

**Ruby and Sinatra**

Worksheet 5: User Authentication



By Dr N Beacham

August 2019

© 2019 University of Aberdeen

In this exercise we show you how to implement an authentication feature for a database-driven website in an application and how to use Sinatra views to check and prevent users accessing parts of an application website.

**Getting started**

Make a copy of the wiki folder called wikibackup4.

Now when you make changes to the files in the original wiki folder, it does not matter if things go wrong. You can always return to the backup version.

**Update program to read file contents**

Open the original *wikiapp.rb* file and make the following changes to the Ruby code.

# **Handling links that access pages that do not exist**

First create a new **notfound.erb** view file within the **views** folder with the following code.

*<p>This page does not exist!</p>*

*<p>Please notify the administrator.</p>*

Save the view file **notfound.erb** within the **views** folder.

Add the following code to the **wadapp.rb** file (just above the **not\_found** Ruby code block) to call the **notfound.erb** view.

*get '/notfound' do*

*erb :notfound*

*end*

Change the code in **wadapp.rb**file to the following.

*not\_found do*

*status 404*

*redirect '/notfound'*

*end*

Note: the **status 404** code is to be retained indicating that the client browser was able to communicate with the WEBrick server, but could not find what was requested.

# **Handle instances where no account exists**

Create a new **noaccount.erb** view file within the **views** folder with the following code.

*<p>This account does not exist!</p>*

Save the view **noaccount.erb** within the **views** folder. Next add the following code to the **wadapp.rb** file to call the **noaccount.erb** view.

get '/noaccount' do

erb :noaccount

end

# **Handle instances where access is to be denied**

Create a new **denied.erb** view file within the **views** folder with the following code.

*<h2> Denied! </h2>*

*<p>You are not authorised to view this page.</p>*

*<p>Please log into an admin privileged account to edit, create, or delete wiki pages.</p>*

Save the view as **denied.erb** within the **views** folder. Add the following code to the **wadapp.rb** file to call the **denied.erb** view.

*get '/denied' do*

*erb :denied*

*end*

# **Adding an admin control feature**

Create a new view file **admincontrols.erb** within the **views** folder with the following code.

*<h2>Admin controls page</h2>*

*<p><b>User list</b></p><br>*

*<p>Username - DateJoined - Edit?</p><br>*

*<% @list2.each do |liste| %>*

*<article>*

*<p>*

*<%= liste.username %> - <%= liste.created\_at %> - <%= liste.edit %>*

*<% if liste.username != "Admin" %>*

*<form action="/user/<%= liste.username %>" method="post" id="edit">*

*<input type="hidden" name="\_method" value="put">*

*<input type="checkbox" name="edit" <%= "checked" if liste.edit %>>*

*<input type="submit" value="Allow Edit?">*

*</form>*

*<% end %>*

*<% if liste.username != "Admin" %>*

*<a href="/user/delete/<%= liste.username %>">Delete User?</a>*

*<% end %>*

*</p>*

*</article>*

*<% end %>*

Save the view **admincontrols.erb** within the **views** folder. Then add the following code to the **wadapp.rb** file (just above the **not\_found** Ruby code block) to call the **admincontrols.erb** view.

*get '/admincontrols' do*

*# protected!*

*@list2 = User.all.sort\_by { |u| [u.id] }*

*erb :admincontrols*

*end*

# **Adding an edit control functionality to the admin feature**

Add the following code to the **wadapp.rb** file (just above the **not\_found** Ruby code block) to allow the administrator to change the editing rights of a user within the database.

*put '/user/:uzer' do*

*n = User.where(:username => params[:uzer]).to\_a.first*

*n.edit = params[:edit] ? 1 : 0*

*n.save*

*redirect '/'*

*end*

# **Adding a delete user functionality to the admin feature**

Add the following code to the **wadapp.rb** file (just above the **not\_found** Ruby code block) to check that the user is logged in as Admin and if so remove the selected user from the database.

