CS-321: Server-Side Web Development
Spring 2021
Northeastern Illinois University
Homework #7: Persistence, Part I
Due Date: Saturday, 03/13/21 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 both the videos, PowerPoints, and referenced materials.

Homework Goals:

1. Create a remote MySQL database.
2. Create an ORM for your domain models.
3. Modify the controllers to save to the database and retrieve from the database.
4. Handle validation for your form.
5. Maintain consistent styling for all pages in your website.

Part I: Creating your database

- ☐ Go to www.freesqldatabase.com and create an account.
- ☐ Create your free database - remember that once the database is live, you will receive an email with the connection and password details.

Part II: ORM

- ☐ **READ ME!!**: If your current domain model is some type of a "User" (i.e someone who will be signing in to your application), you will need to create a new domain model - something else that will be represented in your code (like a posting, a note, etc - i.e. something that your user will interact with). You do not need to delete the "User" model, but you cannot use it for this homework. You will additionally need to update your form to work with this new model (as we will have different forms as part of working with Spring Security). Remember that all domain models go in the `models` package.
- ☐ Add the dependencies for Spring Data JPA and MySQL connections to the pom file.
- ☐ Turn your non-user model into an entity.
- ☐ Add getters and setters for all the private fields, a default constructor, and an overloaded constructor.
- ☐ Add a field for the database ID and two fields that track the date/time that the entity was created and modified (and the associated methods needed to add those values in the database with the correct annotations).

- ☐ Create a package (inside your base package) named `data` and inside this package, create an interface that extends the `CRUDRepository` interface.
- ☐ Push your code to GitHub (don't run it yet!).

Part III: DB Configuration

- ☐ Following the lecture example, include the appropriate configuration in the `application.yml` file to connect to the database.
- ☐ Your database name, username, and password configurations should be stored as environment variables in IntelliJ (so that they are not pushed to GitHub).
- ☐ Run your code to verify that you do not have any errors in your terminal upon starting the application (i.e. no database connection issues).
- ☐ Push your code to GitHub.

Part IV: Saving to the DB

- ☐ Modify the controller that has the `PostMapping` method for your form so that it has an instance variable of the type of your repository interface.
- ☐ Create a constructor with the `@Autowired` annotation that takes the type of your repository as a parameter and sets the instance variable (this is DI!!).
- ☐ Modify your `PostMapping` method to remove the `RedirectAttributes` parameter and to save to your database using the repository methods.
- ☐ Modify the controller that handles the redirect/display view to remove the flash attribute from the `GetMapping` method parameter. The method should just return the view name. Modify the associated view to just have the navigation at this point (it won't display anything yet).
- ☐ Run your application and verify that you can save to your database (using the phpMyAdmin functionality provided by freesqldatabase.com
- ☐ Push your code to GitHub.

Part V: Validation

- ☐ Add the dependency for Spring Boot validation to the pom file and load the Maven changes.
- ☐ Modify your domain model so that the fields that are saved in the database have appropriate validation annotations and messages. Messages should have proper grammar, spelling, and capitalization. You should not add validation annotations to the created/modified date/time fields.
- ☐ Modify the `PostMapping` method for your form so that it can handle validation.

- ☐ Modify the associated form so that errors are displayed. The errors should be appropriately styled using Bootstrap. The form controls should not substantially shift to the left or right when displaying error messages.
- ☐ Push your code to GitHub.

Part VI: Reading from the DB

- ☐ Modify the `GetMapping` method for the redirect/view/display controller to retrieve all the items from the database and add it to the model.
- ☐ Modify the associated view to display all the items, styled nicely using Bootstrap.
- ☐ Run your application and verify that you can retrieve from your database.
- ☐ Push your code to GitHub.

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.