---

add interface to create admin users
admin users is kept in user table

$ rails g scaffold User name password_digest

[digest type is for BCyrpt password]

$ rails db:migrate

modify user model to add validates:

change it to:

```
class User < ActiveRecord::Base
  validates :name, presence: true, uniqueness: true
  has_secure_password  #so that we have user.authenticate method
end
```

has_secure_password: Adds methods to set and authenticate against a BCrypt password.

in Gemfile uncomment out this:
[so that has_secure_password and be used]

```
# Use ActiveModel has_secure_password
gem 'bcrypt', '~> 3.1.7'
```

install this gem:

> bundle install (or just bundle. default to bundle install)

modify the following actions:
users#create
users#update
users#index

users_controller.rb

redirect to users list, instead of showing a user

```
  # POST /users.json
```

```
  def create
    @user = User.new(user_params)

    respond_to do |format|
      if @user.save
    format.html { redirect_to users_url, notice: “User
#{@user.name} was successfully created.” }
        format.json { render :show, status: :created, location:
@user }
      else
        format.html { render :new }
        format.json { render json: @user.errors,
status: :unprocessable_entity }
      end
    end
  end
```

—

```
  # PATCH/PUT /users/1
  # PATCH/PUT /users/1.json
  def update
    respond_to do |format|
      if @user.update(user_params)
        format.html { redirect_to users_url, notice: “User
#{@user.name} was successfully updated.” }
        format.json { render :show, status: :ok, location: @user }
      else
        format.html { render :edit }
        format.json { render json: @user.errors,
status: :unprocessable_entity }
      end
    end
  end
```

```
  def index
    #@users = User.all
    @users = User.order(:name)
  end
```

change
app/views/users/index.html.erb

```
<h1>Listing Users</h1>
```

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

change the app/views/users/_form.html.erb to add legend

```
<div class='order_form'>

<%= form_for(@user) do |f| %>
  <% if @user.errors.any? %>
    <div id="error_explanation">
      <h2><%= pluralize(@user.errors.count, "error") %> prohibited
this user from being saved:</h2>

      <ul>
      <% @user.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
      </ul>
    </div>
  <% end %>

<fieldset>
<legend>Enter user details</legend>
  <div class="field">
    <%= f.label :name %><br>
    <%= f.text_field :name %>
  </div>
  <div class="field">
    <%= f.label :password %><br>
    <%= f.password_field :password %>
  </div>
  <div class="field">
    <%= f.label :password_confirmation %><br>
    <%= f.password_field :password_confirmation %>
  </div>
  <div class="actions">
    <%= f.submit %>
  </div>
</fieldset>
          <% end %>
```


go to /users in browser

if you get cannot load bcript error
restart your server

create new user using the interface.

---

create actions for admin to check number of orders:

$ **rails g controller admin index**

just one page to list order number

change the view to display number of orders:
app/views/admin/index.html.erb

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

<h1>Welcome</h1>

It's <%= Time.now %>
We have <%= pluralize(@total_orders, "order") %>.
```

change controller
app/controllers/admin_controller.rb

```
class AdminController < ApplicationController
  def index
    @total_orders = Order.count
  end
end
```

create actions to authenticate user (logon/logout)

```
$ rails g controller access new create destroy
```

```
access#new:  for admin user logon
access#create: for authenticate admin user
access#destroy: for admin user logout
```

change view for user logon

app/views/access/new.html.erb

```
<div class="order_form">
  <% if flash[:alert] %>
    <p id="notice"><%= flash[:alert] %></p>
  <% end %>
  <%= form_tag do %>
    <fieldset>
      <legend>Please Log In</legend>
      <div>
        <%= label_tag :name, 'Name:' %>
        <%= text_field_tag :name, params[:name] %>
      </div>
      <div>
     <%= label_tag :password, 'Password:' %>
        <%= password_field_tag :password, params[:password] %>
      </div>
      <div>
        <%= submit_tag "Login" %>
      </div>
    </fieldset>
  <% end %>
</div>
```

change actions in default methods

```
class AccessController < ApplicationController
  def new

   def new
    if session[:user_id]
      redirect_to admin_url, notice: "already logged on"
      return
    end
  end
```

```
#this is actually “post ‘/logon’
  def create
    user = User.find_by(name: params[:name])
    if user and user.authenticate(params[:password])
      session[:user_id] = user.id
      redirect_to admin_url
    else
      redirect_to login_url, alert: "Invalid user/password combination"
    end
  end


  def destroy
    session[:user_id] = nil
    redirect_to shopper_url, notice: "Logged out"
  end

end
```

change route so that “admin” point to “admin#index” action
“login” point to “access#new” action
also add route point to “access#create” and “access#destroy”

add logout button to side bar (and other links)

```
<body class='<%= controller.controller_name %>'>


  <div id="banner">
    <%= image_tag("logo.png") %>
    <%= @page_title || "Our Products" %>
  </div>
  <div id="columns">
    <div id="side">
<div id = 'cart'>
<%= render @cart %>
</div>
      <ul>
        <li><a href="http://www....">Home</a></li>
    <li><a href="http://www..../faq">Questions</a></li>
    <li><a href="http://www..../news">News</a></li>
    <li><a href="http://www..../contact">Contact</a></li>
      </ul>

       <% if session[:user_id] %>
    <ul>
          <li><%= link_to 'Orders',   orders_path   %></li>
          <li><%= link_to 'Products', products_path %></li>
```

```
          <li><%= link_to 'Users',    users_path    %></li>
        </ul>
    <%= button_to 'Logout', logout_path, method: :delete   %>
      <% end %>


    </div>
    <div id="main">

      <%= yield %>

    </div>
  </div>


</body>
```

set access

change application controller so that an authorize action will be taken first

application_controller.rb

```
class ApplicationController < ActionController::Base

  before_action :authorize

  # Prevent CSRF attacks by raising an exception.
  # For APIs, you may want to use :null_session instead.
  protect_from_forgery with: :exception

 def authorize
    unless User.find_by(id: session[:user_id])
      redirect_to login_url, notice: "Please log in"
    end
  end
end
```

whitelist some controllers to skip authorize

add
skip_before_action :authorize

to the controllers you don’t want authorize action to be called