Login/Authorization Notes

Nov 2016 – by Prof Bill

These are my notes on the login/authorization stuff in Odyssey built by CSC 495 in Spring 2016.

- odyssey495.herokuapp.com - the deploy
- github.com/noctrl-csc495-spring2016/Odyssey495 – the Github repo

There are four sections here: A. Session login and logout, B. Regular users, and C. Admin users, and D. Miscellany.

A. Session login and logout

Login and logout has three parts: 1) routes for login/logout, 2) login web page, 3) controller and helper code to perform login/logout, and 4) login guards for all the rest of the controllers in the system.

1. Routes

config/routes.rb – I don't think these routes are set in stone, link

```
root                                'sessions#new'

...

get    'login'                   => 'sessions#new'
post   'login'                   => 'sessions#create'
delete 'logout'                  => 'sessions#destroy'

get    'users/index'             => 'users#index'
get    'accounts/account'        => 'users#show'
```

2. Web page

There is only one session-related web page (embedded Ruby page, actually).

app/views/sessions/new.html.erb – login page HMTL, link

3. Code

The actual work of login/logout is done in the Sessions controller and helper.

app/controllers/sessions_controller.rb – sessions controller code with 3 methods: new, create, and destroy, link

app/helpers/sessions_helper.rb – sessions helper code... more work here than in the controller, actually, link

4. Guards

Other controllers in the system require that a user be logged in before accessing anything.

Here's one example:

app/controllers/days_controller.rb – Controller for Days object, link

```
class DaysController < ApplicationController
  include DaysHelper
  before_action :logged_in
  before_action :admin_or_standard
...
```

We will definitely need a guard for “logged_in”.

We probably won't need a guard like “admin_or_standard”. This was part of Odyssey because that system had a lower level of user called “Entry” whose access was restricted to a few pages.

B. Regular Users

Regular users. Not Admins.

1. The Account menu

The Account menu appears in the navbar/header. For a regular user, the Account menu has two choices: Settings and Logout. For admin users, an Admin choice is added.

app/views/layouts/_navbar.html.erb – this layout is the navbar in the header, link

```
<a class="dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-
haspopup="true" aria-expanded="false">
  Account<% if is_admin? %>s<% end %> <span class="caret"></span>
</a>
<ul class="dropdown-menu">
  <% if is_admin? %>
    <li role="presentation"><a href="/users">Admin</a></li>
  <% end %>
  <li><%= link_to "Settings", {:controller => "users", :action => "edit", :id
=> current_user.id}%></li>
  <li><%= link_to "Log out", logout_path, method: "delete" %></li>
</ul>
```

Notice: If user is an admin, then 1) Account is Accounts (attention to detail, ha!) and 2) Admin is added to the dropdown.

2. Logging out

This is covered in the sessions code on page 1. In the navbar layout above, see that “call” to the delete method in the navbar code in the logout dropdown?

3. User Settings

In the navbar layout (again), you can see the “call” to the edit method for User.

app/controllers/users_controller.rb – User controller code, link

```
class UsersController < ApplicationController
  before_action :logged_in
  before_action :user_exists, except: [:index, :new, :create, :prune]
  before_action :is_admin, except: [:update, :edit]
  before_action :is_super, only: [:prune]
```

This controller has multiple guards. Disregard “prune”. It's a special power of the Super Admin to prune old things from the database.

This is probably the most complex code we'll encounter. Most of it deals with Admins and pages to add/remove/view users. The commenting is pretty good though, so that helps.

Here's the view page for Account/Settings:

app/views/users/edit.html.erb – The view/HTML for user settings, changing password, link

C. Admin users

This is stuff for Admins. And the ONE Super Admin.

Note – I like the Jason K idea. We know there will always be one and only one user, so make the User with database ID of #1 the Super Admin. This means you don't need a boolean in the data model, and you change the is_super() method to check the ID.

1. Views, web pages

View all users, add a new user, and edit/show user

app/views/users/index.html.erb – this is the view for Account/Admin, the page lists all users for an Admin, link

app/views/users/new.html.erb – page to create a new user, mostly a form, link

app/views/users/show.html.erb – show a user to edit him/her, change password, and there's a remove link down at the bottom, link

I think that last “show” is a misnomer because “edit” was already taken for non-Admin users to change their password.

2. Controller and Helper

I've already covered the User controller code. There's a lot of Admin stuff in there to support the pages above. And here, for completeness... the User helper code

> app/helpers/users_helper.rb – very simple helper methods, link

D. Miscellany

Seed the system with one user, the Super Admin!

> db/seeds.rb – username=profbill, password=password, email=wtkrieger@noctrl.edu, link

For dev purposes, load up some users.

> test/fixtures/users.yml – add a user for each class member, password=password, link

I won't go in-depth on testing. There's 200 lines of testing code here.

> test/controllers/users_controller_test.rb – some good ones here, link

If you're coding this up, you can do i tin the order of the sections in this document. Each is fairly separate and can be done independently.

1. Login and logout,
2. User settings (change password), and finally
3. Admin pages

That's it. I think.

thanks... yow, bill