*get '/user/delete/:uzer' do*

*# protected!*

*n = User.where(:username => params[:uzer]).to\_a.first*

*if n.username == "Admin"*

*erb :denied*

*else*

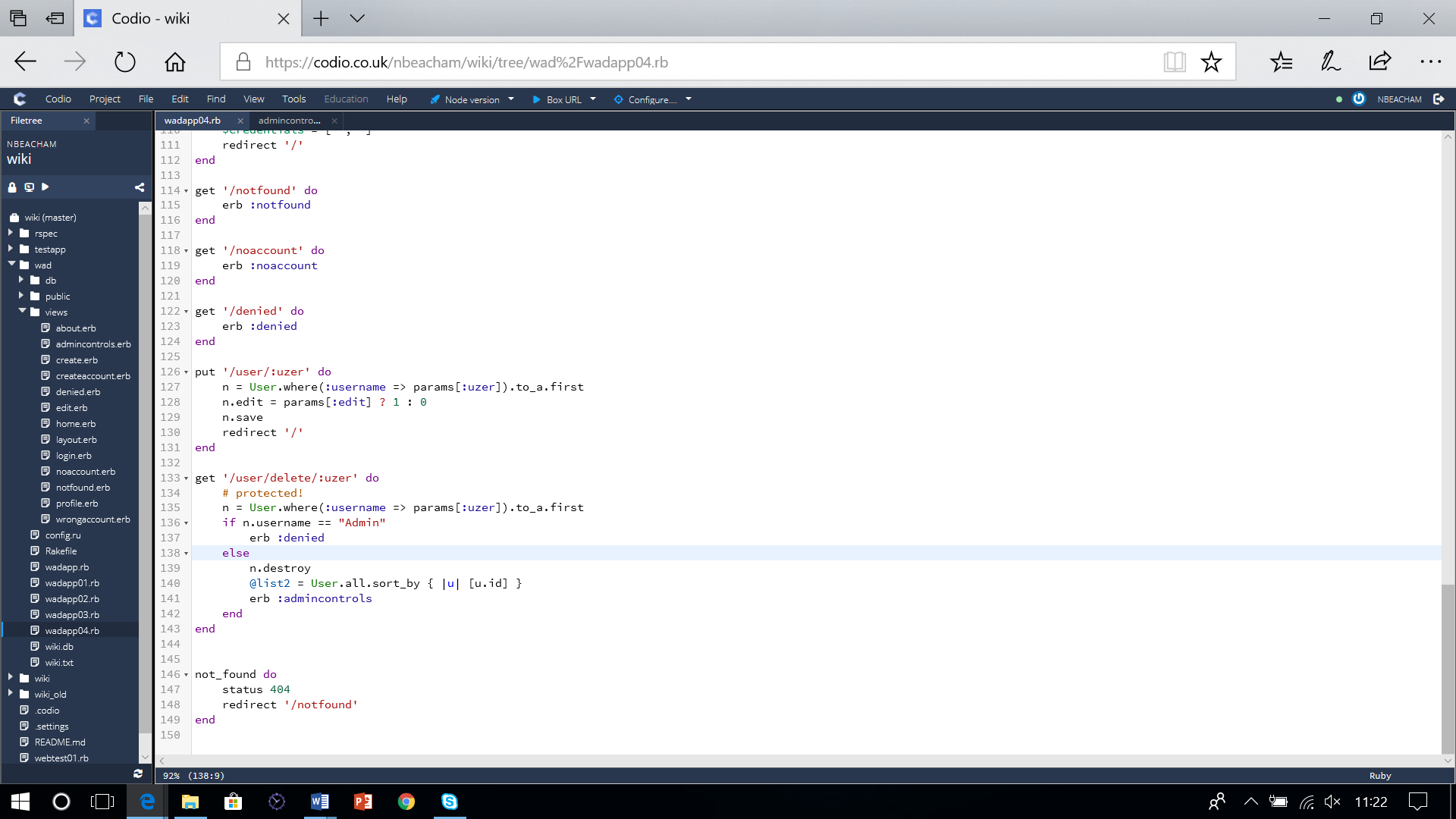
*n.destroy*

*@list2 = User.all.sort\_by { |u| [u.id] }*

*erb :admincontrols*

*end*

*end*



# **Save and run the application**

Save the file as**wadapp.rb**and run the application. You will see that the application now allows you to create new user accounts, log in and out, and display a user’s profile. Under the admin controls link you can also allow registered users to become editors.

Note: there is however one fundamental flaw. If you enter the following urls you can access features within the application without being logged in or registered.

*/admincontrols*

*/edit*

*/user/delete/…*

We therefore need to enter a helper to protect aspects of the application unauthorised by users.

# **Adding Helper for protection**

Add the following Helper code to the **wadapp.rb** file (just above the **$myinfo** global variable).

