



OBJECT ORIENTED WEB PROGRAMMING USING RUBY

Day 12: 5/June/2012

Session Management

Today's Goal

- ❑ Session Management is closely related with Personalization.
- ❑ Also, we have some leftovers from last week.
- ❑ We finish some unfinished works, and then, investigate the session management structure with current project.

Image Display

Last week, we did not reached to show the last step of image display.

Let us finish the 'image-show' for causes screen.

Where was the problem?

Cause of this problem (of the program)

It seems rails try to find the image file with 'img_tag' description. That means, the 'causes' controller's 'file' method was not reached from the routing process.

```
Started GET "/assets?action=file&controller=causes&filename=gazou4.png&id=4" for 127.0.0.1
un 28 09:58:25 +0900 2012
Served asset - 404 Not Found (5ms)
```

```
ActionController::RoutingError (No route matches [GET] "/assets"):
```

- actionpack (3.2.3) lib/action_dispatch/middleware/debug_exceptions.rb:21:in `call'
- actionpack (3.2.3) lib/action_dispatch/middleware/show_exceptions.rb:56:in `call'
- railties (3.2.3) lib/rails/rack/logger.rb:26:in `call_app'
- railties (3.2.3) lib/rails/rack/logger.rb:16:in `call'
- actionpack (3.2.3) lib/action_dispatch/middleware/request_id.rb:22:in `call'
- rack (1.4.1) lib/rack/methodoverride.rb:21:in `call'
- rack (1.4.1) lib/rack/runtime.rb:17:in `call'
- activesupport (3.2.3) lib/active_support/cache/strategy/local_cache.rb:72:in `call'
- rack (1.4.1) lib/rack/lock.rb:15:in `call'

Modification for Image Display

Add

`get 'causes/file' => 'causes#file'`

in config/routes.rb file, before;

`resources :causes`

This is all!

```
1 Spielberg::Application.routes.draw do
2   resources :solution_votes
3
4   resources :cause_votes
5
6   devise_for :users
7
8   get "welcome/index"
9   resources :solutions
10  get 'causes/file' => 'causes#file'
11  resources :causes
12  resources :guests
13  resources :problems
14
```



PROBLEM SOLVING ENGINE


[TOP](#) | [Register new Problem](#) | [ruby Official Site](#) | [My Twitter\(Not Ready\)](#) | [AD Spa](#) | [FOR RE](#)

As a Cause of the problem: I cannot remove this bug.

Fact: Add Image

Pros: 0 Cons: 0

Attachments

Name	Size	Content type	Content	Created at
gazou4.png	66982	image/png		2012-07-03 16:22:55 UTC

[Edit](#) | [Back](#)

Your Exercise

Would you add [Remove] button or a link to delete the attached file to the cause?

Hint: invoke the 'destroy' method to the 'cause_attachment' instance.

To invoke the 'destroy' method,

- 1) write delete_attachment method in the causes controller, and
- 2) hand attachment instance id to the controller.

Voting Control

Now we finish modifying 'Voting History Control.' A guest may vote 'Pro' for certain 'cause' of the 'problem,' if he/she may think that the cause could be the beginning essence of the problem, but if he/she might hit upon an idea that the real reason for the problem lies in the other places, he/she might change his/her vote for 'Con.'

views/causes/vote.html.erb

```
index.html.erb | vote.html.erb | causes_controller.rb | routes.rb
1 <h3>Voted for Cause '<%= @cause.fact %>'</h3>
2 as a cause of the problem:
3 <%= @problem.title %><br/><br/>
4 [<%= @user.email %>] voted :
5 <%= @vote %><br/>
6
7 <%= link_to 'Back', '/causes/index/'+@problem.id.to_s %>
8
```

Review from the
Day 10 Slide



PROBLEM SOLVING ENGINE

[TOP](#) | [Register new Problem](#) | [ruby Official Site](#) | [My Twitter\(Ni](#)

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;

Review from the
Day 10 Slide



PROBLEM SOLVING ENGINE

[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 Plan of the Screen

Now we add [Vote Clear] button, and then show only [Con] button when he/she voted for [Pro], and show only [Pro] button when he/she voted for [Con].

