



---

# OBJECT ORIENTED WEB PROGRAMMING USING RUBY

Day 8: 7/June/2012

Log-in Authentication

# Today's Goal (From syllabus)

---

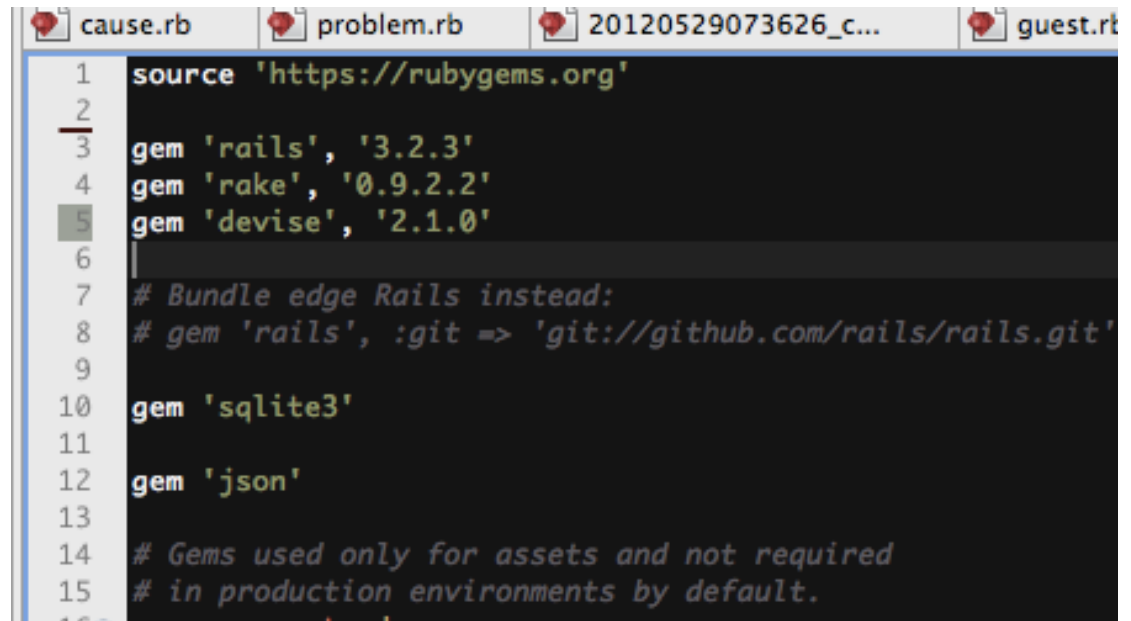
- ❑ Learn authentication so that only specific users can use the Web information of the system.
- ❑ We use 'Devise' to

# Fix Gemfile

---

- ▣ Add one line to the file 'project/Gemfile'

gem 'devise', '2.1.0'



```
cause.rb  problem.rb  20120529073626_c...  guest.r...  
1 source 'https://rubygems.org'  
2  
3 gem 'rails', '3.2.3'  
4 gem 'rake', '0.9.2.2'  
5 gem 'devise', '2.1.0'  
6  
7 # Bundle edge Rails instead:  
8 # gem 'rails', :git => 'git://github.com/rails/rails.git'  
9  
10 gem 'sqlite3'  
11  
12 gem 'json'  
13  
14 # Gems used only for assets and not required  
15 # in production environments by default.  
16
```

# Bundle install

---

▣ Type

bundle install

▣ In the project folder

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ bundle install
Fetching source index for https://rubygems.org/
Enter your password to install the bundled Ruby gems:
Using rake (0.9.2.2)
Using i18n (0.6.0)
Using multi_json (1.3.4)
Using activesupport (3.2.3)
Using builder (3.0.0)
Using activemodel (3.2.3)
Using erubis (2.7.0)
Using journey (1.0.3)
Using rack (1.4.1)
Using rack-cache (1.2)
Using rack-test (0.6.1)
Using hike (1.2.1)
Using tilt (1.3.3)
Using sprockets (2.1.3)
Using actionpack (3.2.3)
Using mime-types (1.18)
Using polyglot (0.3.3)
Using tree-top (1.4.10)
Using mail (2.4.4)
Using actionmailer (3.2.3)
Using arel (3.0.2)
Using tzinfo (0.3.33)
Using activerecord (3.2.3)
Using activeresource (3.2.3)
Installing bcrypt-ruby (3.0.1) with native extensions
Using bundler (1.0.15)
Using coffee-script-source (1.3.1)
Using execjs (1.3.1)
Using coffee-script (2.2.0)
Using rack-ssl (1.3.2)
Using json (1.7.0)
Using rdoc (3.12)
Using thor (0.14.6)
Using railties (3.2.3)
Using coffee-rails (3.2.2)
Installing orm_adapter (0.0.7)
Installing warden (1.1.1)
Installing devise (2.1.0)
Using jquery-rails (2.0.2)
Using rails (3.2.3)
Using sass (3.1.16)
Using sass-rails (3.2.5)
Using sqlite3 (1.3.6)
Using uglifier (1.2.4)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

# Confirm installation of the Gem

---

▣ Type  
`gem list devise`

Make sure that  
Devise (2.1.0)  
replied.

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ gem list devise  
  
*** LOCAL GEMS ***  
  
devise (2.1.0)  
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

# Install Devise to the application

---

□ Type

rails generate devise:install

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rails generate devise:install
create  config/initializers/devise.rb
create  config/locales/devise.en.yml
=====

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here
   is an example of default_url_options appropriate for a development environme
nt
   in config/environments/development.rb:

       config.action_mailer.default_url_options = { :host => 'localhost:3000' }

   In production, :host should be set to the actual host of your application.

2. Ensure you have defined root_url to *something* in your config/routes.rb.
   For example:

       root :to => "home#index"

3. Ensure you have flash messages in app/views/layouts/application.html.erb.
   For example:

       <p class="notice"><%= notice %></p>
       <p class="alert"><%= alert %></p>

4. If you are deploying Rails 3.1 on Heroku, you may want to set:

       config.assets.initialize_on_precompile = false

   On config/application.rb forcing your application to not access the DB
   or load models when precompiling your assets.
=====
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ █
```

# Read the message from the system carefully

---

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here is an example of default\_url\_options appropriate for a development environment in config/environments/development.rb:

```
config.action_mailer.default_url_options = { :host => 'localhost:3000' }
```

In production, :host should be set to the actual host of your application.

2. Ensure you have defined root\_url to \*something\* in your config/routes.rb. For example:

```
root :to => "home#index"
```

3. Ensure you have flash messages in app/views/layouts/application.html.erb. For example:

```
<p class="notice"><%= notice %></p>
<p class="alert"><%= alert %></p>
```

4. If you are deploying Rails 3.1 on Heroku, you may want to set:

```
config.assets.initialize_on_precompile = false
```

On config/application.rb forcing your application to not access the DB or load models when precompiling your assets.

# Set default\_url\_options

---

- ❑ Add the following statement to /config/environments/development.rb
- ❑ read the set-up message carefully

`config.action_mailer.default_url_options = { :host => 'localhost:3000' }`

```
12 # Show full error reports and disable caching
13 config.consider_all_requests_local       = true
14 config.action_controller.perform_caching = false
15
16 # Don't care if the mailer can't send
17 config.action_mailer.raise_delivery_errors = false
18
19 config.action_mailer.default_url_options = { :host => 'localhost:3000' }
20
21 # Print deprecation notices to the Rails logger
22 config.active_support.deprecation = :log
23
24 # Only use best-standards-support built into browsers
25 config.action_dispatch.best_standards_support = :builtin
26
27 # Raise exception on mass assignment protection for Active Record models
28 config.active_record.mass_assignment_sanitizer = :strict
29
30 # Log the query plan for queries taking more than this (works
31 # with SQLite, MySQL, and PostgreSQL)
32 config.active_record.query_log_threshold = 0.5
```



# Welcome Screen

---

- We do not have Welcome screen yet.
- So let us create welcome screen now.

rails generate controller welcome index

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rails generate controller welcome
index
  create  app/controllers/welcome_controller.rb
  route  get "welcome/index"
  invoke erb
  create  app/views/welcome
  create  app/views/welcome/index.html.erb
  invoke test_unit
  create  test/functional/welcome_controller_test.rb
  invoke helper
  create  app/helpers/welcome_helper.rb
  invoke test_unit
  create  test/unit/helpers/welcome_helper_test.rb
  invoke assets
  invoke coffee
  create  app/assets/javascripts/welcome.js.coffee
  invoke scss
  create  app/assets/stylesheets/welcome.css.scss
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

# Set the Login Default Screen

---

- We set the Login Default Screen to the welcome screen, which we had created in the last page.

First, we modify

(project name)/config/routes.rb

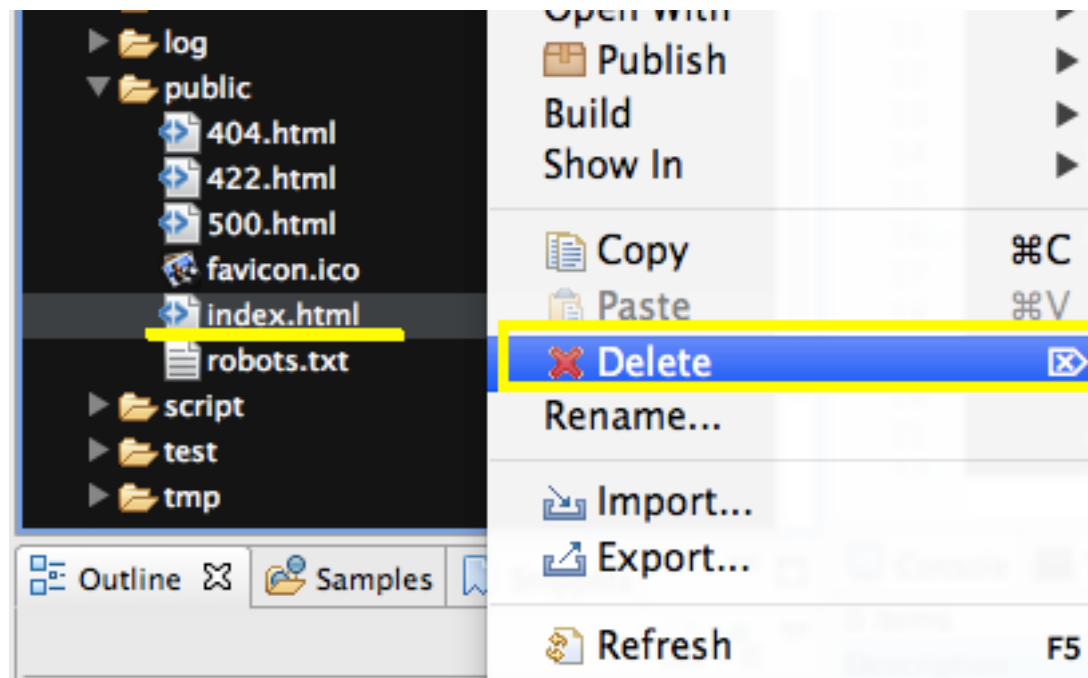
- Uncomment the line around line #61, to activate  
`root :to => 'welcome#index'`

```
51
52 # Sample resource route within a namespace:
53 # namespace :admin do
54 #   # Directs /admin/products/* to Admin::ProductsController
55 #   # (app/controllers/admin/products_controller.rb)
56 #   resources :products
57 # end
58
59 # You can have the root of your site routed with "root"
60 # just remember to delete public/index.html.
61 root :to => 'welcome#index'
62
63 # See how all your routes lay out with "rake routes"
64
65 # This is a legacy wild controller route that's not recommended for RESTful apps
66 # Note: This route will make all actions in every controller accessible via GET
67 # match ':controller(/:action(/:id))(.:format)'
68 end
```

# Remove public/index.html

---

We have to make sure to remove the file  
**public/index.html**



# Add two lines for login result message display

- Modify

(project)/app/views/layouts/application.html.erb

- Add the following two lines before `<%= yield %>`

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

`<p class="alert"><%= alert %></p>`

```
3 <head>
4   <title>Spielberg</title>
5   <%= stylesheet_link_tag "application", :media => "all" %>
6   <%= javascript_include_tag "application" %>
7   <%= csrf_meta_tags %>
8 </head>
9 <body>
10
11   <p class="notice"><%= notice %></p>
12   <p class="alert"><%= alert %></p>
13   <%= yield %>
14
15 </body>
16 </html>
17
```

# View for devise

---

- ▣ Here we generate views for devise. Type `rails generate devise:views`

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rails generate devise:views
  invoke  Devise::Generators::SharedViewsGenerator
  create  app/views/devise/shared
  create  app/views/devise/shared/_links.erb
  invoke  form_for
  create  app/views/devise/confirmations
  create  app/views/devise/confirmations/new.html.erb
  create  app/views/devise/passwords
  create  app/views/devise/passwords/edit.html.erb
  create  app/views/devise/passwords/new.html.erb
  create  app/views/devise/registrations
  create  app/views/devise/registrations/edit.html.erb
  create  app/views/devise/registrations/new.html.erb
  create  app/views/devise/sessions
  create  app/views/devise/sessions/new.html.erb
  create  app/views/devise/unlocks
  create  app/views/devise/unlocks/new.html.erb
  invoke  erb
  create  app/views/devise/mailer
  create  app/views/devise/mailer/confirmation_instructions.html.erb
  create  app/views/devise/mailer/reset_password_instructions.html.erb
  create  app/views/devise/mailer/unlock_instructions.html.erb
kobayashi-ikuo-no-MacBook:spielberg kobayashi$
```

