



OBJECT ORIENTED WEB PROGRAMMING USING RUBY

Day 9: 14/June/2012

Table Structure

Today's Goal (From syllabus)

□ Relational structure of tables are studied so that students could construct the database with complicated data architecture.

□ 'Complicated Architecture?'

I do not think so, but, this is the basis of constructing the large scale system.

I should have written as 'data architecture with many associated factors.'

Trouble Shooting

At the beginning, I had written the instruction as to remove the following line in the file layouts/application.html.erb.

```
<%= javascript_include_tag "application" %>
```

Please resurrect this line, instead, remove the following file.

```
app/assets/javascripts/*.js.coffee
```

This will allow us destroy the record, and sign out from the system.

To cope with Rails

'Rails' is a very aggressive platform to intake every useful concepts and technologies, even many supporters seem to complain to stay conservative. It is really tough to keep in touch with the latest rails specification to proceed this course lecture. As an old programmer, I myself want to stay in a stable environment without any stimulating progressive technology, however, the course students are young engineers who may change the future computer technology. So, I would surely make mistakes in the future, but I dare try to intake the latest technology in this course.

Students, please learn from my mistakes. I will try to explain how I have made mistakes, and what I should have done with those new technologies.

CoffeeScript vs. JavaScript

CoffeeScript had become a standard scripting factor of rails from version 3.1.

CoffeeScript is a language to generate JavaScript. (I understand) CoffeeScript and JavaScript are the same as 'scss' and 'css.'

Rails tends to intake the language with much simple, structured and short expression to generate other language source. (Personally, I like this attitude.)

CoffeeScript vs. Cygwin

With the installation instruction I had provided for the course, might have not been sufficient for this course, especially with Cygwin on Windows.

I should have added (probably one) another command to support CoffeeScript, I guess.

For the lecture of this semester, please forget about CoffeeScript.

Screen of the last day(After login)



PROBLEM SOLVING ENGINE

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

translation missing: ja.devise.sessions.user.signed_in

AD Space
FOR RENT

Listing problems

Title	Content	
World is not peaceful	There are terrorism and wars. How we can cope with them?	Show Edit Destroy

[New Problem](#)

We want [Add Cause] link here.

App/views/problems/index.html.erb

```
<h1>Listing problems</h1>
```

```
<table>
```

```
<tr>
```

```
<th>Title</th>
```

```
<th>Content</th>
```

```
<th></th>
```

```
<th></th>
```

```
<th></th>
```

```
</tr>
```

This '@problem' is from controllers method 'index.'

```
def index
  @problems = Problem.all

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

```
<% @problems.each do |problem| %>
```

```
<tr>
```

```
<td><%= problem.title %></td>
```

```
<td><%= problem.content %></td>
```

```
<td><%= link_to 'Show', problem %></td>
```

```
<td><%= link_to 'Edit', edit_problem_path(problem) %></td>
```

```
<td><%= link_to 'Destroy', problem, :confirm => 'Are you sure?', :method => :delete %></td>
```

```
</tr>
```

```
<% end %>
```

```
</table>
```

```
<br />
```

```
<%= link_to 'New Problem', new_problem_path %>
```

We want to add link here.

From 'Problems' to register 'Causes'

Today's first link is from
'Problems#index' (top screen) to register
'new Causes.'

Here, we want to hand a parameter
'problem_id' (primary key number) to
the new method of 'causes' controller.

controllers/causes_controller.rb

When it is linked to `'new_cause_path,'`
`'causes'` controller is called at the method
`'new.'`

New record `@cause` is created, and this
parameter is handed to `new.html.erb`.

```
23  
24 # GET /causes/new  
25 # GET /causes/new.json  
26 def new  
27   @cause = Cause.new  
28  
29   respond_to do |format|  
30     format.html # new.html.erb  
31     format.json { render :json => @cause }  
32   end  
33 end  
34
```

config/routes.rb

Add one line

```
match 'causes/new(/:problem_id)', :to => 'causes#new'
```

:product_id is an optional parameter, so round brackets should be put around the parameter name.

