**Spring MVC**

**Student Management Systems:**

**R1:** The system shall be able to add student by entering student information (first name, last name, program name, email address)

**R2:** The system shall be able to view the students that are stored in the database

**R3:** The system shall be able to edit the student information (name, program)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | First Name | Last Name | Program Name | Email | EDIT | DELETE |
| 1 | X | Y | MS | XY@nwmissouri.edu | EDIT | DELETE |
|  |  |  |  |  |  |  |

**R4:** The system shall be able to update the student’s information based on id

**R5.** The system shall be able to delete the student based on id

**MySQL Configuration SetUp:** Please add the following lines to the ***application.properties*** file.

spring.datasource.url=jdbc:mysql://localhost:3306/fms?useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false

spring.datasource.username=root

spring.datasource.password=root

server.port=8080

#Hibernate

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect

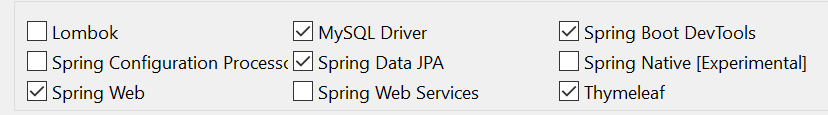
#Hibernate auto ddl

spring.jpa.hibernate.ddl-auto=update

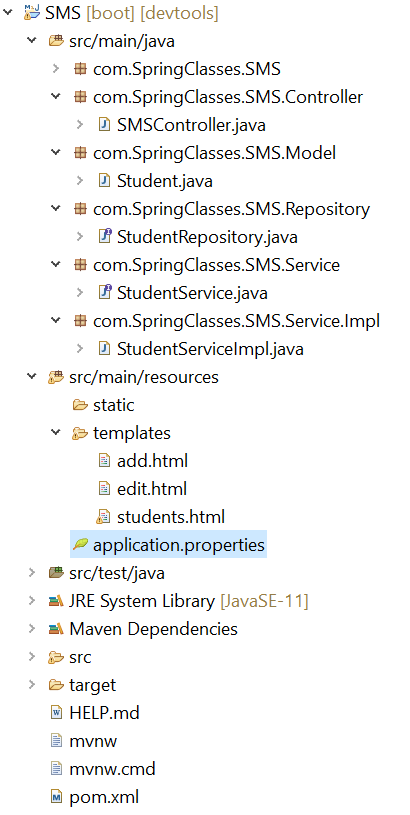
spring.jpa.show-sql=true

logging.level.org.hibernate.SQL=DEBUG

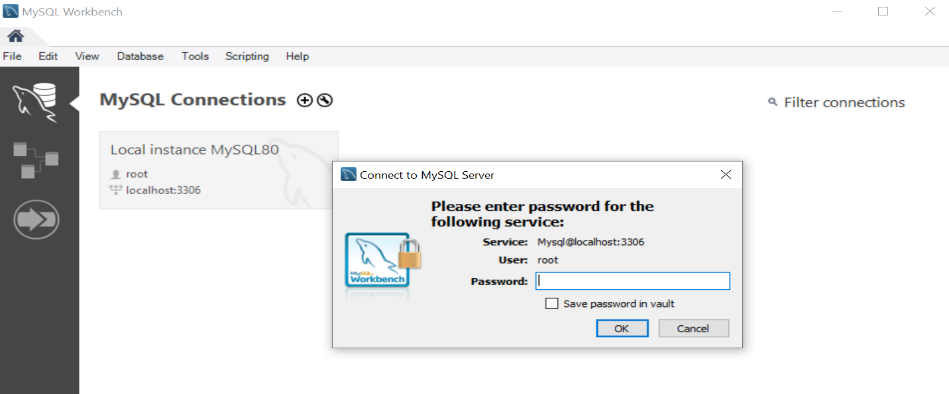
**Step-1:** Create a Spring boot Starter Project named “**SMS**” project which requires the following five packages:

****

**Step-2:** Create the following sub-packages as given in the following structure:



**Step-3:** Open MYSQL workbench and login as a root user.



**Step-4:** Create a database named **“sms”** via the following query.  
 **create database sms;   
  
Step-5:** Create a class named **“Student”** within the **com.SpringClasses.SMS.Model** sub-package to capture student’s data.