# User Model for Authentication

---

- ▣ Generate Class User for devise. Type, rails generate devise user

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rails generate devise user
  invoke  active_record
  create   db/migrate/20120605172832_devise_create_users.rb
  create   app/models/user.rb
  invoke   test_unit
  create   test/unit/user_test.rb
  create   test/fixtures/users.yml
  insert   app/models/user.rb
  route    devise_for :users
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ █
```

# Users vs. Guests

---

Users table is for Authentication. It includes Guests, Administrators, and Topics Managers(?), or whoever need to login to the system must be enrolled as a user of the system.

Guests are the participants to the virtual discussion in this system.

# migration

---

- ▣ Now we migrate, using all 'automatic' result as default. Type `rake db:migrate`

```
kobayashi-ikuo-no-MacBook:spielberg kobayashi$ rake db:migrate
== DeviseCreateUsers: migrating =====
-- create_table(:users)
   => 0.0899s
-- add_index(:users, :email, {:unique=>true})
   => 0.0017s
-- add_index(:users, :reset_password_token, {:unique=>true})
   => 0.0010s
== DeviseCreateUsers: migrated (0.0929s) =====

kobayashi-ikuo-no-MacBook:spielberg kobayashi$ █
```



# Now, check the result on the way

---

- Start the server, and open <http://127.0.0.1:3000/>
  - Now, the following screen is given,
- Because, welcome/index.html has become the top page.

```
1 <h1>Welcome#index</h1>
2 <p>Find me in app/views/welcome/index.html.erb</p>
3
```

**Welcome#index**

Find me in app/views/welcome/index.html.erb

# Modify Welcome Screen

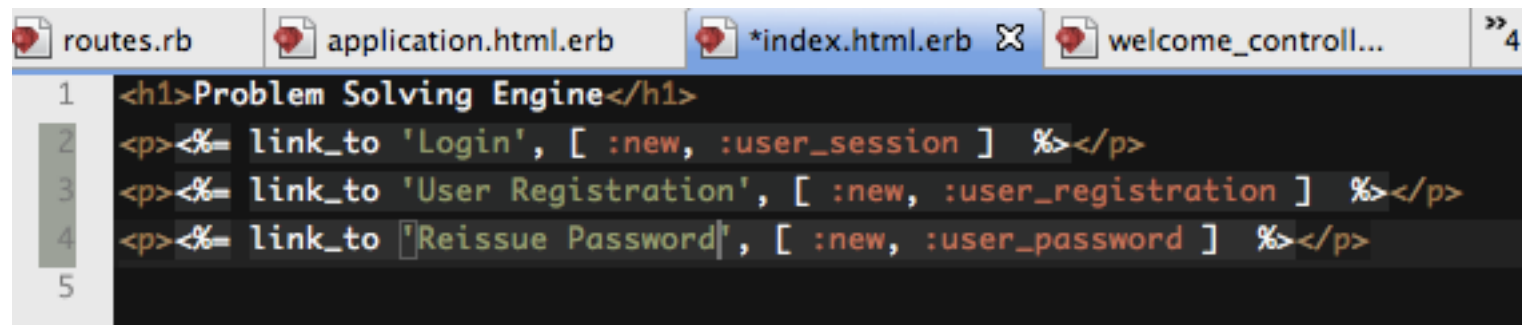
- Now we modify  
(project)/app/views/welcome/index.html.erb
- To lead to the login screen, which we have generated with the 'devise:views' command.

```
<h1>Problem Solving Engine</h1>
```

```
<p><%= link_to 'Login', [ :new, :user_session ] %></p>
```

```
<p><%= link_to 'User Registration', [ :new, :user_registration ] %></p>
```

```
<p><%= link_to 'Reissue Password', [ :new, :user_password ] %></p>
```

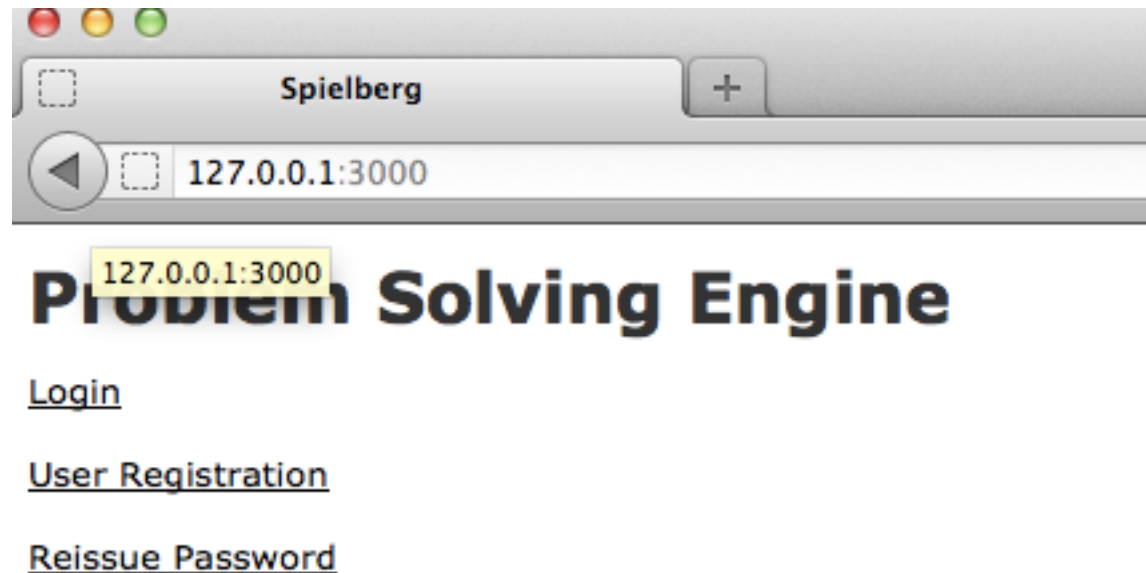
A screenshot of a code editor with four tabs: 'routes.rb', 'application.html.erb', '\*index.html.erb' (selected), and 'welcome\_controll...'. The selected tab shows the following code:

```
1 <h1>Problem Solving Engine</h1>
2 <p><%= link_to 'Login', [ :new, :user_session ] %></p>
3 <p><%= link_to 'User Registration', [ :new, :user_registration ] %></p>
4 <p><%= link_to 'Reissue Password', [ :new, :user_password ] %></p>
5
```

# New Top Screen

---

Now the Top page of PSE has become like the following



# Authentication Path/Redirection

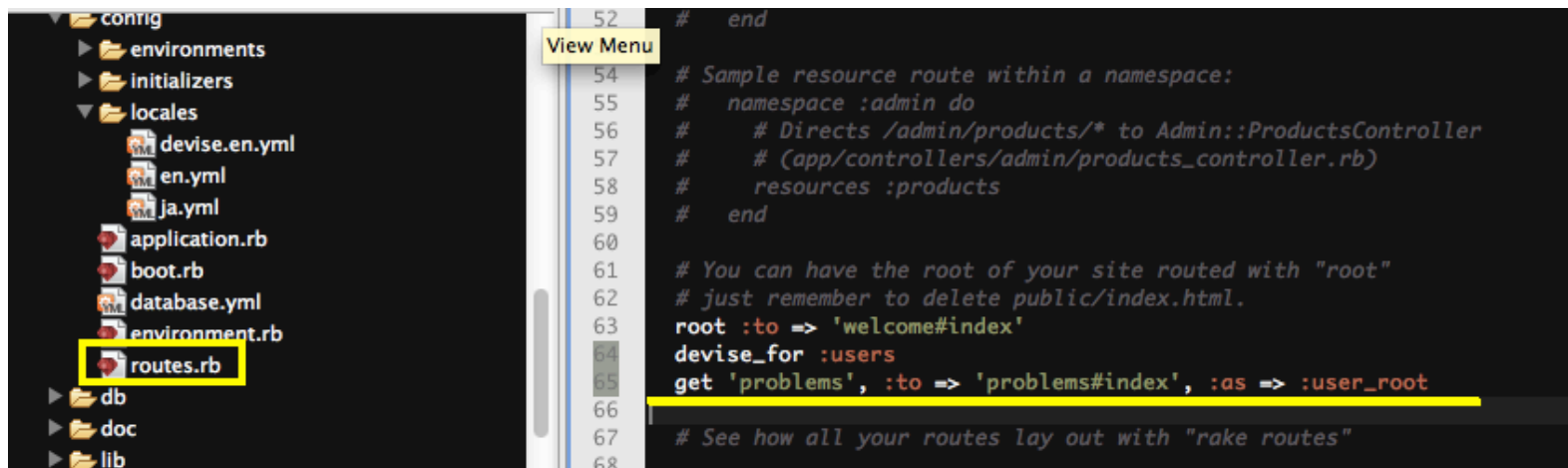
Once again, modify **config/routes.rb**, add two lines under the previous correction.

```
root :to => 'welcome#index'
```

```
devise_for :users
```

```
get 'problems', :to => 'problems#index', :as => :user_root
```

Reference: <http://railscasts.com/episodes/209-introducing-devise>



## config/routes.rb

---

devise\_for :users

- ▣ This modification is to register paths to the login form and user registration form.

get 'problems', :to => 'problems#index', :as => :user\_root

- ▣ This line is the redirection path after the authentication.

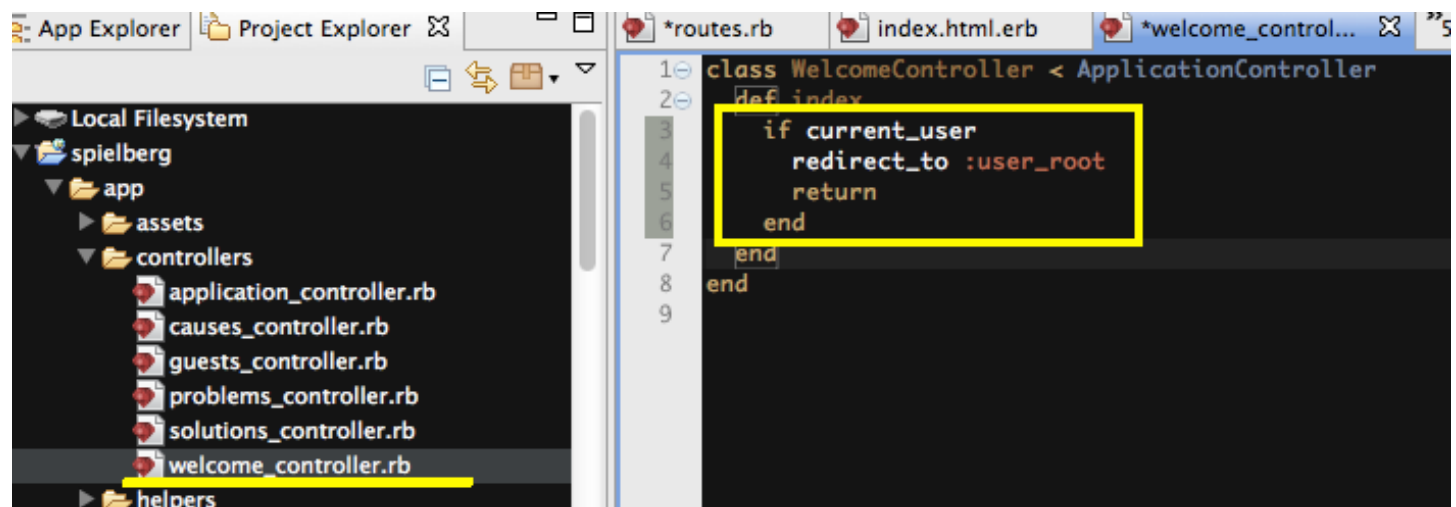
# welcome\_controller.rb

---

## □ Modify

(project)/app/controllers/welcome\_controller.rb