Fetch history

Review from the
Day 10 Slide

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

```
1 class CausesController < ApplicationController
2   # GET /causes
3   # GET /causes.json
4   def index
5     @causes = Cause.find_all_by_problem_id(params[:problem_id])
6     @problem = Problem.find(params[:problem_id])
7     @votes = CauseVote.find_all_by_guest_id(current_user)
8
9     respond_to do |format|
10      format.html # index.html.erb
11      format.json { render :json => @causes }
12    end
13  end
14 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.

Review from the
Day 10 Slide

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
```

Here was a bug.
current_user is an
instance of User,
not an integer
value.

Review from the
Day 10 Slide

Clean the database table

We did not have 'one vote per person' control, so while debugging, my tables went wrong. I cleaned the table directly by issuing SQL command on sqlite3.

I am sorry for this very bad manner.

```
sqlite> select * from cause_votes;
1|1|1|2|2012-06-21 06:33:47.564975|2012-06-21 06:33:47.564975
2|1|0|2|2012-06-21 06:40:58.553790|2012-06-21 06:40:58.553790
3|1|0|2|2012-06-21 06:41:03.411161|2012-06-21 06:41:03.411161
4|1|0|2|2012-06-21 06:41:51.723691|2012-06-21 06:41:51.723691
5|1|0|2|2012-06-21 06:41:56.199577|2012-06-21 06:41:56.199577
6|1|0|2|2012-06-21 06:43:06.485843|2012-06-21 06:43:06.485843
7|1|0|2|2012-06-21 06:43:10.197223|2012-06-21 06:43:10.197223
8|1|0|2|2012-06-21 06:43:40.778944|2012-06-21 06:43:40.778944
9|1|0|2|2012-06-21 06:43:45.392621|2012-06-21 06:43:45.392621
10|1|1|2|2012-06-21 06:45:10.117913|2012-06-21 06:45:10.117913
11|1|0|2|2012-06-21 06:45:13.493741|2012-06-21 06:45:13.493741
12|1|1|2|2012-06-21 06:45:57.385495|2012-06-21 06:45:57.385495
13|1|0|2|2012-06-21 06:46:00.831348|2012-06-21 06:46:00.831348
sqlite> delete from cause_votes where cause_id=2;
sqlite> select * from cause_votes;
sqlite>
```

Learn from my mistake

First, I thought `current_user` was an integer value. So, I programmed as;

```
@guest.id = current_user
```

It was wrong. `Current_user` was an instance of User Class. So, I should have been written as;

```
@guest.id = current_user.id
```

But, it seems both worked fine! It was incredible Ruby's acceptance! But, I should have been much more careful.

Detailed Explanation

should be done on the source code.

Please read the source code in the following pages.

I omit writing the explanation on these slides.

Please listen to me in the class room.

I will upload some of the source files into the lecture support system.

views/causes/show.html.erb (1/2)

```
<p id="notice"><%= notice %></p>
```

```
<h3>As a Cause of the problem: <%= @cause.problem.title %>
```

```
<p>
  <b>Fact:</b>
  <%= @cause.fact %>
</p>
```

```
<p>
  <b>Pros:</b>
  <%= @cause.pros %>
  <b>Cons:</b>
  <%= @cause.cons %>
</p>
```

```
<p>
  <%= if @cause.cause_attachments.length>0 %>
  <b>Attachments</b>
  <table border="1">
    <tr>
      <%= for column in @cause.cause_attachments.content_columns %>
        <th><%= column.human_name %></th>
      <% end %>
    </tr>
```

