CS-321: Server-Side Web Development
Spring 2021
Northeastern Illinois University
Homework #10: Security
Due Date: Thursday, 04/22/21 by 6:00 p.m. CDT

IMPORTANT: Every step in this homework assignment refers to materials that was presented in the lecture videos and PowerPoints, so you should be actively using the videos, PowerPoints, and referenced materials.

Homework Goals:

1. Enable security in Spring Boot.
2. Set up users using database-backed authentication.
3. Create login and registration functionality.

Part I: Enabling security

- ☐ Include the Spring Boot security dependency in your project.
- ☐ Following Lecture 10.1, create security configuration class. This should be placed in the appropriate package in your application.
- ☐ Override the `configure` methods to create an in-memory user and to set up access rules for the routes in your application. Who should be allowed to see each page in your application. Do you have an admin role? A user role? Both? If you have both, make sure that you have an in-memory user for each role to test access to your pages.
- ☐ Test logging in using the default login page provided by Spring Security.
- ☐ Make sure all your code compiles and that your application runs as before. Push to GitHub.

Part II: Custom Login/Logout

- ☐ Create a custom login page - this should be consistent with your website theme. Make sure to set up the necessary login form rules in the `configure` method in the security configuration class for where the user should be redirected to after logging in. Decide how you want your users to log in - via email or via username?
- ☐ Handle all errors in the view for logging in.
- ☐ Create logout functionality so that you are easily able to test logging in and logging out. This should be placed in a location that is intuitive for users to access. Make sure to set up the necessary logout form rules in the `configure` method in the security configuration class for where the user should be redirected to after logging in.

- ☐ Push your code to GitHub.

Part III: Restricting View Elements

- ☐ Following the examples in Lecture 10.2, install the Spring Security Dialect dependency.
- ☐ Decide what navigation elements users should be able to see depending on their role and whether they are logged in or not. For example, an anonymous user should not be able to see navigation for editing, viewing things that belong to a particular user, etc.
- ☐ Test this functionality and push your code to GitHub.

Part IV: Database-Backed Authentication

- ☐ Following the example in Lecture 11, create (or modify your existing) User class. The User class should implement `UserDetails`. Think carefully about what users would want to store about themselves in your application.
- ☐ You should have a "roles" component even if you only have one role so that you are able to grant authorities (i.e. privileges).
- ☐ Make sure to indicate which columns (i.e. instance variables) are unique.
- ☐ Create the associated repository.
- ☐ Verify that your code runs and that the tables are created in the database.
- ☐ Create the `UserDetailsService` implementation.
- ☐ Modify the security configuration class to work with your database-backed user and the `UserDetailsService` implementation.
- ☐ Create a data loader class that implements the `CommandLineRunner` interface and loads several users into your database. Remember that this runs regardless of whether your yml file is set to update or create, so you should comment out the code after loading it the first time!
- ☐ Test that you can log in with your pre-loaded DB-backed users.
- ☐ Push your code to GitHub.

Part VI: Registration

- ☐ Create a registration form for new users. Note that if you have multiple roles, this should be the registration form for generic users (not admins) - your admin user will stay pre-loaded.
- ☐ Either create a new controller for your registration form, or use your HomeController.
- ☐ You will need to add validation to your user model.

- ☐ Make sure to handle all errors in using both server-side validation and client-side validation (CSS), as well as handling database uniqueness errors.
- ☐ Bootstrap is required for all pages - we don't want them looking like they're from the 90's!
- ☐ Test that you can create a new user using the registration form and then log in with that user.
- ☐ Push your code to GitHub.

TO-DO: Things that are not in this last homework but that are still required for the Final Project

- Hooking up your User to your other models. Your user should be able to view/edit/delete their own items, but they should not be able to edit or delete other users' items. This involves setting up a DB relationship between your other models and your user model.
- User account functionality. A user should be able to change/modify aspects of their account. Validation is required and error messages.
- Error pages.
- Review the Final Project requirements! These are posted on the course website.

Submitting your code to D2L:

1. Zip up the project folder and submit to the D2L assignment folder. I look at your code in GitHub and also use D2L as a way to know when your GitHub repo is ready to be reviewed.
2. As a comment in D2L (there's a place for students to comment when they upload files), please provide the database name, the username, and the password for your database. Basically, copy your environment variables and paste them as a comment in D2L.