**Employee System Database**

Technical Manual

Class: CPSC 488 Section 2

02/26/2023

Maya Smallwood [mns1013@sru.edu](mailto:mns1013@sru.edu)

Paul Schlabach [pms1006@sru.edu](mailto:pms1006@sru.edu)

Bradon Walters [bdw1006@sru.edu](mailto:bdw1006@sru.edu)

# Contents

[**Overview**](#_e87ql4s3kayu) **3**

[**UMLS**](#_cmkk8uq0ldu) **4**

[**Classes**](#_v8k56g9ialwq) **5**

[**Resources**](#_t9eaf1p1c64c) **6**

[**Application Properties**](#_1ebqtwf4gaq2) **7**

# Overview

This workspace is intended to act as an addition to an existing project or can be configured into something more; this workspace is made up of an employee system database that utilizes MySQL Workbench to manage employees while using Spring Boot Thymeleaf Model, View, and Controller (MVC) framework to develop the web applications. The program allows registration of new users, edit of users, and deletion of users. It also has authtrion for login based upon if the user’s role is Admin or an Employee taking the user to the apportice landing-page. It enforces the password policy using the Spring Framework Security of BcryptPasswordEncoder library. While also using Models that create entities and connections to MySQL. To use this program with their own projects, programmers should copy the classes, controllers format, models, and the securityconfig into their projects. The Users and UserRepository may need to be configured for your likely considering that is our main classes that work with our registration pages.

# UMLSFigure 1. Case Diagram for Admin ControlFigure 2. Case Diagram for Employee Control

# Classes

# Resources

# Application Properties

**spring.jpa.hibernate.ddl-auto**=update

Used to embed a database and create a schema using the update attribute to update the connected Database every time a change is made and the program runs.

**spring.datasource.url**=jdbc:mysql://localhost:3306/demo?createDatabaseIfNotExist=true

The URL or link to access the MySQL workbench database. It is set to automatically create a schema rather than importing it from a SQL file.

**spring.datasource.username**=root

The username that is authorized to the server to be accessed.

**spring.datasource.password**=GAzebo23\!!

The password that is authorized to the server to be accessed. Will be required to be changed based upon the individual programmer’s password on that machine.

**spring.jpa.properties.hibernate.format\_sql**=true

**spring.main.allow-bean-definition-overriding**=true

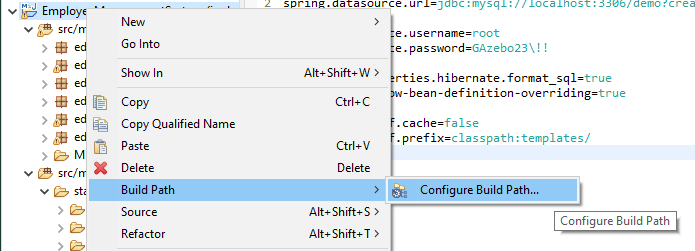
Both of these allows an overriding of the @Bean tag names, an error that would occur from time to time.

**spring.thymeleaf.cache**=false

**spring.thymeleaf.prefix**=classpath:templates/

Used to helped thymeleaf locate the HTML pages inside of templates, this is optional another fix is the following:

**Right click on your main project and navigate to the following:**

Build Path → Configure Build Path…  
  
Figure 3. Thymeleaf Build Path

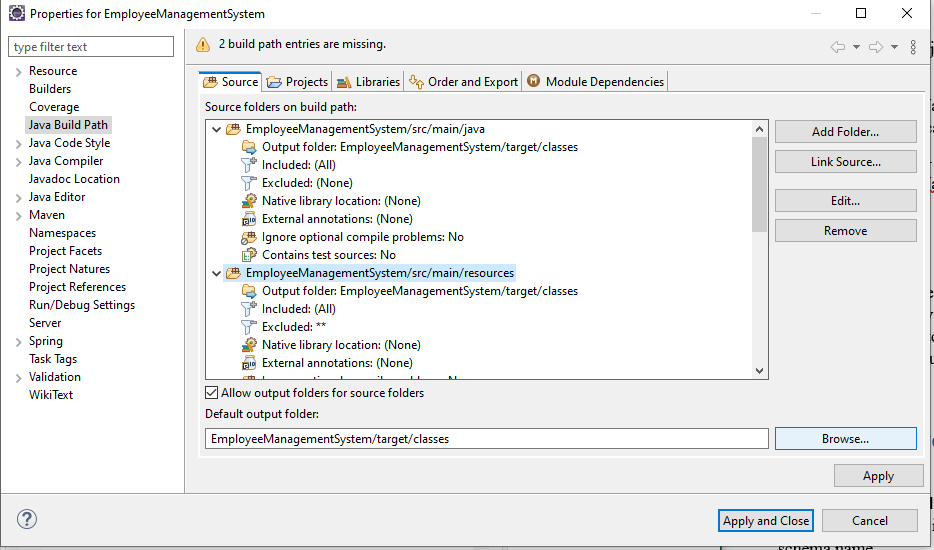
Tab over to Source → and then move down to “YourProject’sName/src/main/resources” → Double click “Excluded: \*\*”  


Figure 4. Thymeleaf Build Path

Next select Remove in the Exclusion Patterns Area”

then select Add Multiple… in the Inclusion Patterns Area:

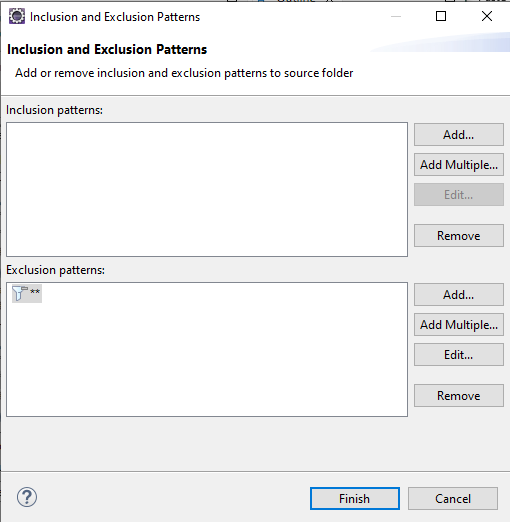
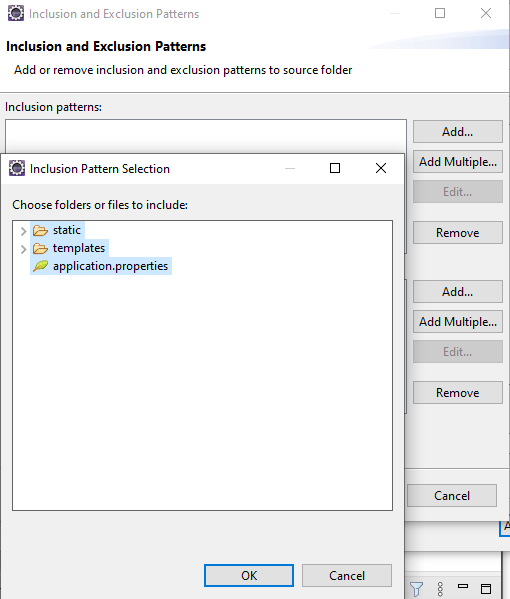
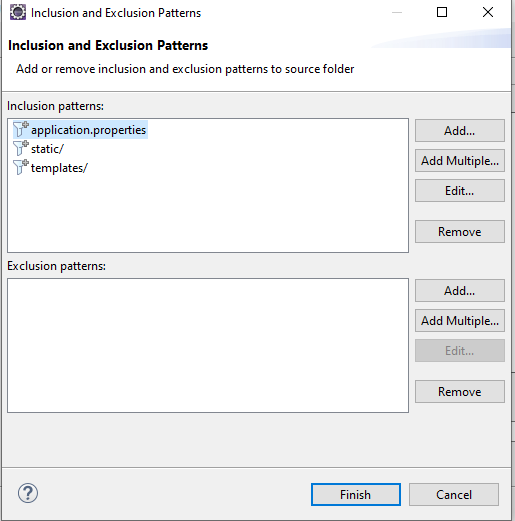


Figure 5. Thymeleaf Build Path

Do a Ctrl + A to select all the following files and then select OK  
Figure 6. Thymeleaf Build Path

Then select Finish  
  
Figure 7. Thymeleaf Build Path

Select Apply and Close and then wait for your project to finish building and now Thymeleaf will stop giving errors of: *“Cannot find template location: classpath:/templates/ (please add some templates or check your Thymeleaf configuration)”*\*Note: You may have to redo this progress if you close and reopen the project, or if you do a Maven→Update Project\*

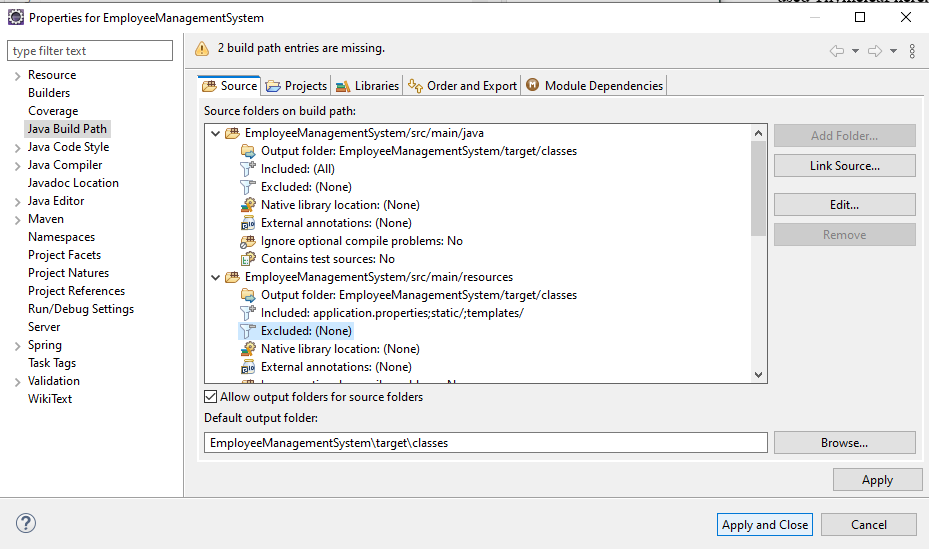


Figure 8. Thymeleaf Build Path