views/causes/show.html.erb (2/2)

```
<% for attachment in @cause.cause_attachments %>
  <tr>
    <% for column in @cause.cause_attachments.content_columns %>
      <% if column.name == 'content' &&
        attachment.content_type =~ /^image\Z/.*(png|jpeg|gif)$/ %>
        <td><%= image_tag url_for({:action => 'file', :id=> attachment.id,
          :filename => attachment.name}), :alt => attachment.name %></td>
      <% else %>
        <td><%= format_column_value attachment, column.name %></td>
      <% end %>
    <% end %>
  </tr>
<% end %>
</table>
<% end %>
</p>

<%= link_to 'Edit', edit_cause_path(@cause) %> |
<%= link_to ('Back', { :controller=>"causes", :action=>"index",
  :problem_id => @cause.problem.id} ) %>
```

Views/causes/vote.html.erb

<h3>Voted for Cause '<%= @cause.fact %>'
</h3>

as a cause of the problem:

<%= @problem.title %>

[<%= current_user.email %>] voted :

<%= @vote %>

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

Views/causes/index.html.erb (1/3)

```
<h1>Listing causes</h1>
<h3>
  of the problem: <%= @problem.title %>
</h3>

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

<table>
  <tr>
    <th>Fact</th>
    <th>Pros</th>
    <th>Cons</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>

  <% @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>
```

Views/causes/index.html.erb (2/3)

```
<% if cause.cause_attachments.length>0 %>
<tr>
  <% for attachment in cause.cause_attachments %>
    <% for column in cause.cause_attachments.content_columns %>
      <% if column.name == 'content' &&
        attachment.content_type =~ /^image\./.*?(png|jpeg|gif)$/ %>
        <td><%= image_tag url_for({:action => 'file', :id=> attachment.id,
          :filename => attachment.name}), :alt => attachment.name %></td>
      <% end %>
    <% end %>
  <% end %>
</tr>
<% end%>
<tr>
  <%= form_tag "/causes/vote/"+cause.id.to_s do %>
  <%= tag :input, { :type => 'hidden', :name => 'problem_id', :value => @problem.id } %>
  <td>
    <% myvote=@votes.select{|vote| vote[:cause_id]==cause.id } %>
    <font color="blue">
    <% if myvote.length == 0 then %>
    Not voted yet.
    <% vote_log = 2 %>
  <% else %>
    <% vote_log = myvote[0][:vote] %>
    <%= vote_log==1?"voted Pro":"voted Con" %>
```

Views/causes/index.html.erb (3/3)

```
<%= submit_tag 'Clear', :name =>'Clear' %>
  <% end %>
</font>
</td>
<td>
  <% if vote_log != 1 %>
    <%= submit_tag 'Pro', :name =>'Pro' %>
  <% end %>
</td>
<td>
  <% if vote_log != 0 %>
    <%= tag :input, { :type => 'submit', :name => 'Con', :value => 'Con' } %>
  <% end %>
</td>
<% end %>
</tr>
<% end %>
</table>

<br />

<%= link_to 'back', problems_path %>
```

Causes#index

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.id)

respond_to do |format|

format.html *# index.html.erb*

format.json { render :json => @causes }

end

end

Causes#vote (1/3)

```
# 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])
  @votes = CauseVote.find_all_by_guest_id(current_user.id)
  myvote = @votes.select{|vote| vote[:cause_id]==@cause.id }
  if myvote.length == 0 then
    vote_log = 2
  else
    vote_log = myvote[0][:vote]
    if vote_log==1 then
      @cause.update_attributes( { :pros => @cause.pros-1 } )
    else
      @cause.update_attributes( { :cons => @cause.cons-1 } )
    end
  end
end
```


Causes#vote (2/3)

```
if params[:Pro] then
  @vote = 'Pro'
  @cause.update_attributes( { :pros => @cause.pros+1 } )
elsif params[:Con] then
  @vote = 'Con'
  @cause.update_attributes( { :cons => @cause.cons+1 } )
end
vote_param = {
  :cause_id => @cause.id,
  :guest_id => current_user.id,  # it should be guest_id actually
  :vote => (@vote=='Pro'?1:0)
}
```