```
14 # Sample of regular route:
15 #   match 'products/:id' => 'catalog#view'
16 # Keep in mind you can assign values other than :controller and :action
17 match 'causes/new(/:problem_id)', :to => 'causes#new'
18
```

Link source

At `app/views/problems/index.html.erb`,
add the following line;

```
<td><%= link_to 'Register Cause', :controller => :causes,  
              :action => :new, :problem_id => problem.id %></td>
```

Here, if there were no necessity to hand `:problem_id`
parameter, just `'new_cause_path'` had been fine, like

```
<td><%= link_to 'Register Cause', new_cause_path %></td>
```

```
12 <% @problems.each do |problem| %>  
13   <tr>  
14     <td><%= problem.title %></td>  
15     <td><%= problem.content %></td>  
16     <td><%= link_to 'Register Cause', :controller => :causes,  
17           :action => :new, :problem_id => problem.id %></td>  
18     <td><%= link_to 'Show', problem %></td>  
19     <td><%= link_to 'Edit', edit_problem_path(problem) %></td>  
20     <td><%= link_to 'Destroy', problem, :confirm => 'Are you sure?', :method => :d  
21   </tr>  
22 <% end %>
```

Link source modification

Root screen has become as the following;

Listing problems

AL
FC

Title	Content	
World is not peaceful	There are terrorism and wars. How we can cope with them?	Register Cause
New Problem		Show Edit Destroy

Modify Controller

Add one line in the 'new' method of causes controller,
app/controllers/causes_controller.rb

```
@cause.problem = Problem.find(params[:problem_id])
```

```
24 # GET /causes/new
25 # GET /causes/new.json
26 def new
27   @cause = Cause.new
28   @cause.problem = Problem.find(params[:problem_id])
29
30 respond_to do |format|
31   format.html # new.html.erb
32   format.json { render :json => @cause }
33 end
34 end
```

Relation from Cause to Problem

Cause.new generated new Cause instance,
and this instance was given the relation to
Problem with

`Problem.find(params[:problem_id])`

Now we can refer the the Problem instance
by writing `cause.problem`.

app/views/causes/new.html.erb

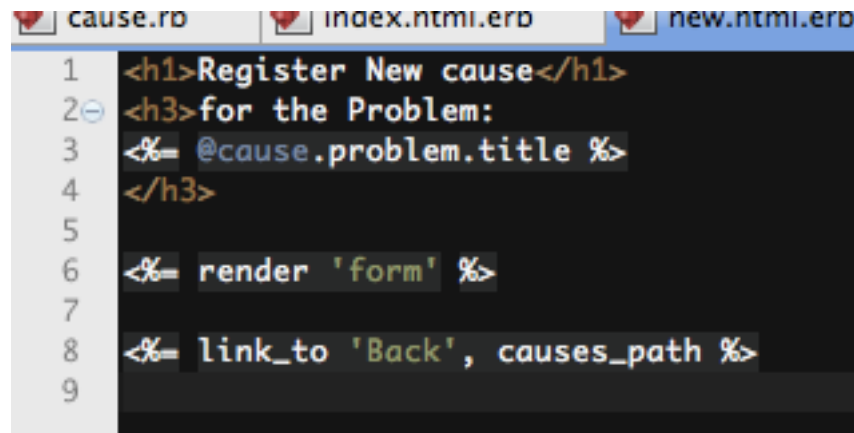
Modify the headings as;

```
<h1>Register New cause</h1>
```

```
<h3>for the Problem:
```

```
<%= @cause.problem.title %>
```

```
</h3>
```



```
cause.rb | index.html.erb | new.html.erb
1 <h1>Register New cause</h1>
2 <h3>for the Problem:
3 <%= @cause.problem.title %>
4 </h3>
5
6 <%= render 'form' %>
7
8 <%= link_to 'Back', causes_path %>
9
```


'Register Causes'

Now when we click the 'Register Causes' link, we can see the following screen.



The screenshot shows the 'Register New cause' page on the 'PROBLEM SOLVING ENGINE' website. The page features a navigation bar with links for 'TOP', 'Register new Problem', 'ruby Official Site', and 'My T'. The main heading is 'Register New cause' for the problem 'World is not peaceful'. Below this, there is a 'Fact' label and a large empty text input box.

PROBLEM SOLVING ENGINE

[TOP](#) | [Register new Problem](#) | [ruby Official Site](#) | [My T](#)

Register New cause

for the Problem: World is not peaceful

Fact

app/views/causes/_form.html.erb

Originally, the file was as the following;
Now the attribute :pros and :cons should be counters and start with 0, so remove those fields, and shrink the size of :fact field to "60x5."

