sorry, it should be :guest actually, but...
  belongs_to :user
end
```

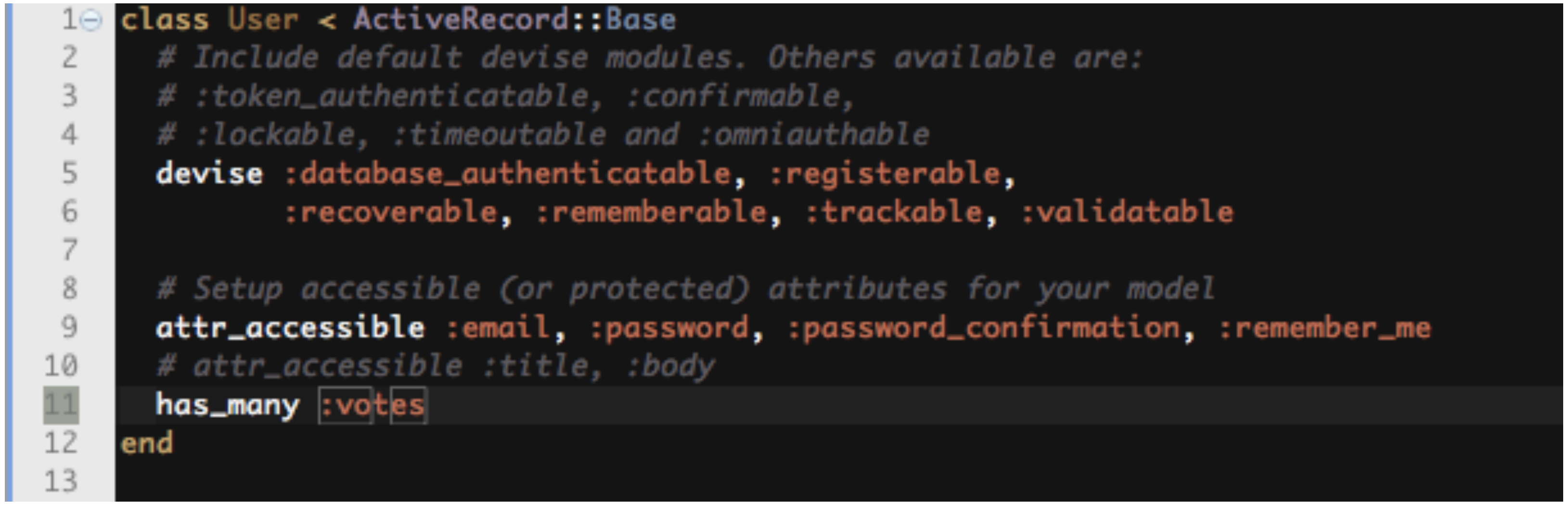


```ruby
class Cause < ActiveRecord::Base
  attr_accessible :cons, :fact, :pros, :problem, :problem_id
  belongs_to :problem
  has_many :votes
end
```

Twisted Design, sorry

Actually, we should install ‘guest’ to control the votes, but I had used ‘user’ instead, because of the time limitations…

You can see my program as a sample of ‘bad manner.’

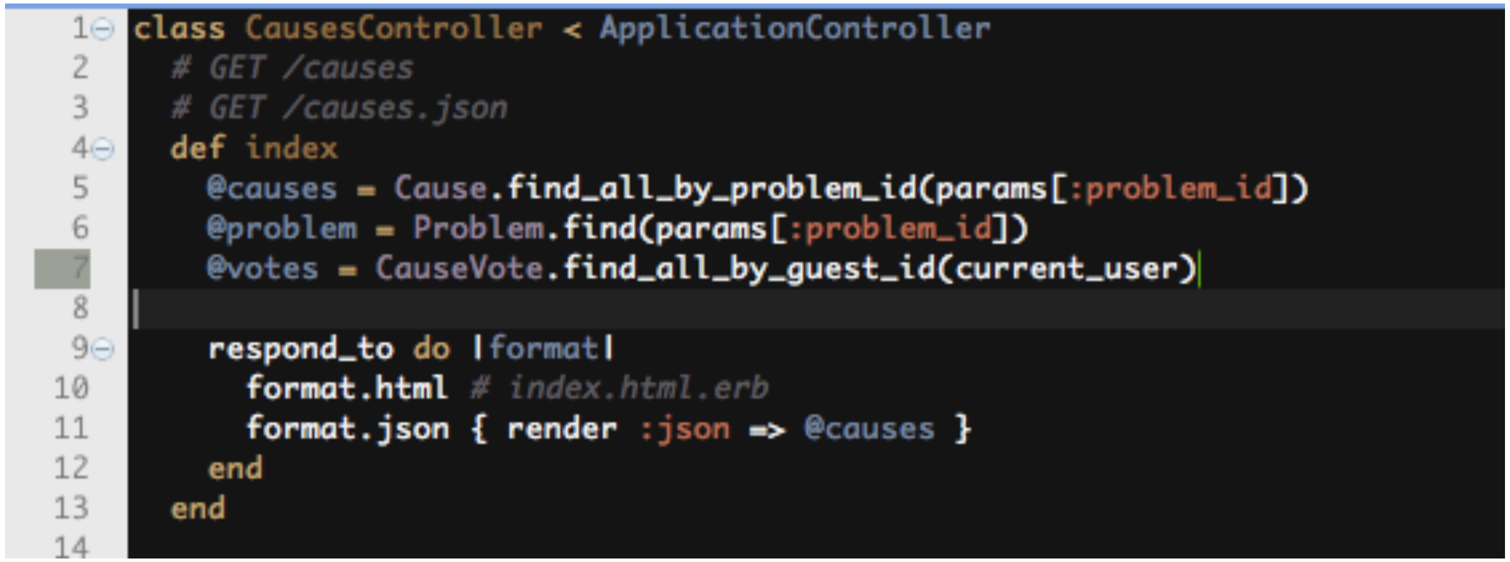