```
if current_user  
  redirect_to :user_root  
  return  
end
```



## welcome\_controller.rb

---

if current\_user

- ❑ Current\_user method returns the login user. Nil will be returned before login.

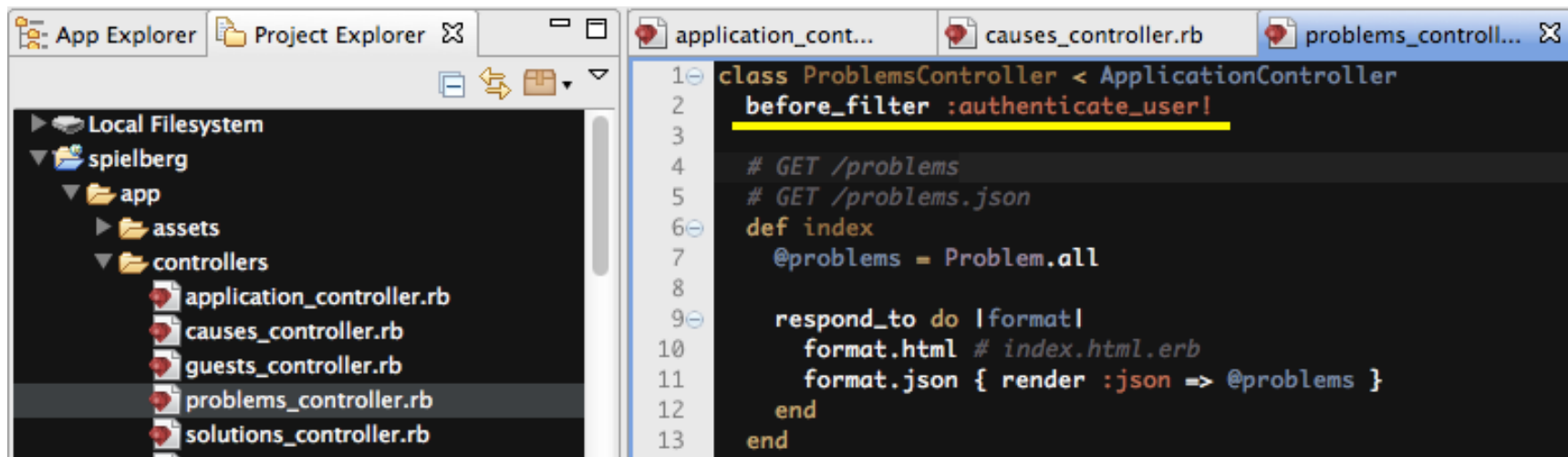
redirect\_to :user\_root

- ❑ When the user had logged-in already and accessed to the root path (/), this method would redirect the request to the :user\_root path (/problems).

# For other controllers

- ▣ Add authentication request as a `before_filter`

`before_filter :authenticate_user!`





# Arrange Screens

---

In the former semester, we had learnt the screen arrangement by modifying html files and css files.

- Though none of the course students joined the former course of Ruby on Rails Basics, I assume all of you would easily understand the html and css grammar, and proceed this screen arrangement.

# Prepare the title banner

---

Open any picture editing tool, to edit the title picture.

I myself opened paint brush, and edited new picture with the canvas size of width 400 x height 100, opened Office.com to import picture, searched with the word 'problem' to get a proper clipart, and added text 'Problem Solving Engine' to make the title.gif file as below.

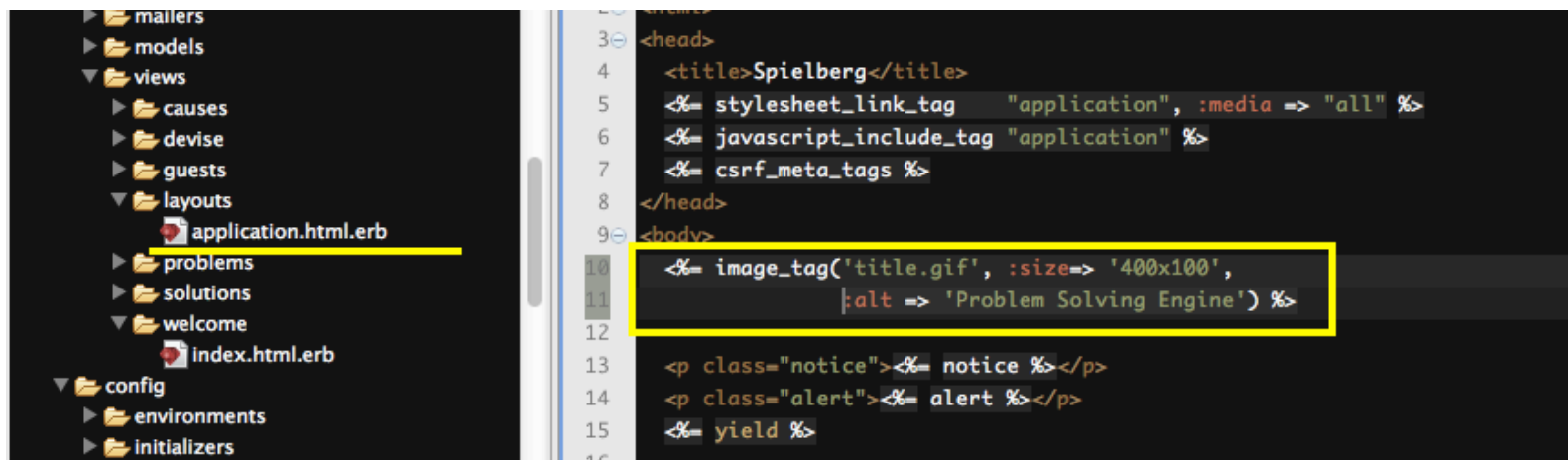


# Application.html.erb

Add one statement