```
13  
14 <div class="field">  
15   <%= f.label :fact %><br />  
16   <%= f.text_area :fact %>  
17 </div>  
18 <div class="field">  
19   <%= f.label :pros %><br />  
20   <%= f.number_field :pros %>  
21 </div>  
22 <div class="field">  
23   <%= f.label :cons %><br />  
24   <%= f.number_field :cons %>  
25 </div>  
26 <div class="actions">  
27   <%= f.submit %>  
28 </div>  
29 <%= end %>  
30
```

Also, we add one line

```
<%= f.hidden_field :problem_id, :value => @cause.problem.id %>
```

app/views/causes/_form.html.erb

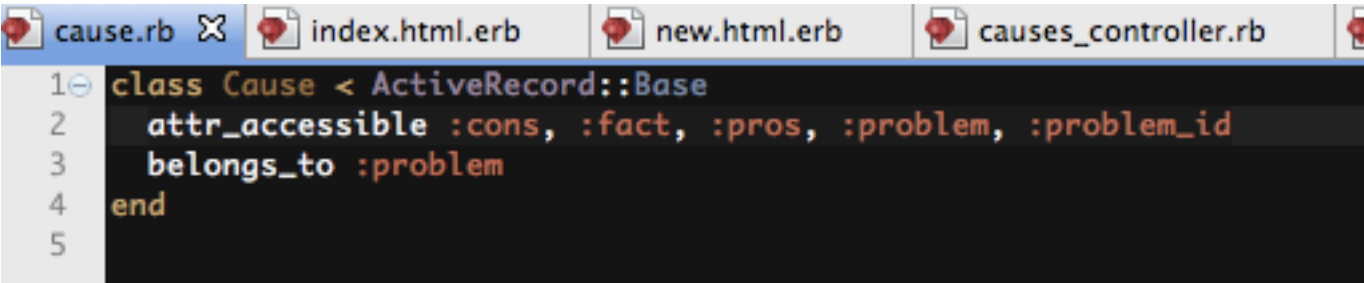
After the modification, the file is now as

```
1 <%= form_for(@cause) do |f| %>
2   <% if @cause.errors.any? %>
3     <div id="error_explanation">
4       <h2><%= pluralize(@cause.errors.count, "error") %> prohibited this cause from b
5
6       <ul>
7         <% @cause.errors.full_messages.each do |msg| %>
8           <li><%= msg %></li>
9         <% end %>
10      </ul>
11    </div>
12  <% end %>
13
14  <div class="field">
15    <%= f.label :fact %><br />
16    <%= f.text_area :fact, :size => "60x10" %>
17  </div>
18  <%= f.hidden_field :problem_id, :value => @cause.problem.id %>
19  <div class="actions">
20    <%= f.submit %>
21  </div>
22 <% end %>
23
```

To allow controller

to set the value to `problem_id`, modify
`app/models/cause.rb`

Add the following to `attr_accessible`;
`:problem_id`



```
cause.rb x index.html.erb new.html.erb causes_controller.rb
1 class Cause < ActiveRecord::Base
2   attr_accessible :cons, :fact, :pros, :problem, :problem_id
3   belongs_to :problem
4 end
5
```

Parameters' round trip

'New' method is executed in the server, and html is handled in the client browser. Then the form input was handed to the 'create' method in the server.

When we test run the program only on our computer, we cannot notice that the protocol message goes to the browser and comes back from the browser.

So, if we want to initialize parameter, we need to do in in the 'create' method, or let the parameter go and back in the hidden field.

Create method

Add two lines;

```
@cause.pros = 0
```

```
@cause.cons = 0
```

```
41 # POST /causes
42 # POST /causes.json
43 def create
44   @cause = Cause.new(params[:cause])
45   @cause.pros = 0
46   @cause.cons = 0
47
48   respond_to do |format|
49     if @cause.save
50       format.html { redirect_to @cause, :notice => 'Cause was successfully created.' }
51       format.json { render :json => @cause, :status => :created, :location => @cause }
52     else
53       format.html { render :action => "new" }
54       format.json { render :json => @cause.errors, :status => :unprocessable_entity }
55     end
56   end
57 end
```

Register New Cause Screen

Is now like



PROBLEM SOLVING ENGINE

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

[AD Space](#)
[FOR RENT](#)

Register New cause

for the Problem: World is not peaceful

Fact

There is tribal hostility.

[登録する](#)

[Back](#)

[TOP](#) | [Register new Problem](#) | [ruby Official Site](#) |

Cause was successfully created.

Cause was successfully created.

Fact: There is tribal hostility.

Pros: 0

Cons: 0

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

Causes original index screen

'causes' table's original index screen was as the following;



PROBLEM SOLVING ENGINE

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

Listing causes

Fact	Pros	Cons	
There is tribal hostility.	0	0	Show Edit Destroy

[New Cause](#)

Causes index modification

In the Causes index(list) screen, we want to see only the causes records related to the specified problems.

If not so, the following problems would occur.

Problems Top Screen

Sample

Listing problems

Title	Content		
World is not peaceful	There are terrorism and wars. How we can cope with them?	Register Cause	Show Edit Destroy
I cannot get married.	No other explanation.	Register Cause	Show Edit Destroy
I cannot get a divorce.	No other explai		

Problematic Causes screen.

Listing causes

New Problem	Fact	Pros	Cons	
	There is tribal hostility.	0	0	Show Edit Destroy
	Earthmen are bellicose by nature.	0	0	Show Edit Destroy
	I am unemployed.	0	0	Show Edit Destroy
	I am so rich.	0	0	Show Edit Destroy
	New Cause			

Let Index of Causes be related with a certain Problem

To replace index of causes controller related with specific problem, first modify `config/routes.rb`

Add one line;