1. Add class level annotation **@Entity** to interact with the student’s information stored in **“sms”** database.
2. Add class level annotation **@Table(name=“students”)** to map with a table named **“students”**.
3. Create a private field named **“id”** of type Long.
4. Add the following annotation just above on the id field to make it as a primary key:
   1. @Id
   2. @GeneratedValue(strategy=GenerationType.***IDENTITY***)
5. Create the following private fields of Type String: firstName, lastName, programName, and email
6. Add **@Column** annotation just above on each of the fields created in step (e) to bind the field name with the column name. For example, for the firstName,

@Column(name ="first\_name", nullable=**false**)

**private** String firstName;

1. Create getter and setter for all the fields

**Step-6:** Create an interface named “**StudentRepository”** within the **com.SpringClasses.SMS.Repository** subpackage that extends JpaRepository to provide the service for CRUD operation on students.

**public** **interface** StudentRepository **extends** JpaRepository<Student, Long>{ }

**Step-7:** Create an interface named “**StudentService**” within **com.SpringClasses.SMS.Service** subpackage to define own methods for the CRUD operation on students as given below:

**public** List<Student> getAllStudents();

**public** Student saveStudent(Student student);

**public** Student getStudentById(Long id);

**public** Student updateStudent(Student student);

**public** **void** deleteStudentById(Long id);

**Step-8:** Create a class named **“StudentServiceImpl”** within the **com.SpringClasses.SMS.Service.Impl** sub-package to implement methods declared in the interface **StudentService**. Since this class is implementing the interface, please add class level annotation **@Service** as given below.

@Service

**public** **class** StudentServiceImpl **implements** StudentService{

}

1. To implement all the methods declared in step-7, we need to access **“sms”** database and perform operations on the table **“students”**. We can achieve this using **StudentRepository**.
2. Declare an object reference variable named **studentRepository** of Type **StudentRepository**.

**private** StudentRepository studentRepository;

1. Create a parameterized constructor to inject an object of type **StudentRepository**.

**public** StudentServiceImpl(StudentRepository studentRepository) {

**this**.studentRepository = studentRepository;

}

1. Implement all the methods using services provided by **StudentRepository** as given below:

@Override

**public** List<Student> getAllStudents()

{

**return** studentRepository.findAll();

}

@Override

**public** Student saveStudent(Student student)

{

**return** studentRepository.save(student);

}

**public** Student getStudentById(Long id)

{

**return** studentRepository.findById(id).get();

}

**public** Student updateStudent(Student student)

{

**return** studentRepository.save(student);

}

**public** **void** deleteStudentById(Long id)

{

studentRepository.deleteById(id);

}

**Step-9:** Create a class named **“SMSController”** within **com.SpringClasses.SMS.Controller** subpackage.

1. Add class level annotation **@Controller** to this class
2. Declare an object reference variable named **studentService** of Type **StudentService**.

**private** StudentService studentService;

1. Create a parameterized constructor to inject an object of type **StudentService**.

**public** SMSController(StudentService studentService)

{

**this**.studentService = studentService;

}

1. Create a controller method named **listStudents** to display all the students stored in the table **“students”** on an HTML page named **“students.html”**.

@GetMapping("/students")

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

model.addAttribute("students",studentService.getAllStudents());

**return** "students";

}

1. Create an HTML page named “students.html” within **src/main/resources/templates/** folder. Please change html tag with <html xmlns:th=*"http://www.thymeleaf.org"*> to use thymeleaf feature.
   1. Change the title with “Student Management Systems”
   2. In the body, create a hyperlink that will be redirecting to the same page through the controller.  
      <div>

<a th:href=*"@{/students}"*>Student Management Systems</a>

</div>

* 1. a Add a button for adding a student via the following code:  
     <div class=*"btn"*> <a th:href=*"@{/students/add}"*>Add</a></div>
  2. Display all the data stored in the model attribute **“students”** on this page.   
       