*helpers do*

*def protected!*

*if authorized?*

*return*

*end*

*redirect '/denied'*

*end*

*def authorized?*

*if $credentials != nil*

*@Userz = User.where(:username => $credentials[0]).to\_a.first*

*if @Userz*

*if @Userz.edit == true*

*return true*

*else*

*return false*

*end*

*else*

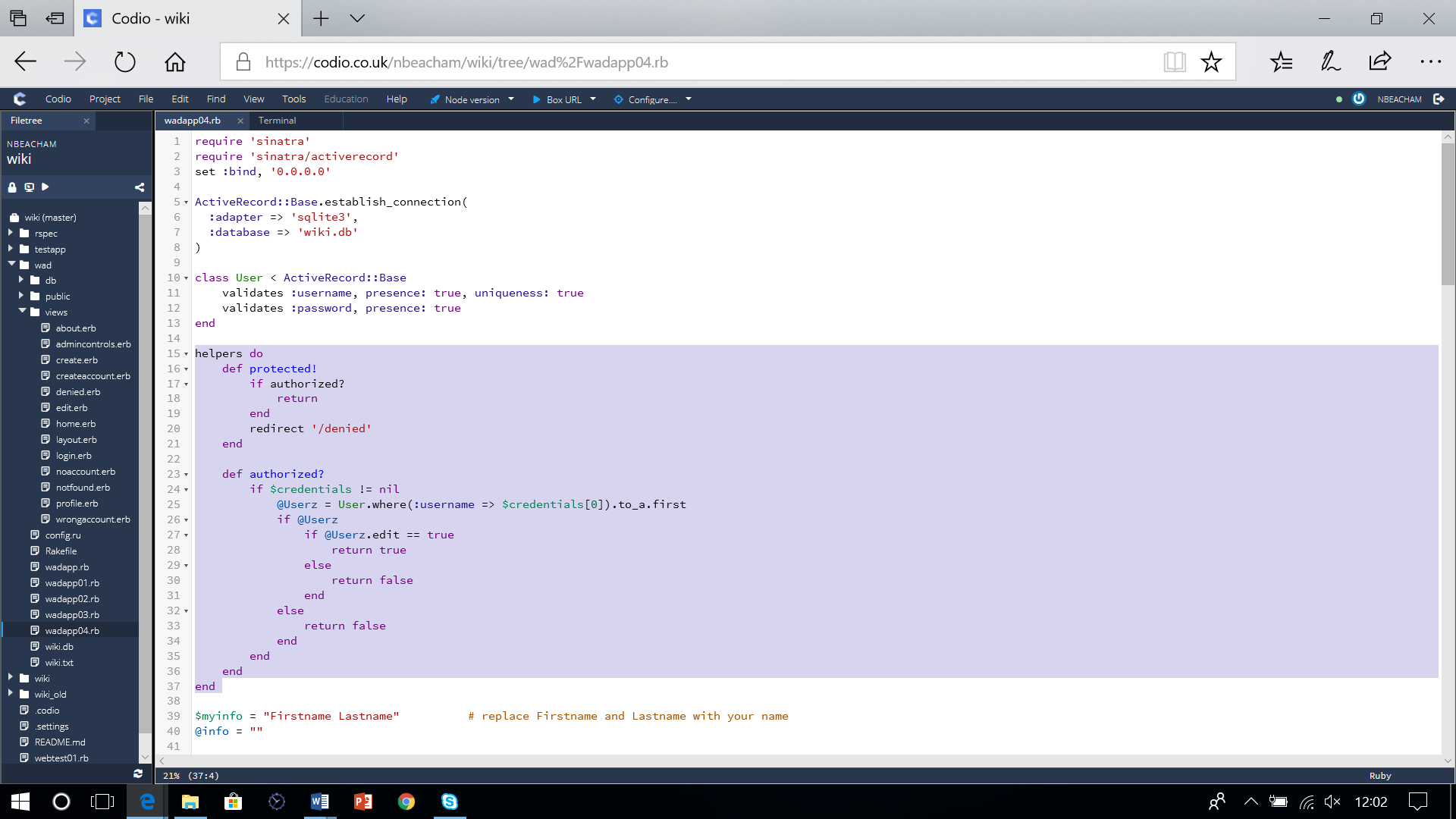
*return false*

*end*

*end*

*end*

*end*



Note: it is important to locate the code towards the top of the code file so it is defined and available to methods below it can call it.

We can now add a protection call to those features where users are not allowed to access unless logged in. Add the code **protected!** to the **admincontrols** routine as shown below by removing the comment symbol ‘**#**’.

