**Assignment – 5**

|  |  |
| --- | --- |
| **Student Name/ID Number:** | Wildan Luqmanul Hakim/BDSE04-0322 |
| **Academic Year:** | 2022 |
| **Unit Assessor:** | Arvinder Kaur |
| **Project Title:** | Assignment 5 - Spring Security, Error Handling, and Logging |
| **Issue Date:** | 30 November 2022 |
| **Submission Date:** | 8 December 2022 |
| **Internal Verifier Name:** |  |
| **Date:** | 8 December 2022 |

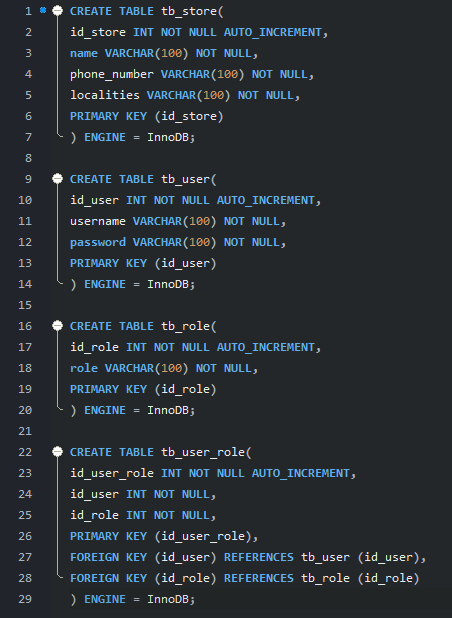
|  |
| --- |
| **Learner declaration** |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.    Student signature: Date: 8 December 2022 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Module No:** | 1 | **IU No:** | 5 | **Exercise No.** | 1 |

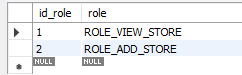
|  |  |
| --- | --- |
| **Lab Assessment Statement** | **Assignment 5 - Spring Security, Error Handling, and Logging**  This assignment is about implementing Authentication & Authorization solution for Know-Your-Neighborhood application.  **Spring Security**   1. Create user, role and user\_role tables in the database.    1. [Note: The words user & role are reserved in some databases.. Use some other name, if so.] 2. Add entries into user, role and user\_role tables.    1. Add two entries into role table. One role is to view the store info (VIEW\_STORE) and the other role is to add/modify store info (ADD\_STORE).    2. Add two entries into user table. Let one user have play VIEW\_STORE role, another user play only both VIEW\_STORE & ADD\_STORE roles. 3. Enhance Spring Data JPA enabled version of ‘Know-Your-Neighborhood’ application to support authentication and authorization. 4. Authentication:    1. Create entities for User and Role.    2. Create a login page for the application. User should be forced to login before visiting any of the application page.    3. Authenticate user with the credentials entered against the database entries added in the previous step.    4. Show a message if user enters invalid user and password. And then allow him to re-enter the credentials. 5. Authorization:    1. Once the user loggedin, make sure that only the user with ADD\_STORE role can add/modify store data. The user with VIEW\_STORE data can only view the stores entered.    2. If there are no stores in the system, show appropriate message to the user. 6. Add CSRF security to the application   **Logging**   1. Develop 5 unit test cases to validate the role based authorization   **Logging**   1. Implement the rolling file appender and log at the class level   **Error Handling**   1. Create Global Exception Handler that can handle an exception. Add method to handle run away exception. [Run away exception is a RuntimeException that is not handled by any exception handler]. 2. Create an exception class StoreNotFoundException. 3. Create an end point to search store by email. 4. Create exception handler specific to the controller. 5. Let the service throw StoreNotFoundException if no store is found for given email. 6. Let the StoreNotFoundException handler route to view stores page and show the error message in view stores page. 7. Enter the url http://<host>:<port>/stores?email@test.com. This should route to view stores page and show the error message. |
| **Technical Environment** | - |
| **Guidelines** | - |
| **Duration** | 180 mins |

**Spring Security**

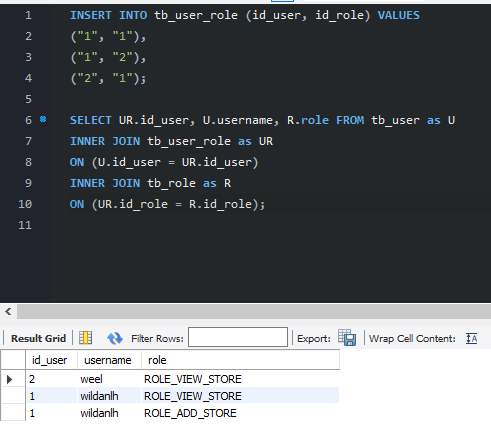
1. Create user, role and user\_role tables in the database.
   1. [Note: The words user & role are reserved in some databases.. Use some other name, if so.]