     <table border=*"True"*>

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

<th>Program Name</th>

<th>Email</th>

<th> Actions </th>

</tr>

</thead>

<tbody>

<tr th:each=*"student: ${students}"*>

<td th:text=*"${student.firstName}"*></td>

<td th:text=*"${student.lastName}"*></td>

<td th:text=*"${student.programName}"*></td>

<td th:text=*"${student.email}"*></td>

<td>

<a th:href = *"@{/students/edit/{id}(id=${student.id})}"* class = *"btn"*>EDIT</a>

<a th:href = *"@{/students/{id}(id=${student.id})}"* class = *"btn"*>DELETE</a></td>

</tr>

</tbody>

</table>

1. Next, we define a controller method for adding a student’s information as given below:

@GetMapping("/students/add")

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

Student student=**new** Student();

model.addAttribute("student", student);

**return** "add";

}

1. Create an html page named **“add.html”** to add a student within **src/main/resources/templates/** folder. Please change html tag with <html xmlns:th=*"http://www.thymeleaf.org"*> to use thymeleaf feature.
   1. Change the title with “Student Management Systems”
   2. Add the following form in the body for the student information:  
      <form th:action=*"@{/students}"* th:object=*"${student}"* method=*"POST"*>

<div>

<label> First Name </label>

<input type=*"text"* name=*"firstName"* th:field=*"\*{firstName}"*

placeholder=*"Enter Student First Name"*/>

</div>

<div>

<label> Last Name </label>

<input type=*"text"* name=*"lastName"* th:field=*"\*{lastName}"*

placeholder=*"Enter Student Last Name"* />

</div>

<div>

<label> Program Name </label>

<input type=*"text"* name=*"programName"* th:field=*"\*{programName}"*

placeholder=*"Enter Student Program Name"* />

</div>

<div >

<label> Email </label>

<input type=*"text"* name=*"email"* th:field=*"\*{email}"*

placeholder=*"Enter Student Email"* />

</div>

<div>

<button type=*"submit"* class=*"btn"*>Submit</button>

</div>

</form>

1. When you submit the above form, in the controller class, @PostMapping with /students with be searched. Hence, we define a controller method named “saveStudent” as given below:

@PostMapping("/students")

**public** String saveStudent(@ModelAttribute("student") Student student) {

studentService.saveStudent(student);

**return** "redirect:/students";

}

1. Next, we define controller method for editing a student’s information as given below:

@GetMapping("/students/edit/{id}")

**public** String editFacultyForm(@PathVariable Long id, Model model) {

model.addAttribute("student", studentService.getStudentById(id));

**return** "edit";

}

1. Create an html page named **“edit.html”** to add a student within **src/main/resources/templates/** folder. Please change html tag with <html xmlns:th=*"http://www.thymeleaf.org"*> to use thymeleaf feature.
   1. Change the title with “Student Management Systems”
   2. Add the following form in the body to edit student information based on id:  
      <form th:action*=“@{/students/{id} (id=${student.id})}"* th:object=*"${student}"* method=*"POST"*>

<div>

<label> First Name </label>

<input type=*"text"* name=*"firstName"* th:field=*"\*{firstName}"*

placeholder=*"Enter Student First Name"*/>

</div>

<div>

<label> Last Name </label>

<input type=*"text"* name=*"lastName"* th:field=*"\*{lastName}"*

placeholder=*"Enter Student Last Name"* />

</div>

<div>

<label> Program Name </label>

<input type=*"text"* name=*"programName"* th:field=*"\*{programName}"*

placeholder=*"Enter Student Program Name"* />

</div>

<div >

<label> Email </label>

<input type=*"text"* name=*"email"* th:field=*"\*{email}"*

placeholder=*"Enter Student Email"* />

</div>

<div>

<button type=*"submit"* class=*"btn"*>Submit</button>

</div>

</form>

1. When you submit the above form, in the controller class, @PostMapping with /students/{id} with be searched. Hence, we define a controller method named “updateStudent” as given below:

@PostMapping("/students/{id}")

**public** String updateStudent(@PathVariable Long id, @ModelAttribute("student") Student student,

Model model) {

Student existingStudent = studentService.getStudentById(id);

existingStudent.setId(id);

existingStudent.setFirstName(student.getFirstName());

existingStudent.setLastName(student.getLastName());

existingStudent.setProgramName(student.getProgramName());

existingStudent.setEmail(student.getEmail());

// save updated student object

studentService.updateStudent(existingStudent);

**return** "redirect:/students";

}

1. Finally, we need to delete student based on id. For this, we define the following controller method:  
   @GetMapping("/students/{id}")

**public** String deleteStudent(@PathVariable Long id) {

studentService.deleteStudentById(id);

**return** "redirect:/students";

}