*get '/admincontrols' do*

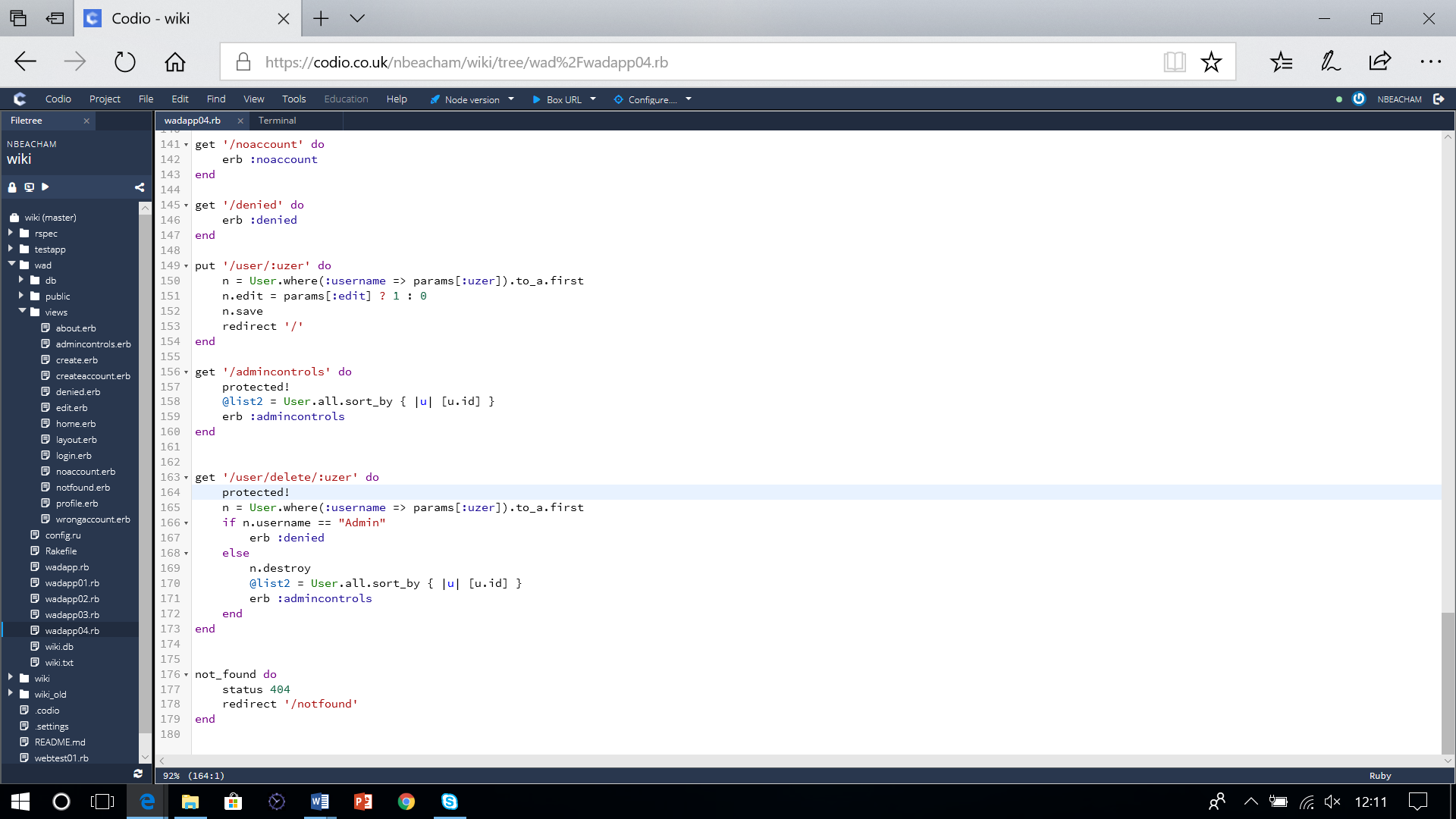
*protected!*

*@list2 = User.all.sort\_by { |u| [u.id] }*

*erb :admincontrols*

*end*

Add the **protected!** code to other routines to help to protect the authentication of the application by removing the comment symbol ‘**#**’ from the admin feature which deletes user accounts.



Great. You will see that the **wadapp.rb** application now allows only the administrator to view user accounts, change user editing rights and delete user accounts.

**Adding a profile feature**

The final part to add is the edit profile feature. At present when a user is logged in selecting the user profile link displays “Page does not exist!”. To allow a user, who has been given editing status, to make changes to the application the following changes need to be made.

Note: editing status for users is set to false by default. The Administrator is the only user who can give the user editing status. Once a user has signed in and created an account, the administrator is then required to login and change the edit status.

Create a new **profile.erb** view file within the **views** folder with the following code.

*<h2>User</h2>*

*<% if @Userz %>*

*<article>*

*<p>Account - <%= @Userz.username %></p>*

*<p>Created - <%= @Userz.created\_at %></p>*

*<p>Last updated - <%= @Userz.updated\_at %></p>*

*<p>Edit status - <%= @Userz.edit %></p>*

*</article>*

*<% end %>*

Save the view **profile.erb** within the **views** folder. Add the following code to the **wadapp.rb** file to call the **denied.erb** view.

*get '/user/:uzer' do*

*protected!*

*@Userz = User.where(:username => params[:uzer]).to\_a.first*

*if @Userz != nil*

*erb :profile*

*else*

*redirect '/denied'*

*end*

*end*

Finally, when you launch the application you should now have the beginnings of a template which you can go on to develop further by adding your own features and web pages.

**Closing the application**

Before exiting, please ensure you have saved and closed all files, and terminated any running applications.