```ruby
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :token_authenticatable, :confirmable,
  # :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :validatable

  # Setup accessible (or protected) attributes for your model
  attr_accessible :email, :password, :password_confirmation, :remember_me
  # attr_accessible :title, :body
  has_many :votes
end
```

Fetch history

When we first call index method of the causes controller, we fetch all the voting records from cause_votes table.



```ruby
class CausesController < ApplicationController
  # GET /causes
  # GET /causes.json
  def index
    @causes = Cause.find_all_by_problem_id(params[:problem_id])
    @problem = Problem.find(params[:problem_id])
    @votes = CauseVote.find_all_by_guest_id(current_user)

    respond_to do |format|
      format.html # index.html.erb
      format.json { render :json => @causes }
    end
  end

```

What we should do next…

Look up @votes hash array, and if we find a vote record for the current user to the selected ‘cause,’ then we show the voting history, in index.html.erb.

If there is a voting record, then, we should update the record after his ‘re-vote,’ else, we should create the vote record.

To create the vote record..

```
vote_param = {
      :cause_id => @cause.id,
      :guest_id => current_user,   # it should be guest_id actually
      :vote => (@vote=='Pro'?1:0)
   }
# if it is a new vote then
  @cause_vote = CauseVote.new( vote_param )
  @cause_vote.save
