

ADP ENDSEM PROJECT

Spring Boot Student Registration with CRUD Operations

Ishita Jena	Pratik S. Mishra	Soumya S. Dash	Gayatri Sahu
2141010031	2141013122	2141004097	2141019146

Group B

Supervised By: Prof. Amul Chourasia



Department Of Computer Science And Engineering
Institute Of Technical Education And Research (ITER)
Siksha 'O' Anusandhan, Deemed To Be University
Bhubaneswar, Odisha

Table of Contents

- 1 Introduction
- 2 Materials And Methods
 - System Architecture
 - Functionalities
 - Technologies Used
- 3 Results And Interpretation
 - Homepage View
 - Adding New Student Record
 - Edit Student Record
 - Deleting Student Record
- 4 Key Takeaways
 - Future Improvements
 - Conclusion

Introduction

Overview

- Built a Student Registration System using Spring Boot to streamline student record management.
- Demonstrates CRUD operations (Create, Read, Update, Delete) with a user-friendly web interface.
- Structured using the Model-View-Controller (MVC) design pattern for modularity and scalability.
- Model Repository Layers: Defines core entities like Student (ID, Name, Course, Fee) and handles database interactions using Spring Data JPA and MySQL.
- Service, Controller View Layers: Encapsulates business logic, processes user requests, and renders a responsive UI with Thymeleaf and Bootstrap.
- Includes features like viewing, adding, editing, and deleting student records, with robust validation and responsive design.
- Utilizes technologies like Spring Boot, Thymeleaf, Bootstrap, MySQL, and Maven for seamless development.

Introduction (continued...)

Why Spring Boot?

- **Robust Framework:** Spring Boot provides a comprehensive ecosystem for building modern applications, integrating seamlessly with tools and technologies like Spring Data JPA, Thymeleaf, and MySQL.
- **Dependency Injection:** Ensures efficient management of application components and promotes modularity and reusability.
- **Database Integration:** Spring Data JPA allows seamless CRUD operations without complex SQL queries.

What If Spring Boot Hadn't Been Used?

- **Manual Configuration:** Setting up the application would have to be done manually, including dependency management, database connections, and server settings.
- **Increased Boilerplate Code:** Writing code for features like dependency injection, transaction management, and database integration.
- **Complex Database Operations:** Without Spring Data JPA, there would be a need to write raw SQL queries.

System Architecture

Model Layer:

- Defines the Student entity (attributes: ID, name, course, fee).
- Mapped to a MySQL table using JPA annotations.

Repository Layer:

- Implements JpaRepository to abstract SQL queries.
- Manages data retrieval, storage, and modification.

Service Layer:

- Encapsulates business logic and validation.
- Provides methods like saveStudent, getAllStudents, and deleteStudent.

Controller Layer:

- Handles HTTP requests and maps them to appropriate service methods.
- Passes data to Thymeleaf templates for rendering.

View Layer:

- Provides a web-based UI using Thymeleaf and Bootstrap for responsiveness.

System Architecture (continued...)

Interaction Flow

- User Action:
 - User interacts with the application (e.g., submits a form to add a student).
- Controller Handling:
 - The controller receives the HTTP request and determines the action to perform.
- Business Logic Execution:
 - The service layer interacts with the repository layer for database operations.
- Database Interaction:
 - The repository layer performs CRUD operations on the MySQL database using JPA and Hibernate.
- Response Generation:
 - Processed data is passed to the controller, which forwards it to Thymeleaf templates for rendering.
 - The user is presented with updated content on the web interface.

Functionalities

Key Features

- View Students: Fetches student data from the service layer and renders it in a tabular format. Includes options for editing and deleting records.
- Add Students: On clicking the “Add New” button, a form appears where users can input details for a new student record.
- Edit Students: On clicking the “Edit” button, existing student records can be modified via pre-filled forms.
- Delete Students: On clicking the “Delete” button, the corresponding student is removed from the database.
- Save Students: Saves the submitted student data after addition, editing or deletion via the service layer and redirects to the homepage.

Additional Features

- Responsive Design: Accessible across devices using Bootstrap.
- Form Validation: Ensures user inputs are correct (no empty fields or invalid data types).

Technologies Used

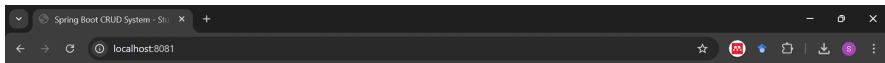
Frameworks and Tools

- Spring Boot: Backend framework for simplifying Java development.
- Thymeleaf: Server-side rendering for dynamic HTML pages.
- MySQL: Reliable database for storing student records.
- Spring Data JPA: Automates database operations with minimal SQL code.
- Bootstrap: Enhances UI design and responsiveness.

Additional Tools

- Maven: Dependency management for efficient project configuration.
- Hibernate: ORM tool integrated with JPA for database schema management.
- Spring Boot DevTools: Enables hot reloading for seamless development.
- MySQL Connector: Acts as a critical bridge between the Spring Boot application and the MySQL database.

Homepage View



Spring Boot CRUD System - Student Registration

Student ID	Student Name	Course	Fee	Edit	Delete
1	Doraemon	Python	1200	Edit	Delete
2	John Wick	Psychology	1500	Edit	Delete

Add New Student

Homepage View (continued)..

