Exercise 5\_2 HD and JDBC

991667681

Balwinder Kaur

Step 1

Click on new and than other

A screenshot of a computer

Description automatically generated

Make a project with name lec51\_H2andJDBC as shown

A screenshot of a computer

Description automatically generated  
  
  
Than we need to select those dependencies that we need in this project  
A screenshot of a computer

Description automatically generated  
  
  
Than click on finish  
Than after that we are making pojo class or we can say a beans package class that includes all the getters and setters

A screenshot of a computer

Description automatically generated  
  
  
Than declare the private Long Id and private String name

A screenshot of a computer

Description automatically generated with medium confidence

Than add the other Lombok annotations and import the libraries as shown below  
  
A screenshot of a computer program

Description automatically generated with medium confidence  
  
  
  
Than make another Class StudentController within the same package src and main

A screenshot of a computer

Description automatically generated  
  
Than we added one method in this class mapping to / and returning the index page   
A screenshot of a computer program

Description automatically generated with low confidence

Than we made a DatabaseAccess class

A screenshot of a computer

Description automatically generated

Than right click on the src/main/resources and than click on the other and select Html file

2A screenshot of a computer

Description automatically generated  
  
  
  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
  
Add this line to the html tag

A screenshot of a computer

Description automatically generated with low confidence  
  
Than add table and form as given in the lecture  
appA screenshot of a computer

Description automatically generated with medium confidence

Than we need to these two spring lines in the application properties

A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
You will get this error when you will enter the local host link of port 8080

A screenshot of a computer

Description automatically generated with medium confidence

Now change the path of the link and enter /h2-console at the end like this

ClickA screenshot of a computer

Description automatically generated  
  
Click on the connect button as everything default is correct

A screenshot of a computer

Description automatically generated

We don’t have any stored info yet so if we will run any query obvisously that will not run

So we need to create a table first in the schema.sql in the src/main/resources as shown

A screenshot of a computer

Description automatically generated with medium confidence”  
Now we will make a data.sql file as shown

A screenshot of a computer screen

Description automatically generated with medium confidence  
  
Let’s add some data in the data.sql file

A screenshot of a computer

Description automatically generated with medium confidence  
  
  
  
Now we need to make a another class with .database and name DatabaseAccess

so we need to do some few steps as shown in the figure belowz

A screenshot of a computer program

Description automatically generated with low confidence  
  
  
  
Now add or we can call the insertStudent method from the database class

A screenshot of a computer

Description automatically generated with medium confidence

Now debug the projrct and run again and than run the local host again

Run the query select \* from students

A screenshot of a computer

Description automatically generated with medium confidence

Now make some changes to index.html page

A screenshot of a computer

Description automatically generated  
  
  
  
Now run the local host and see what happens

A screenshot of a computer

Description automatically generated with medium confidence

See this is the page we are getting

Now enter a name in the name box  
A screenshot of a computer

Description automatically generated with medium confidence  
  
  
Now Click GO

A screenshot of a computer

Description automatically generated  
  
Now run the h2-console see what happens  
A screenshot of a computer

Description automatically generated  
  
  
So we can see that how the name is inserted into the database table

A screenshot of a computer

Description automatically generated with medium confidence  
  
  
Bean Code  
package ca.sheridancollege.kohliman.beans;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

public class Student {

private long id;

private String name;

}

Student Controller

package ca.sheridancollege.kohliman.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import org.springframework.web.bind.annotation.ModelAttribute;

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

import ca.sheridancollege.kohliman.beans.Student;

import ca.sheridancollege.kohliman.database.DatabaseAccess;

@Controller

public class StudentController {

@Autowired

private DatabaseAccess da;

@GetMapping("/")

public String index(Model model) {

// da.insertStudentHardCoded();

model.addAttribute("student", new Student());

model.addAttribute("studentList", da.getStudentList());

return "index";

}

@PostMapping("/insertStudent")

public String insertStudent(Model model, @ModelAttribute Student student) {

da.insertStudent(student);

model.addAttribute("student", new Student());

model.addAttribute("studentList", da.getStudentList());

return "index";

}

}  
  
  
  
  
  
  
DatabaseAccess Class  
package ca.sheridancollege.kohliman.database;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;

import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;

import org.springframework.stereotype.Repository;

import ca.sheridancollege.kohliman.beans.Student;

