JAVA THYMLEAF PROJECT

EXERCISE 4

Manmeet Singh Kohli

991667681

PROG32758

Balwinder Kaur

STEPS with screenshot

Step 1

Create a new project with name lec41\_thymleaf

A screenshot of a computer

Description automatically generated  
  
Click next

Now add only required dependencies as shown in the image below

A screenshot of a computer

Description automatically generated  
  
As Mentioned in the exercise we need to make a CourseController class in the controllers package as shown in the image below  
A screenshot of a computer

Description automatically generated  
  
Now we made a Coursecontroller class in the controllers package and added @Controller and @GetMapping notations and a method index returning index page. Also, we injected a Model in the argument in the method .Also we added a attribute as a CourseList in which we are course objects

Let’s Look at the Beans Package.

Here we encapsulated fields like prefix,code,name and id ..

Used Lombok annotations to shorten the length of code.

A screenshot of a computer

Description automatically generated  
  
  
  
  
  
A screenshot of a computer

Description automatically generated with medium confidence

Added @Data for getter and setter ,hashcode and for toString methods.

Added @AllArgsConstructor to declare a method with all arguments or all variables declared in the class passing through it.

Now in the CourseController Class we declared a courseList and a method returning index page .

Injecting Model class and adding courseList as an attribute to model object.

Moreover we are declaring a new method with @PostMapping annotation to get the data from form we can the url in the PostMapping is helping us to connect to the form in the index.html page.

So what we doing here is getting all the parameters from the form with @RequestParams annotations as an argument and setting all the values by encapsulation from beans package and storing them in the Course object and than adding that object to the CourseList.

I hope you understand what I am trying to explain.   
  
  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
A screenshot of a computer program

Description automatically generated with medium confidence  
  
  
  
  
  
  
Let’s look on index.html page  
The image is ignoring the css part right know .  
Let’s See what we are doing In this page..

A screenshot of a computer program

Description automatically generated with low confidence  
So basically we can see that we are making a table and in the first row of the table we are adding the table data and the th:each iterated over the each object in the course list and the th:text which is showing mainly the aCourse.getid and rest same for all..

So basically we can see that when we click on addcourse all the data is shown on the top of the page just because of this table and thmyleaf dynamic data  
  
  
  
  
  
  
  
  
A screenshot of a computer

Description automatically generated  
  
  
  
When done adding the course just open the new window than run the localhost because we added the model attributes to the index page..  
  
  
  
  
  
A screenshot of a computer

Description automatically generated  
  
  
  
  
  
Coding Part  
Index.html Page  
  
<!DOCTYPE html>

<html xmlns:th=*"https://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style>

**body** {

font-family: *Arial, sans-serif*;

background-color: *#f2f2f2*;

margin: *0*;

padding: *0*;

}

**table** {

border-collapse: *collapse*;

margin: *20px*;

}

**table** **th,**

**table** **td** {

padding: *8px*;

border: *1px solid #ccc*;

}

**form** {

margin: *20px*;

}

**input**[type="number"]**,**

**input**[type="text"]**,**

**input**[type="submit"] {

margin-top: *8px*;

padding: *6px*;

font-size: *16px*;

}

</style>

</head>

<body>

<table>

<tr th:each=*"aCourse:${courseList}"*>

<td th:text=*"${aCourse.id}"*>55</td>

<td th:text=*"${aCourse.prefix}"*>PROG</td>

<td th:text=*"${aCourse.code}"*>32758</td>

<td th:text=*"${aCourse.name}"*>Manmeet</td>

</tr>

</table>

<form method=*"post"* action=*"#"* th:action=*"@{/addCourse}"*>

ID: <input type=*"number"* name=*"id"* /><br>

Prefix: <input type=*"text"* name=*"prefix"* /><br>

Code: <input type=*"number"* name=*"code"* /><br>

Name: <input type=*"text"* name=*"name"* /><br>

<input type=*"submit"* value=*"Add Course!"* />

</form>

</body>

</html>

Course Controller  
**package** ca.sheridancollege.kohliman.controllers;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** java.util.List;

**import** java.util.concurrent.CopyOnWriteArrayList;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** ca.sheridancollege.kohliman.beans.Course;

@Controller

**public** **class** CourseController {

List<Course> courseList = **new** CopyOnWriteArrayList<Course>();

@GetMapping("/")

**public** String index(Model model) {

model.addAttribute("courseList",courseList);

**return** "index";

}

@PostMapping("/addCourse")

**public** String addCourse(@RequestParam("id")Long id,

@RequestParam("prefix") String prefix,

@RequestParam("code") String code,

@RequestParam("name") String name) {

Course course = **new** Course(id, prefix, code, name);

courseList.add(course);

**return** "index";

}

}

Course Class  
  
**package** ca.sheridancollege.kohliman.beans;

**import** lombok.AllArgsConstructor;

**import** lombok.Builder;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@Builder

**public** **class** Course {

**private** Long id;

**private** String prefix;

**private** String code;

**private** String name;

}

Summary  
So in thymleaf project what we are doing is in the course class we are just encapsulating the fields and than in the coursecontroller class we are declaring two methods one for the index page to show to the client while running the tomcat server and the second is the PostMapping one where we are doing form handling getting the data from the form and using RequestParam annotations to get the data directly from the form and setting the all the fields via encapsulating and adding in the course object and than that object to the courseList .

This Course list is being called in the attribute of the model class in the index method.

So how dynamically the data got changed which is being added by the client.

This is the power of thymleaf