Causes#vote (3/3)

```
if myvote.length==0 then
  @cause_vote = CauseVote.new( vote_param )
  @cause_vote.save
elsif params[:Clear] then
  @cause_vote = CauseVote.find( myvote[0][:id] )
  @cause_vote.destroy
else
  @cause_vote = CauseVote.find( myvote[0][:id] )
  @cause_vote.update_attributes( vote_param )
end

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

Final Screen for Today



PROBLEM SOLVING ENGINE

[TOP](#) | [Register new Problem](#) | [ruby Official Site](#) | [My Twitter\(Not Ready\)](#) | [Sign out](#)

[AD Space](#)

[FOR RENT](#)

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. voted Pro <input type="button" value="Clear"/>	9	12	Show Edit Destroy <input type="button" value="Con"/>
Earthmen are bellicose by nature. voted Con <input type="button" value="Clear"/>	1	2	Show Edit Destroy <input type="button" value="Pro"/>
Lady Gaga goes so wild.	0	0	Show Edit Destroy



[Not voted yet.](#)

[back](#)



Report Theme of Today

No report is required.

But, catch up and display the screen showed in the previous page.

Session Information

Add one line,

`p session`

in some where of your ruby source code.

I put it in `causes#index`, and checked the information. What is it?

```
class CausesController < ApplicationController
  # GET /causes
  # GET /causes.json
  def index
    p session
    @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.id)
```