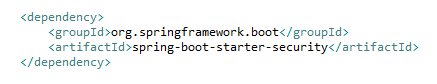
1. Add entries into user, role and user\_role tables.
   1. Add two entries into role table. One role is to view the store info (VIEW\_STORE) and the other role is to add/modify store info (ADD\_STORE).



* 1. Add two entries into user table. Let one user have play VIEW\_STORE role, another user play only both VIEW\_STORE & ADD\_STORE roles.

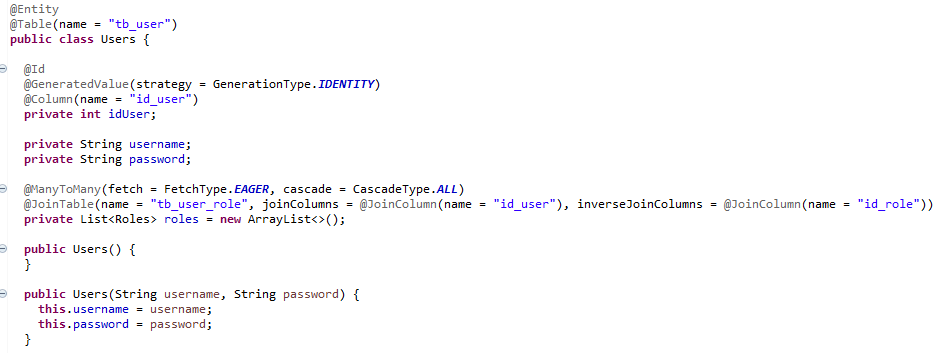


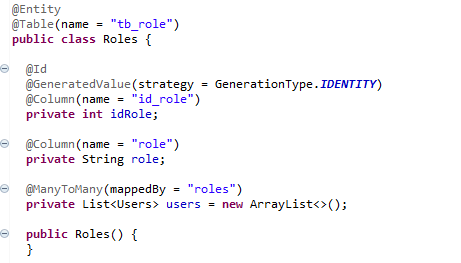
1. Enhance Spring Data JPA enabled version of ‘Know-Your-Neighborhood’ application to support authentication and authorization.

Add maven dependency  


Create Security Config class  


1. Authentication:
   1. Create entities for User and Role.

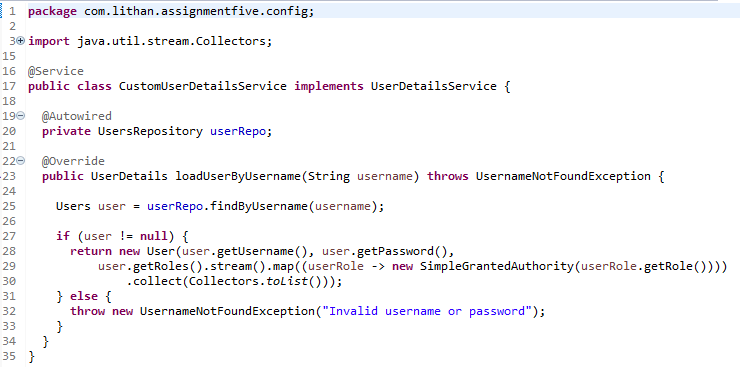
**Users**  


**Roles**  


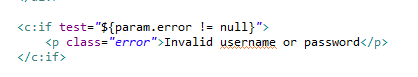
* 1. Create a login page for the application. User should be forced to login before visiting any of the application page.

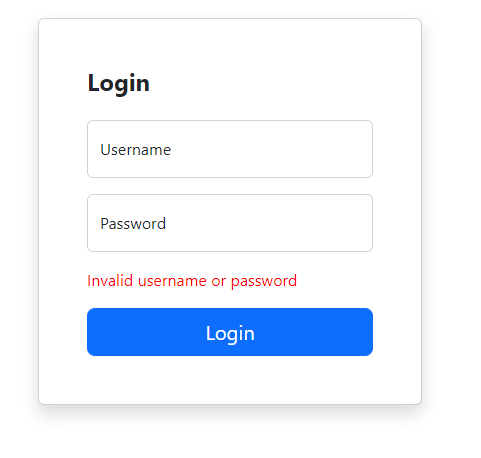
**login.jsp  
**

* 1. Authenticate user with the credentials entered against the database entries added in the previous step.

