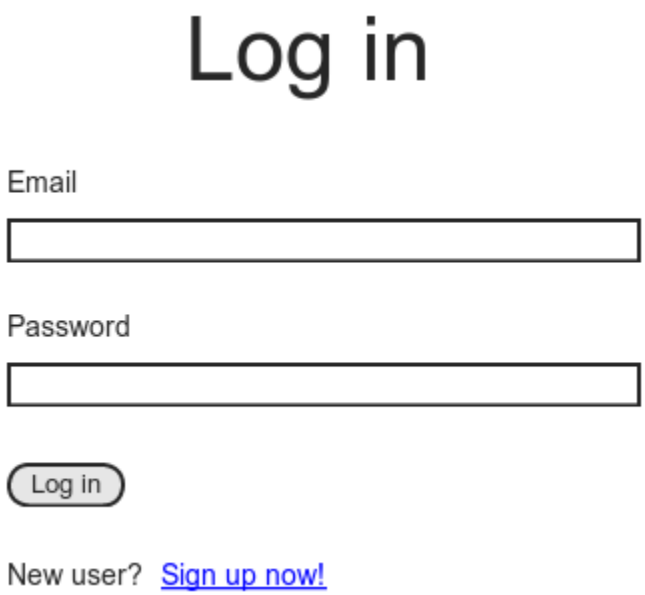
CS 416
Web Programming

Ruby on RAILS
Chapter 8-9 User login, Ajax search

Dr. Williams
Central Connecticut State University

A look ahead

- First look session management & authentication



- Adding Ajax search of users
- Getting started

```
git checkout -b login
```

Create sessions controller

- Generate Sessions controller and *new* view

```
rails generate controller Sessions
new
```

- Add RESTful routes for sessions

```
get '/login',          to: 'sessions#new'
post '/login',         to: 'sessions#create'
delete '/logout',      to: 'sessions#destroy‘
```

Modify header partial for login_path

Login form

```
<% provide(:title, "Log in") %>
<h1>Log in</h1>

<div class="row">
  <div class="col-md-6 col-md-offset-3">
    <%= form_for(:session, url: login_path) do |f| %>

      <%= f.label :email %>
      <%= f.email_field :email, class: 'form-control' %>

      <%= f.label :password %>
      <%= f.password_field :password, class: 'form-control' %>

      <%= f.submit "Log in", class: "btn btn-primary" %>
    <% end %>

    <p>New user? <%= link_to "Sign up now!", signup_path %></p>
  </div>
</div>
```

Finding and authenticating user

```
def new
 end

 def create
   user = User.find_by(email: params[:session][:email].downcase)
   if user && user.authenticate(params[:session][:password])
     # Log the user in and redirect to the user's show page.
   else
     flash.now[:danger] = 'Invalid email/password combination'
     render 'new'    end
 end

 def destroy
 end
```

Including session functionality across the site

- Modify Application controller to include Sessions Helper

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
  include SessionsHelper
end
```

- Create common log in

```
module SessionsHelper

  # Logs in the given user.
  def log_in(user)
    session[:user_id] = user.id
  end
end
```

Completing login

```
if user && user.authenticate(params[:session][:password])
  log_in user
  redirect_to user
else
  ...
```

Completing sessions helper – get current user and check if logged in

```
# Returns the current logged-in user (if any).
def current_user
  @current_user ||= User.find_by(id: session[:user_id])
end

# Returns true if the user is logged in, false otherwise.
def logged_in?
  !current_user.nil?
end
```

Creating dynamic header

```
<% if logged_in? %>
  <li><%= link_to "Users", users_path %></li>
  <li class="dropdown">
    <a href="#" class="dropdown-toggle"
                data-toggle="dropdown">
      Account <b class="caret"></b>
    </a>
    <ul class="dropdown-menu">
      <li><%= link_to "Profile", current_user %></li>
      <li><%= link_to "Settings", '#' %></li>
      <li class="divider"></li>
      <li>
        <%= link_to "Log out", logout_path,
                    method: "delete" %>
      </li>
    </ul>
  </li>
<% else %>
  <li><%= link_to "Log in", login_path %></li>
<% end %>
```

Login upon signup

- Add call to login helper method in create