```
{"warden.user.user.key"=>["User", [1], "$2a$10$gWYsD1nujj.UuZH5gousd0"], "session_id"=>"50c882aba2bf72969df5bc6c78e63a42", "flash"=>#<ActionDispatch::Flash::FlashHash:0x102be9968 @flashes={}, @closed=false, @used=#<Set: {}>, @now=nil>, "_csrf_token"=>"0fwaVdLJjFJECIfNqnzvCbIsGeUbwiVpLeM2SkZ8LCI="}
```

Forget or Remember?

Even when a user closed a browser, still underlying connections between a client machine and a server could be kept.

So we can support two different style of connection (session) management.

- 1) Forget user, (reset the connection) when a user closes the browser.
- 2) Remember user, unless the user explicitly 'sign out' from the server, and keep the session information even when the client computer shutdown.

Also see cookies

Also, add one line,

`p cookies`

in some where of your ruby source code.

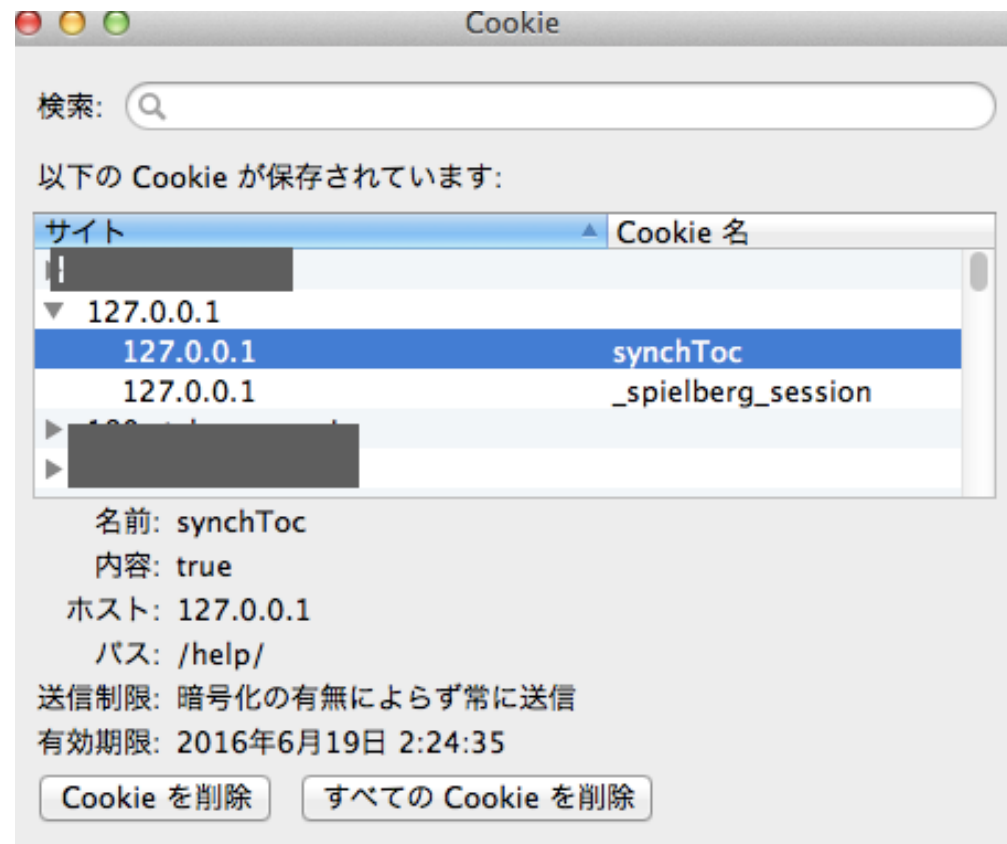
```
#<ActionDispatch::Cookies::CookieJar:0x1028f9b18 @delete_cookies={}, @cookies={"_spielberg_session"=>"BAh7CCIZd2FyZGVuLnVzZXIudXNlci5rZXlbCCIjVXNlcjsGaQYiIiQyYSQxMCRnV1lzRDFudWpqLV1Wkg1Z291c2RPIg9zZXNzaW9uX2lkIiU1MGM4ODJhYmEyYmY3Mjk2OWRmNWJjNmM3OGU2M2E0MiIQX2NzcmZfdG9rZW4iMU9md2FWZExKakZKRUNJZk5xbnp2Q2Jsc0dlVWJ3aXZwTGVMlNrWjhMQ0k9--786c20e911e7767fe85c1404e8475a0ca7a5e842"}, @secure=false, @host="127.0.0.1", @closed=false, @set_cookies={}, @secret="586cf2f215d0ca0ce1936dd7b2951f2f4b73a13d990502aab6899503e5d5a57975d40d15b3a5e28addf986e75ffd7db6e5b3c804d47f19a7e22cd060382defdc", @signed=#<ActionDispatch::Cookies::SignedCookieJar:0x1028f9c58 @verifier=#<ActiveSupport::MessageVerifier:0x1028f99b0 @serializer=Marshal, @digest="SHA1", @secret="586cf2f215d0ca0ce1936dd7b2951f2f4b73a13d990502aab6899503e5d5a57975d40d15b3a5e28addf986e75ffd7db6e5b3c804d47f19a7e22cd060382defdc">, @parent_jar=#<ActionDispatch::Cookies::CookieJar:0x1028f9b18 ...>>>
```

Where are cookies?

In a cupboard in the kitchen.... (Sorry!)

Cookies are
stored in the
client local
computer.

See the browser's
screen, and we
can see the list
on the browser.



Check and see cookies

If you have not see the cookies screen, I advise you to open cookies maintenance screen on the browser, and count how many cookies your computer has!

I think most of those are not harmful, still they are sometimes used to execute some malwares.

By mentioning the word 'cookie', and I would like to conclode today's course.

Prepare for the Next Week

We discuss the 'extension' of our project, "Problem Solving Engine."

If we allow the participants of the system to debate on certain topics, what kind of design could be possible?

If we allow the chairman of certain topics to control the discussion, what kind of design and screen could be possible?

Let us think about all of these WEB system design, based on Object Oriented Characteristics.