```
match 'causes/index(/:problem_id)', :to => 'causes#index'
```

```
13
14 # Sample of regular route:
15 # match 'products/:id' => 'catalog#view'
16 # Keep in mind you can assign values other than :controller and :action
17 match 'causes/new(/:problem_id)', :to => 'causes#new'
18 match 'causes/index(/:problem_id)', :to => 'causes#index'
19
20 # Sample of named route:
```

Add new link to Top screen

Modify

`app/views/problems/index.html.erb`

Add one (logical) line;

```
<td><%= link_to 'List Causes', :controller => :causes,  
  :action => :index, :problem_id => problem.id %></td>
```

```
12 <%= @problems.each do |problem| %>  
13   <tr>  
14     <td><%= problem.title %></td>  
15     <td><%= problem.content %></td>  
16     <td><%= link_to 'List Causes', :controller => :causes,  
17       :action => :index, :problem_id => problem.id %></td>  
18     <td><%= link_to 'Register Cause', :controller => :causes,  
19       :action => :new, :problem_id => problem.id %></td>  
20     <td><%= link_to 'Show', problem %></td>  
21     <td><%= link_to 'Edit', edit_problem_path(problem) %></td>  
22     <td><%= link_to 'Destroy', problem, :confirm => 'Are you sure?', :meth  
23   </tr>  
24 <%= end %>  
25 </table>
```

Index method of causes controller

Modify

`app/controllers/causes_controller.rb`

Add two lines;

`@causes = Cause.find_all_by_problem_id(params[:problem_id])`

`@problem = Problem.find(params[:problem_id])`

```
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
8  respond_to do |format|
9    format.html # index.html.erb
10   format.json { render :json => @causes }
11  end
12 end
```

Modify causes index

Modify

`app/views/causes/index.html.erb`

Modify headings;

`<h3>`

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

`</h3>`

And then, remove

link to `new_cause_path`, (because, create a cause should always be with `'problem_id,'` so, let it go back the top.

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

```
1 <h1>Listing causes</h1>
2 <h3>
3   of the problem: <%= @problem.title %>
4 </h3>
5
```

```
29 <br />
30
31 <%= link_to 'back', problems_path %>
32
```

Now the screens are

better, just like;

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](#)

Listing causes

of the problem: I cannot get married.

Fact	Pros	Cons	
I am unemployed.	0	0	Show Edit Des

[back](#)

Listing causes

of the problem: I cannot get a divorc

Fact	Pros	Cons	
I am so rich.	0	0	Show Edit Destroy

[back](#)

Now the top screen is



PROBLEM SOLVING ENGINE

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

AD Space
FOR RENT

Listing problems

Title	Content	List Causes	Register Cause	Show Edit Destroy
World is not peaceful	There are terrorism and wars. How we can cope with them?	List Causes	Register Cause	Show Edit Destroy
I cannot get married.	No other explanation.	List Causes	Register Cause	Show Edit Destroy
I cannot get a divorce.	No other explanation.	List Causes	Register Cause	Show Edit Destroy

[New Problem](#)

Today's Practice

Finish the relationship design between Problems and Causes.

Then, do the similar modification to the relationship between Causes and solutions, to introduce the solutions screen.

Report Topics for today.

Modify the code to design the causes-solutions relationship screen, and report the code and screens, in English.

Also, there are several points that may cause the crash of the system.

(The modification is not completed yet.)

Please point out the problem of the program and report those.

Prepare for the Next Week

In today's modification, we did not care for the guest who added the new record.

So, we add the link field to guests in all tables to record the guests, and allow only the guests who added to remove the problems, causes and/or solutions.

Change the views depending on the guests.

Also, add 'pros' and 'cons' buttons for guests to vote. If once a guest voted, display the screen with the mark on 'pros' or 'cons' for his screen.