```
if @user.save
  log_in @user
  flash[:success] = "Welcome to the Sample App!"
  redirect_to @user
else
  render 'new'
end
```

Adding logout

- Add helper method

```
# Logs out the current user.
def log_out
  session.delete(:user_id)
  @current_user = nil
end
```

- Use in session controller

```
def destroy
  log_out
  redirect_to root_url
end
```

Adding search

- Existing RESTful Route to list users:

```
users GET    /users(.:format)    users#index
```

- Add the functionality so the page lists all users, but filters down users by Ajax

- Start by completing index path
  - Add index method to controller

```
def index
  @users = User.all
end
```

Create views

- Basic layout is placeholder for table with results to be populated

- Simple index view with partial for results

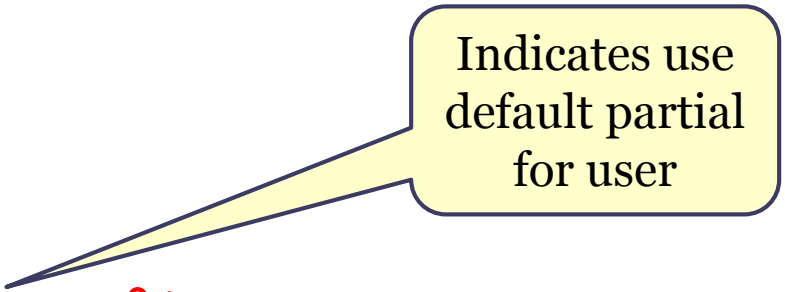
```
<h1>User search</h1>
<div id="results">
  <%= render 'results' %>
</div>
```

Results partial

- In *_results.html.erb*

```
<table width="100%" border="1">
  <thead>
    <tr>
      <th width="25%">Name</th>
      <th width="50%">Email</th>
      <th width="25%"></th>
    </tr>
  </thead>

  <tbody>
    <%= render @users %>
  </tbody>
</table>
```



User partial

- In *_user.html.erb*

```
<tr>
    <td><%= user.name %></td>
    <td><%= user.email %></td>
    <td><%= link_to 'Show', user %></td>
</tr>
```

- Now if specify to show a user in a page and don't explicitly specify format this will be used as default display

Adding search

- Add route for where to post our search query

```
post '/search',  to: 'users#search‘
```

- Add collection of search terms to index page:

```
<%= form_tag(search_path, id: "search_form") do %>
  <%= text_field_tag :search, params[:search] %>
  <%= submit_tag "Search", name: nil %>
<% end %>
```

Add search path

- Controller method

```
def search
  name = params[:search] + '%'
  @users = User.where(['name LIKE ?', name])
end
```

- Add simple view search.html.erb

```
<div id="results">
  <%= render 'results' %>
</div>
```

Now working form post search results…

Making search Ajax

- Modify index to make the form *remote*

```
<%= form_tag(search_path, remote: true, id:
"search_form") do %>
```

- Now add to controller how to respond to each format:

```
def search
    name = params[:search] + '%'
    @users = User.where(['name LIKE ?', name])
    respond_to do |format|
      format.html
      format.js
    end
  end
```

Ajax response

- Format of response says which view to render, JS response says render ***search.js.erb*** rather than ***search.html.erb***

- In *search.js.erb* have Jquery insert output of results partial into results div tag

```
$("#results").html(
"<%= escape_javascript(render("results")) %>");
```

Presto! Ajax results

Results as you type

- To make the results filter as we type rather than submit add JS to all search pages..

- In application.js

```
$(document).on('turbolinks:load',function() {
    $('#search').on('keyup', function() {
        $('#search_form').submit();
    });
});
```