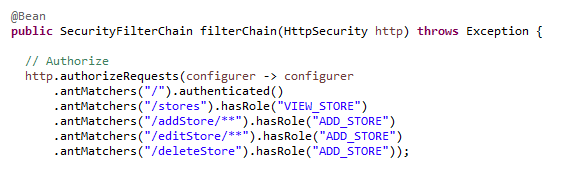


* 1. Show a message if user enters invalid user and password. And then allow him to re-enter the credentials.

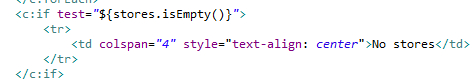


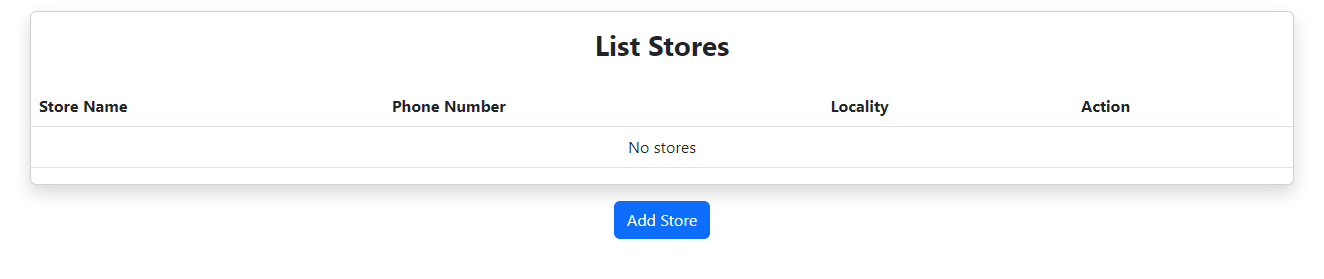


1. Authorization:
   1. Once the user loggedin, make sure that only the user with ADD\_STORE role can add/modify store data. The user with VIEW\_STORE data can only view the stores entered.



* 1. If there are no stores in the system, show appropriate message to the user.



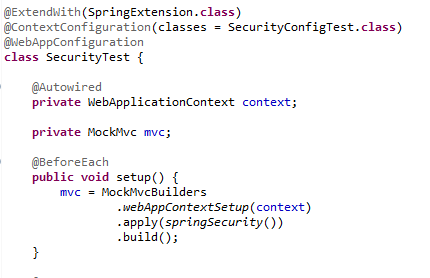


1. Add CSRF security to the application

**login.jsp**  


**Logging**

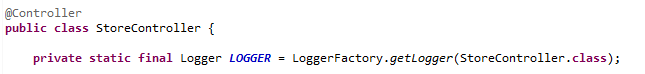
1. Develop 5 unit test cases to validate the role based authorization

