JAVA TECHNOLOGY

(503111)

LAB 8

EXERCISE 1

An exercise on how to set up a basic mvc web project using Spring Boot and the Thymeleaf view template.

Use the Spring Initializr tool provided online at https://start.spring.io/ to create a spring project according to the requirements below:

- Project type: maven
- Language: java
- Project Metadata: set up to you
- Packaging: Jar
- Java version: choose according to the JDK version on your computer
- Add the following dependencies: Lombok, Spring Web, Thymeleaf.
- Download the project as a zip, extract it, and open it with one of the supported programming tools: Intellij Idea, Visual Studio Code, Spring Tool Suite, etc.

Create a web controller named HomeController and set the request mappings as follows:

- /: This path only accepts requests sent by the GET method. When accessing this url, the web server needs to return the content from the index.html file, which is equivalent to the homepage. Your index.html file can contain any html content, however it must contain a hyperlink that links to the /contact path.

- **/contact**: This path accepts both GET and POST requests. When the request is sent using GET method, the server returns the content of the contact.html file containing an html form to collect user information. When the user submits the form, the information will be sent to /contact by POST method. The server then displays the information just received (using a html template file).
- **/about**: This path only accepts GET requests. When accessing this url, the web server returns a message saying "About this site" directly without using any html template.

If the user sends a request using an unsupported http method (e.g. sending a POST to /about) or visits any other url that is not on the mentioned list, the site should display a specific error message each case, **do not** use default the built-in Whitelabel Error Page in Spring Boot.

EXERCISE 2

Using the given html template files to create a website to manage employes. The website should have the employee management functions organized into the following urls:

/employees:

- GET request: List all employees in a html table
- POST request: not available

/employees/add:

- GET request: show an html form to add a new employee
- POST request: receive the employee information sent from the GET request, save the new employee and redirect to /employees to show the result.

/employees/edit/{id}:

- GET request: show an html form to edit the employee information.
- POST request: receive the employee information sent from the GET request, update the employee from data storage then redirect to /employees to show the result.

/employees/delete/{id}:

- GET request: not available
- POST request: perform delete employee based on the given id then redirect to /employees to show the result.

Use in conjunction with Spring Data JPA learned in previous lessons to read/write real data from a database server.