```
<%= image_tag('title.gif', :size=> '400x100', :alt => 'Problem Solving Engine') %>
```

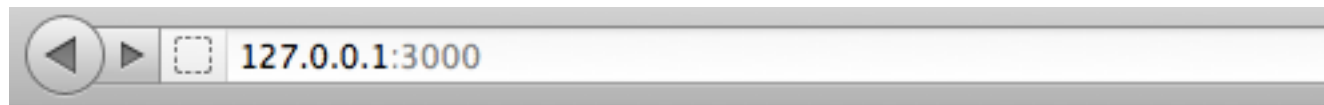
to **app/views/layouts/application.html.erb**



# Top Screen Now

---

is as the following.



PROBLEM SOLVING ENGINE

## **Problem Solving Engine**

[Login](#)

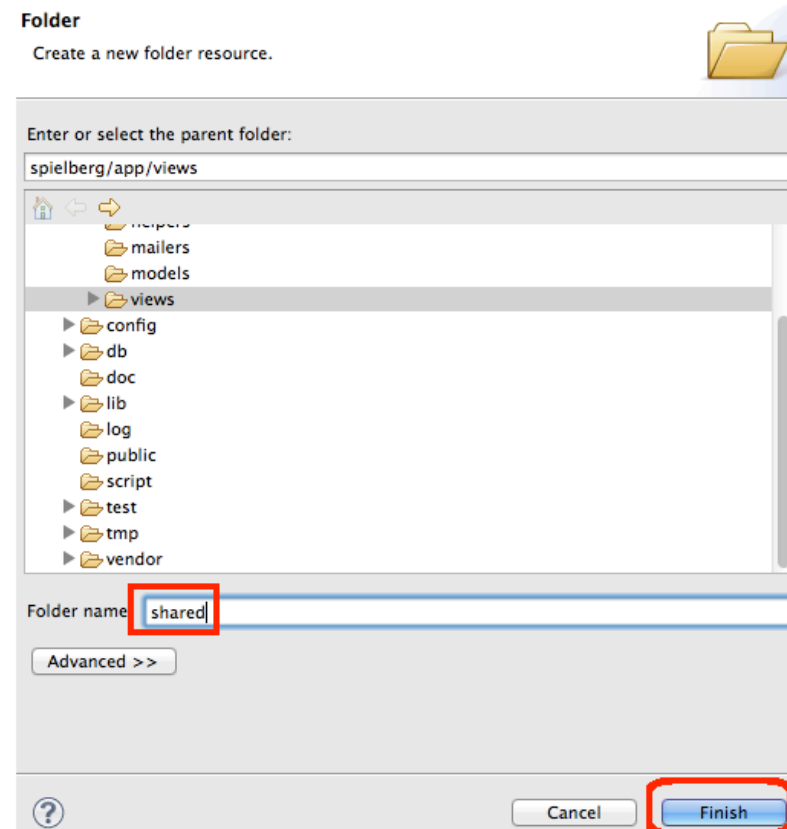
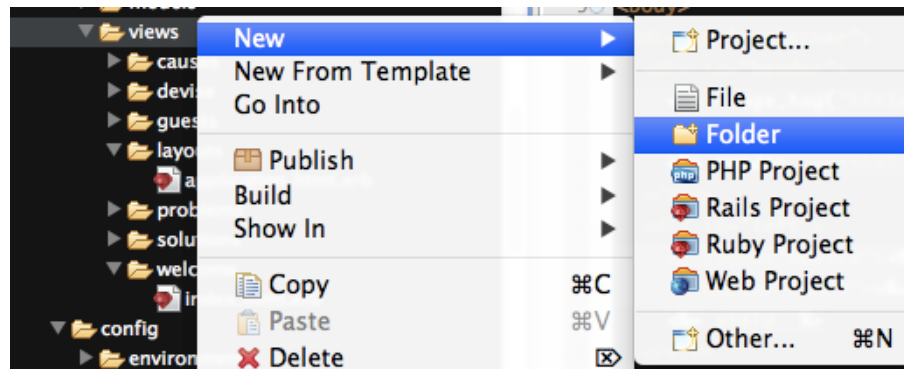
[User Registration](#)

[Reissue Password](#)

# Prepare the partial files

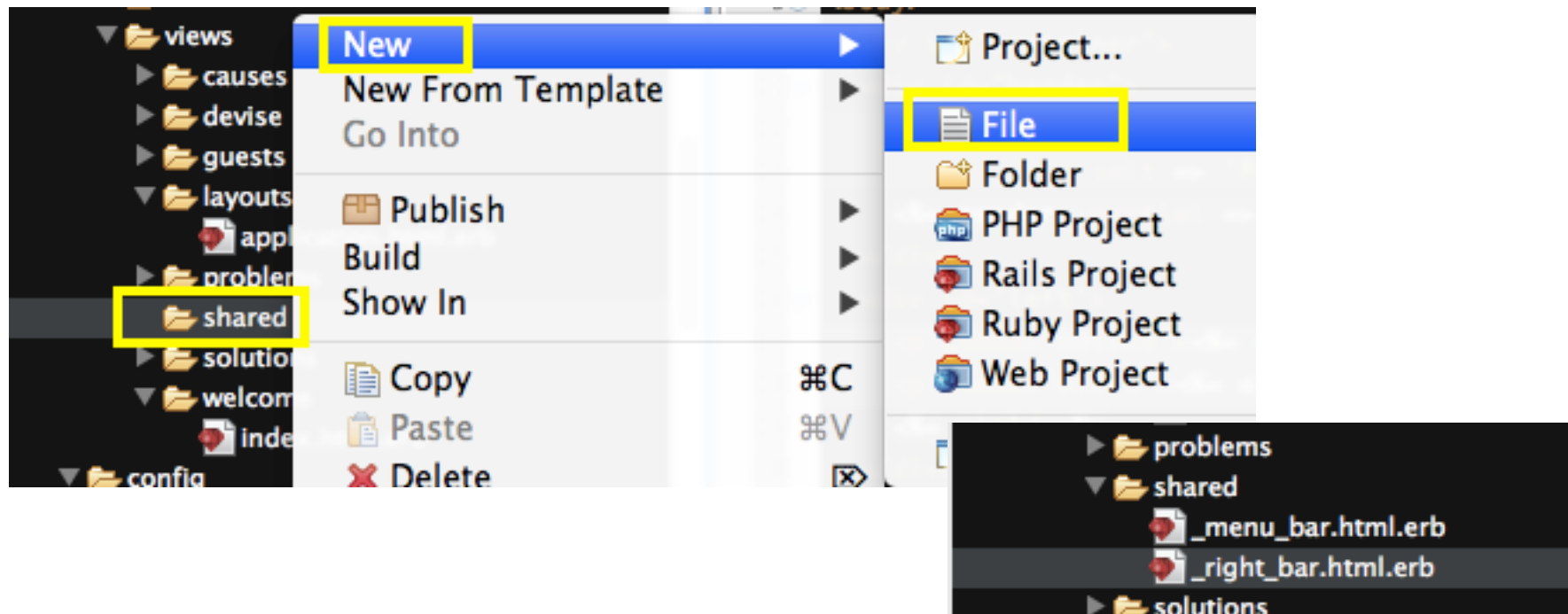
In order to add 'log-out' menu link, we now prepare the menu\_bar partial html.

Make shared folder in app/views.



# In 'shared,' prepare partial files

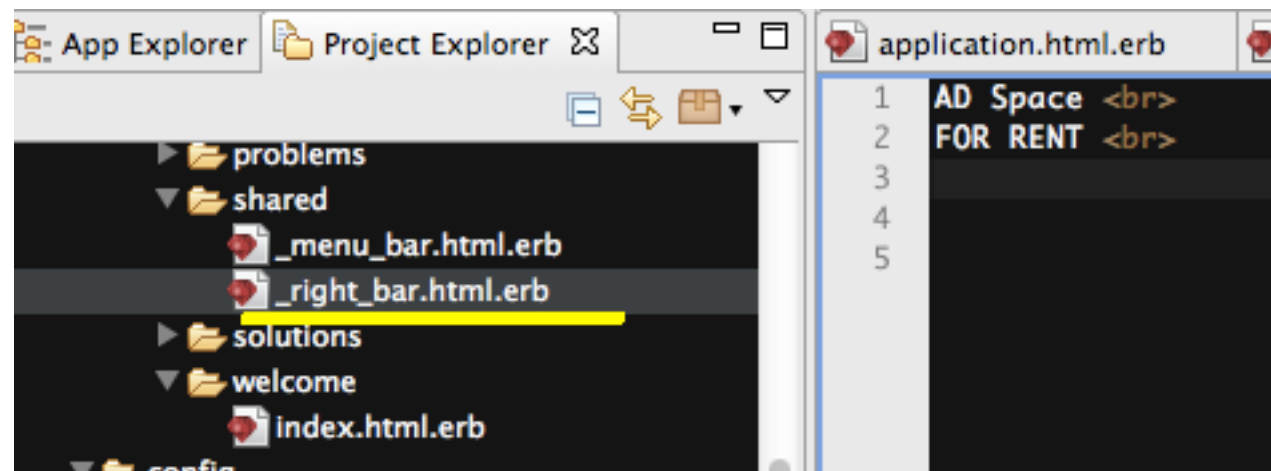
In app/views/shared, we create  
\_menu\_bar.html.erb, and  
\_right\_bar.html.erb



[illegible]

## \_right\_bar.html.erb

AD Space<br>  
FOR RENT<br>

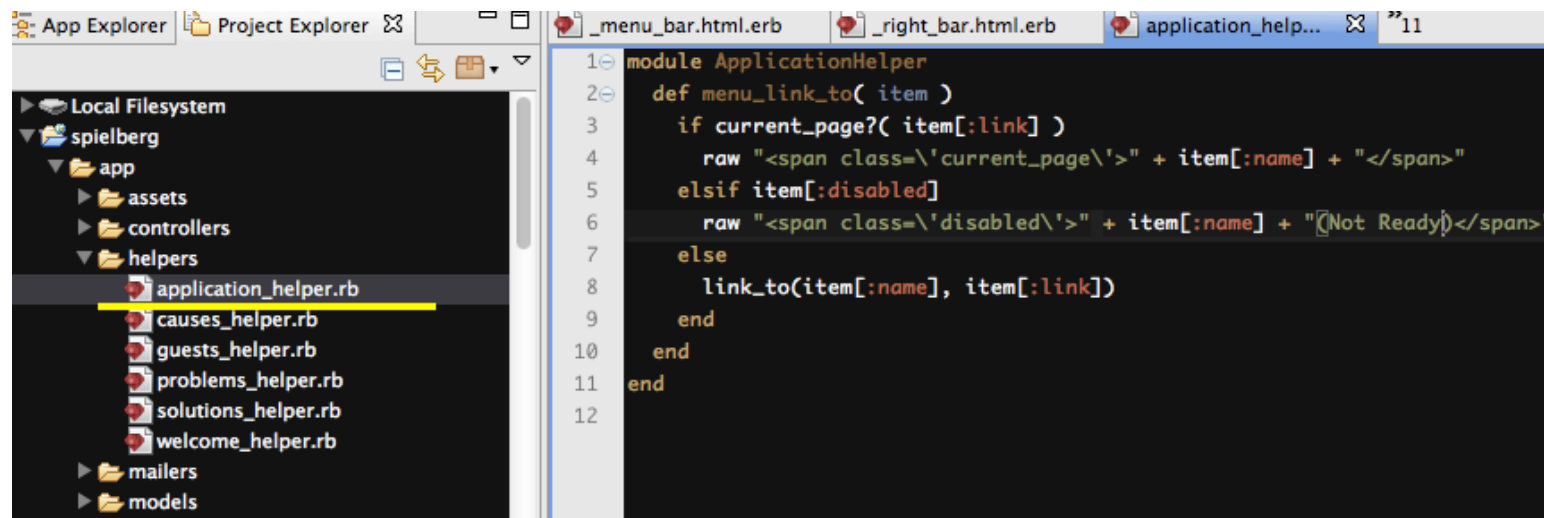




# Modify Application\_helper.rb

In `_menu_bar.html.erb`, we called the method `'menu_link_to.'`

So we need to describe this method, and it is usual to define it in `application_helper.rb`.



The screenshot shows a code editor with two panes. The left pane displays the 'Project Explorer' for a Rails application named 'spielberg'. The 'helpers' directory is expanded, showing several helper files, with 'application\_helper.rb' selected and highlighted in yellow. The right pane shows the code inside 'application\_helper.rb', which defines a module 'ApplicationHelper' and a method 'menu\_link\_to'.