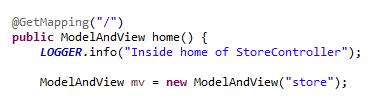




**Logging**

1. Implement the rolling file appender and log at the class level

**StoreController.java  
**

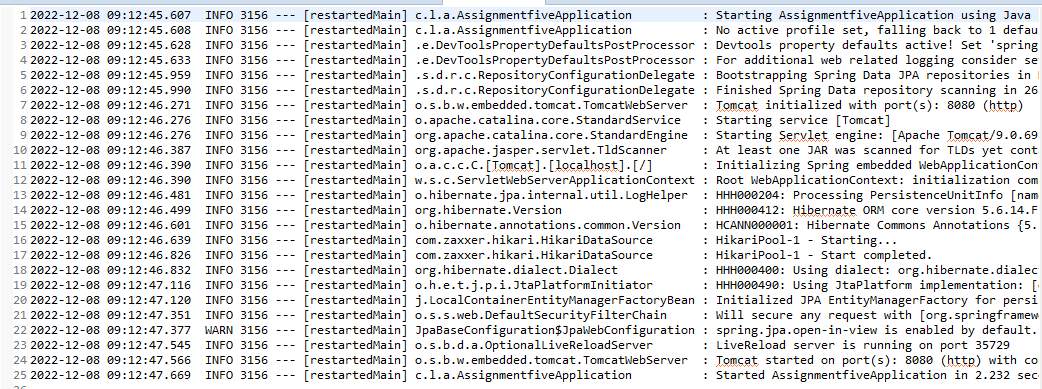
****

****

****

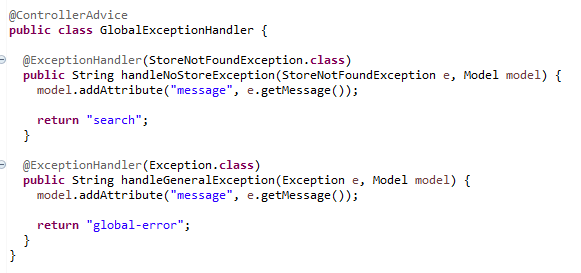
**application.properties  
**

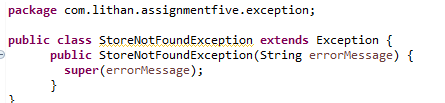
****

****

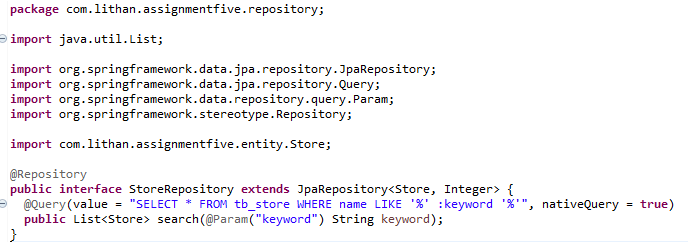
**Error Handling**

1. Create Global Exception Handler that can handle an exception. Add method to handle run away exception. [Run away exception is a RuntimeException that is not handled by any exception handler].
2. Create an exception class StoreNotFoundException.
3. Create an end point to search store by email.
4. Create exception handler specific to the controller.
5. Let the service throw StoreNotFoundException if no store is found for given email.
6. Let the StoreNotFoundException handler route to view stores page and show the error message in view stores page.
7. Enter the url http://<host>:<port>/stores?email@test.com. This should route to view stores page and show the error message.

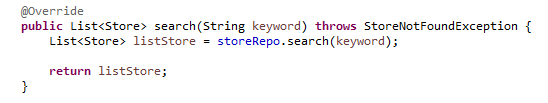
**Global Exception Handler & Store Not Found Exception Handler  
**

**StoreNotFoundException  
**

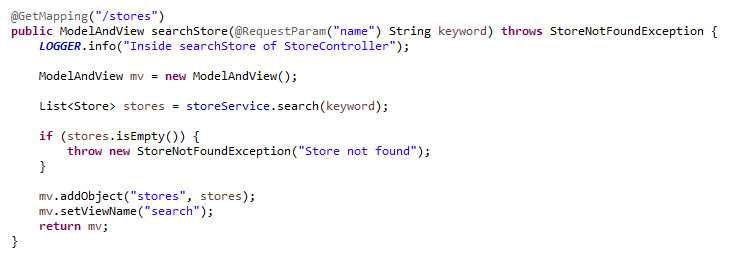
**Search end point by store name**

**StoreRepository  
**

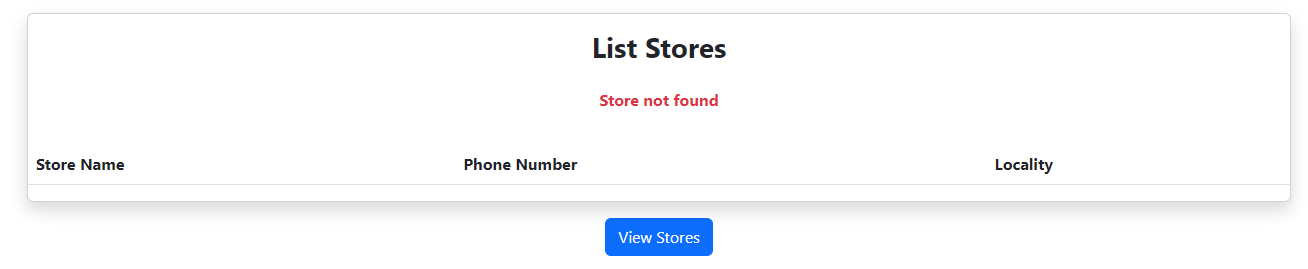
**StoreServiceImpl**

****

**StoreController**

****

**Testing the error  
**

****

**Source Code  
**