@Repository

public class DatabaseAccess {

@Autowired

protected NamedParameterJdbcTemplate jdbc;

public void insertStudentHardCoded() {

MapSqlParameterSource namedParameters = new MapSqlParameterSource();

String query="INSERT INTO student(name) VALUES ('Frank')";

int rowsAffected = jdbc.update(query, namedParameters);

if (rowsAffected > 0)

System.out.println("Inserted student into database.");

}

public void insertStudent(Student student) {

MapSqlParameterSource namedParameters = new MapSqlParameterSource();

String query="INSERT INTO student(name) VALUES (:name)";

namedParameters.addValue("name", student.getName());

int rowsAffected = jdbc.update(query, namedParameters);

if (rowsAffected > 0)

System.out.println("Inserted student into database.");

}

public List<Student> getStudentList()

{ MapSqlParameterSource namedParameters =new MapSqlParameterSource();

String query = "SELECT \* FROM student";

return jdbc.query(query, namedParameters, new BeanPropertyRowMapper<Student>(Student.class));

}

}  
  
  
  
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: *20px*;

}

**h1** {

color: *#333*;

}

**table** {

border-collapse: *collapse*;

width: *100%*;

margin-bottom: *20px*;

}

**table** **th,** **table** **td** {

padding: *8px*;

text-align: *left*;

border-bottom: *1px solid #ddd*;

}

**table** **th** {

background-color: *#f5f5f5*;

}

**table** **td***:nth-child*(*2n*) {

background-color: *#f9f9f9*;

}

**form** {

margin-top: *20px*;

}

**form** **input**[type="text"] {

padding: *8px*;

width: *200px*;

border: *1px solid #ccc*;

border-radius: *4px*;

}

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

padding: *8px 16px*;

background: *linear-gradient(45deg, #ff0000, #ff7f00, #ffff00, #00ff00, #0000ff, #8b00ff)*;

color: *white*;

border: *none*;

cursor: *pointer*;

border-radius: *4px*;

}

**form** **input**[type="submit"]*:hover* {

background: *linear-gradient(45deg, #ff0000, #ff7f00, #ffff00, #00ff00, #0000ff, #8b00ff)*;

background-size: *300%*;

animation: *rainbow 2s linear infinite*;

}

@**keyframes** **rainbow** {

0% {

background-position: *0% 50%*;

}

*100*% {

background-position: *100% 50%*;

}

}

*.form-error* {

color: *red*;

font-size: *12px*;

margin-top: *4px*;

}

</style>

</head>

<body>

<h1>Hi Welcome to my index.html Page</h1>

<h2>Please Enter Your name </h2>

<table th:if=*"${studentList != null AND studentList != ''}"*>

<tr th:each=*"s : ${studentList}"*>

<td th:text=*"${s.name}"*>Frank</td>

</tr>

</table>

<hr>

<form method=*"post"* action=*""* th:action=*"@{/insertStudent}"* th:object=*"${student}"*>

<input type=*"hidden"* name=*"id"* th:field=*"\*{id}"*/>

Name: <input type=*"text"* name=*"name"* th:field=*"\*{name}"* /><br>

<span class=*"form-error"* th:if=*"${#fields.hasErrors('name')}"* th:errors=*"\*{name}"*></span>

<input type=*"submit"* value=*"Go!"* />

</form>

</body>

</html>

Schema.sql

**CREATE** **TABLE** student(

id LONG **PRIMARY** **KEY** AUTO\_INCREMENT,

name **VARCHAR**(255)

);

data.sql

**INSERT** **INTO** student(name) **VALUES**

('Sally'),

('Harminder'),

('Hao'),

('Jaspreet');

Summary  
  
Rest everything is same as explained in the previous exercise but the different here is that we added a new method in the DatabaseAccess Class named insertStudent() that is overriding but in the query we changed the :name that will add the hardcoded name and is added to the Database table at the same time that we can see from the h2-console as well.And we are calling that method from the StudentController Class .And in the Database Class we are declaring a method that will return a StudentList ,also declaring the MapSqlParameters namedParameters and than declaring the Query and returning the StudentList as a Jdbc query . So in this case when we will enter the hardcodename inside the Name column than that will call the method insertStudent and the name will be added to the StudentList also will be added to the query as we can see in the abor screenshots.