# else
   @cause_vote.update_attributes( vote_param )
# end
```

Personalization?

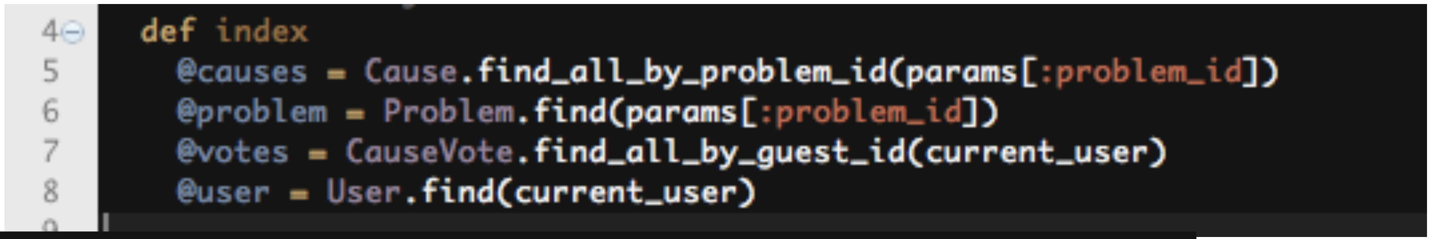
I wanted to add ‘voting history’ on the screen… but once again, time up….

I have added ‘user name’ on the screen instead…

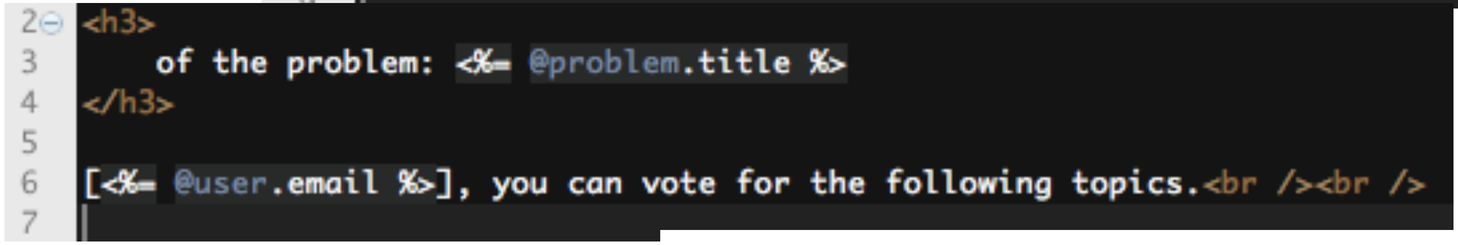
Today’s lecture was the demonstration to advertise wine and sell vinegar.

I will complete the system within a ~~month~~ year.

Causes index screen

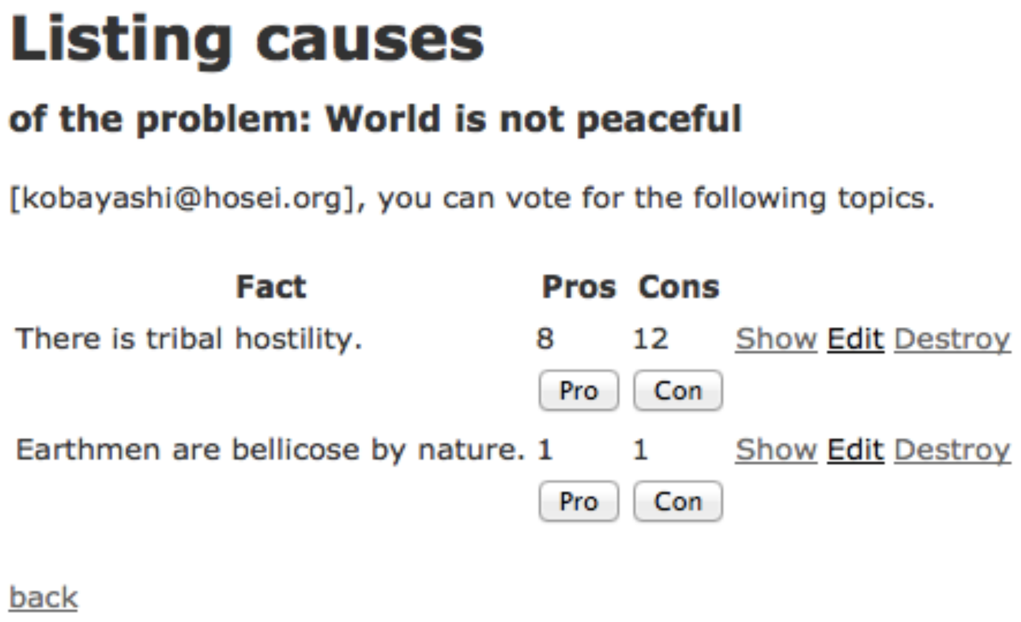


```ruby
def index
  @causes = Cause.find_all_by_problem_id(params[:problem_id])
  @problem = Problem.find(params[:problem_id])
  @votes = CauseVote.find_all_by_guest_id(current_user)
  @user = User.find(current_user)
```



```erb
<h3>
    of the problem: <%= @problem.title %>
</h3>

[<%= @user.email %>], you can vote for the following topics.<br /><br />
```



Listing causes

of the problem: World is not peaceful

[kobayashi@hosei.org], you can vote for the following topics.

Fact	Pros	Cons	
There is tribal hostility.	8	12	Show Edit Destroy
	Pro	Con	
Earthmen are bellicose by nature.	1	1	Show Edit Destroy
	Pro	Con	

back

Prepare for the Next Week

The lecture plan for the next week is ‘Upload and download images.’

I myself would try to show my own program source code by next week of the personalization result, but I cannot say I would surely do. So please you yourself try to install the design plan which I have shown to you.

Syllabus? I will check it by next week. Once again, sorry for my negligence.