```
1 module ApplicationHelper
2   def menu_link_to( item )
3     if current_page?( item[:link] )
4       raw "<span class='\current_page\'>" + item[:name] + "</span>"
5     elsif item[:disabled]
6       raw "<span class='\disabled\'>" + item[:name] + "(Not Ready)</span>"
7     else
8       link_to(item[:name], item[:link])
9     end
10  end
11 end
12
```

# Application\_helper.rb

---

```
module ApplicationHelper
  def menu_link_to( item )
    if current_page?( item[:link] )
      raw "<span class=\"current_page\">" + item[:name] + "</span>"
    elsif item[:disabled]
      raw "<span class=\"disabled\">" + item[:name] + "(Not Ready)</span>"
    else
      link_to(item[:name], item[:link])
    end
  end
end
```

# Divide Screens

---

To divide view into 'top', 'left' and 'right', we use `<div>` tag in `application.html.erb`

```
<div id="container">
  <div id="header">
    <%= image_tag('title.gif', :size=> '400x100',
                  :alt => 'Problem Solving Engine') %>
    <%= render :partial => 'shared/menu_bar' %>
  </div>
  <div id='left'>
    <p class="notice"><%= notice %></p>
    <p class="alert"><%= alert %></p>
    <%= yield %>
  </div>
  <div id='right'>
    <%= render :partial => 'shared/right_bar' %>
  </div>
</div>
```

# Top Screen is now

---

as the following.

We have not prepared css files yet, so the right bar is located at the bottom.

---



# Now we arrange scaffold.css.scss

Located at app/assets/stylesheets

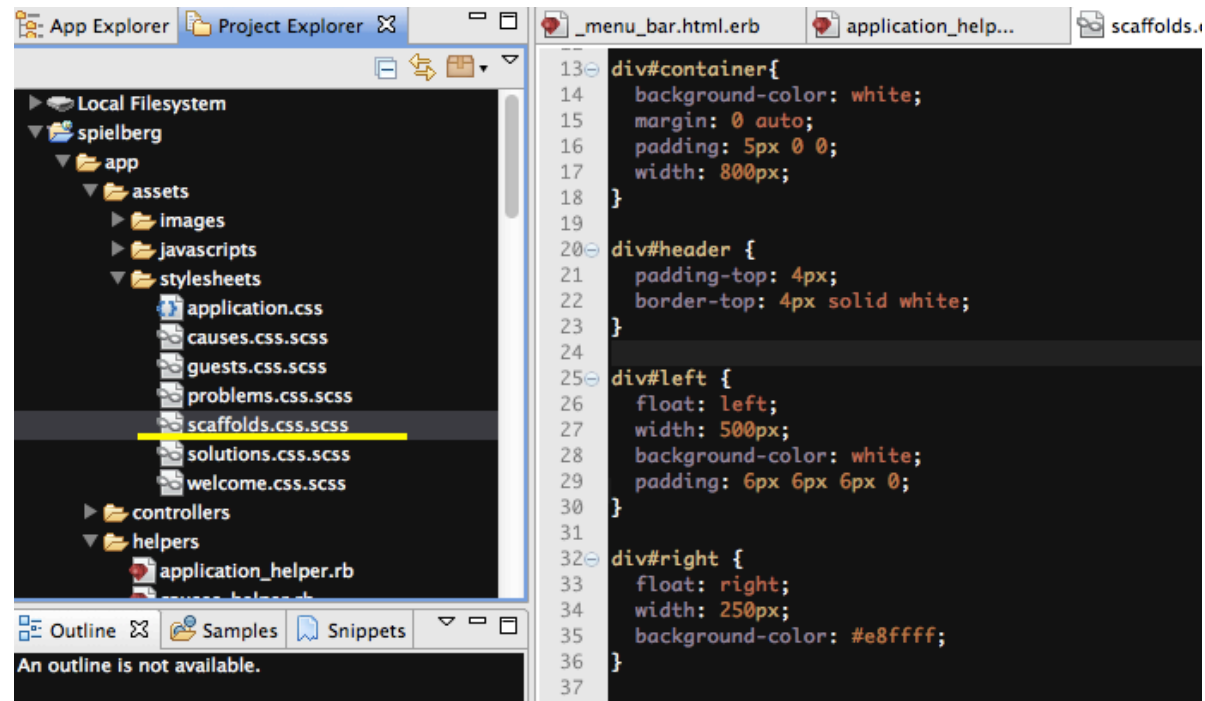
```
div#container{
  background-color: white;
  margin: 0 auto;
  padding: 5px 0 0;
  width: 800px;
}

div#header {
  padding-top: 4px;
  border-top: 4px solid white;
}

div#left {
  float: left;
  background-color: white;
  padding: 6px 6px 6px 0;
}

div#right {
  float: right;
  width: 228px;
  background-color: #e8ffff;
}

div#menu_bar{
  color: black;
}
```



# Whole scaffold.css.scss

---

```
body {
  background-color: white;
  color: #333;
  font-family: verdana, arial, helvetica, sans-serif;
  font-size: 13px;
  line-height: 18px; }
```

```
div#container{
  background-color: white;
  margin: 0 auto;
  padding: 5px 0 0;
  width: 800px;
}
```

```
div#header {
  padding-top: 4px;
  border-top: 4px solid white;
}
```

```
div#left {
  float: left;
  width: 500px;
  background-color: white;
  padding: 6px 6px 6px 0;
}
```

```
div#right {
  float: right;
  width: 250px;
  background-color: #e8ffff;
}
```

```
div#menu_bar{
  color: black;
}
```

```
p, ol, ul, td {
  font-family: verdana, arial, helvetica, sans-serif;
  font-size: 13px;
  line-height: 18px; }
```

```
pre {
  background-color: #eee;
  padding: 10px;
  font-size: 11px; }
```

```
a {
  color: #000;
  &:visited {
    color: #666; }
  &:hover {
    color: #fff;
    background-color: #000; } }
```

```
div {
  &.field, &.actions {
    margin-bottom: 10px; } }
```

```
#notice {
  color: green; }
```

```
.field_with_errors {
  padding: 2px;
  background-color: red;
  display: table; }
```

```
#error_explanation {
  width: 450px;
  border: 2px solid red;
  padding: 7px;
  padding-bottom: 0;
  margin-bottom: 20px;
  background-color: #f0f0f0;
  h2 {
    text-align: left;
    font-weight: bold;
    padding: 5px 5px 5px 15px;
    font-size: 12px;
    margin: -7px;
    margin-bottom: 0px;
    background-color: #c00;
    color: #fff; }
  ul li {
    font-size: 12px;
    list-style: square; } }
```

# And the screen is now

---

As the following.



## Add several lines for Log-Out

# Modify

```
(project)/app/views/shared/_menu_bar.html.erb
```

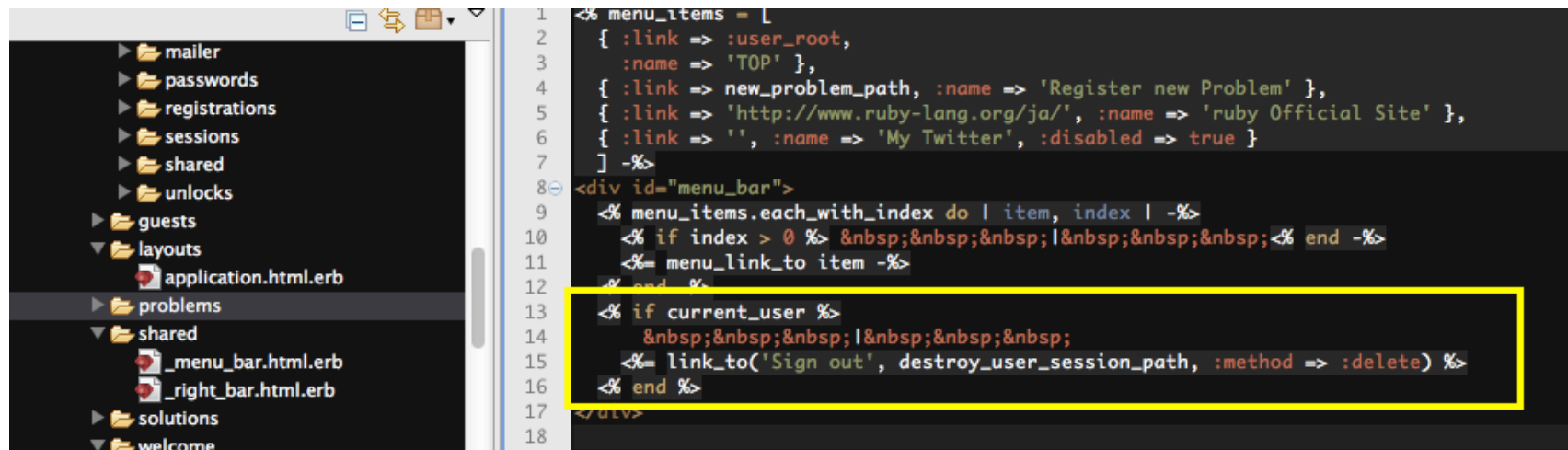
## Add following 4 lines;

<% if current user %>

      |

```
<%= link_to('Sign out', destroy_user_session_path, :method => :delete) %>
```

<% end %>





# Sign out link in Top Screen

---



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?	<a href="#">Show</a> <a href="#">Edit</a> <a href="#">Destroy</a>

[New Problem](#)

# Many Modification for today

---

Follow the instruction, finish programming up to today's screens, and then

- (1) In many screens, there are messages that the translation is missing for 'ja.'  
If your mother tongue is Japanese, add the translation.
- (2) Arrange the screen, to see the List of Problems.
- (3) If possible, design the top screen.

## Prepare for the Next Week

---

We proceed to get problem links screen, as we introduced this project concept. We finish introducing the relations, (one-to-many relation between problems and causes tables,) and introduce the 'problem specific screen,' to see many causes to one problem, and many solutions to one cause. By designing these relationships, clarify the table structures in the system.