Spring Boot Application Initialization

Starts the Spring Boot application, initializes Spring context and embedded server (Tomcat)

Auto-Configuration & Dependency Injection

Spring Boot auto-configures dependencies based on the classpath and `application.properties` (JPA, Web, etc.). Instantiates required beans (e.g., `DataSource`, `JpaRepository`).

Database Configuration

Spring Boot reads database connection details (URL, username, password) from `application.properties`. Configures MySQL connection, JPA, and Hibernate.

Service Layer Interaction

Calls `StudentRepository` to fetch all students.

Controller and View Resolution

`StudentController.java` handles HTTP requests, interacts with `StudentService.java` and returns a view

Entity and Repository Setup

`@Entity` annotation from `Student.java` maps `Student` to the database. Spring Boot auto-generates `StudentRepository` CRUD methods.

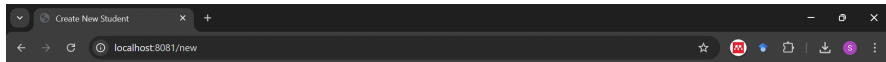
Thymeleaf Integration

Renders the list of students in a table using `index.html` (Thymeleaf Template) and the data passed from `StudentController`.

Server Startup and User Interaction

Embedded Tomcat starts, listening on port 8081. User accesses `http://localhost:8081/` in the browser. Action: Triggers the `viewHomePage` method in `StudentController` to render the homepage with student data.

Adding New Student Record



Create New Student

Please fill out the form to create a new student record.

Student Name

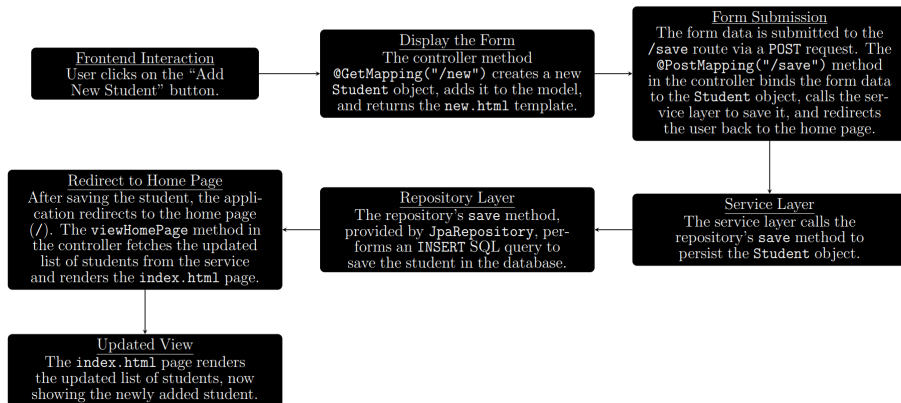
Course

Fee

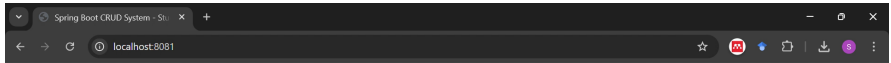
Save

Clear

Adding New Student Record (continued...)



Adding New Student Record (continued...)



Spring Boot CRUD System - Student Registration

Student ID	Student Name	Course	Fee	Edit	Delete
1	Doraemon	Python	1200	Edit	Delete
2	John Wick	Psychology	1500	Edit	Delete
3	Harry Potter	Database	2500	Edit	Delete

Add New Student

Edit Student Record



Create New Student

Please fill out the form to create a new student record.

Student Name

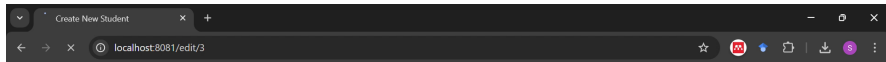
Course

Fee

Save

Clear

Edit Student Record (continued...)



Create New Student

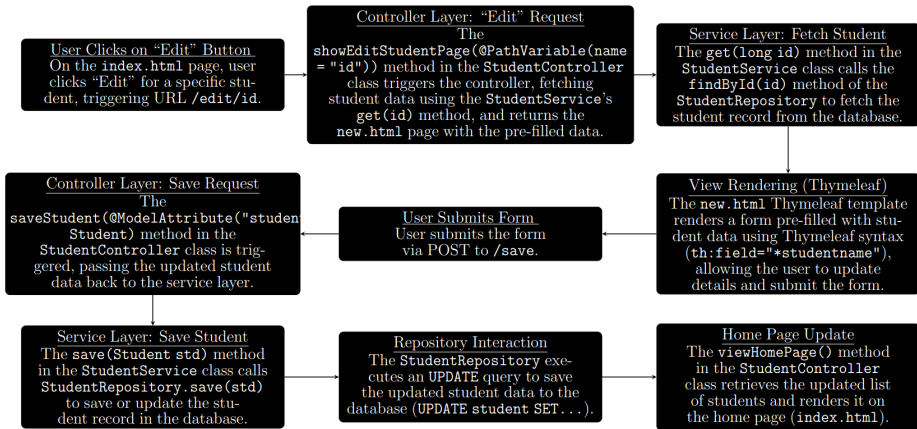
Please fill out the form to create a new student record.

Student Name

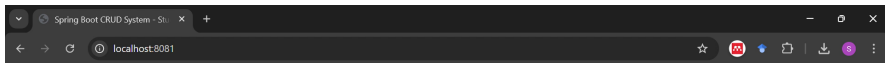
Course

Fee

Edit Student Record (continued...)



Edit Student Record (continued...)

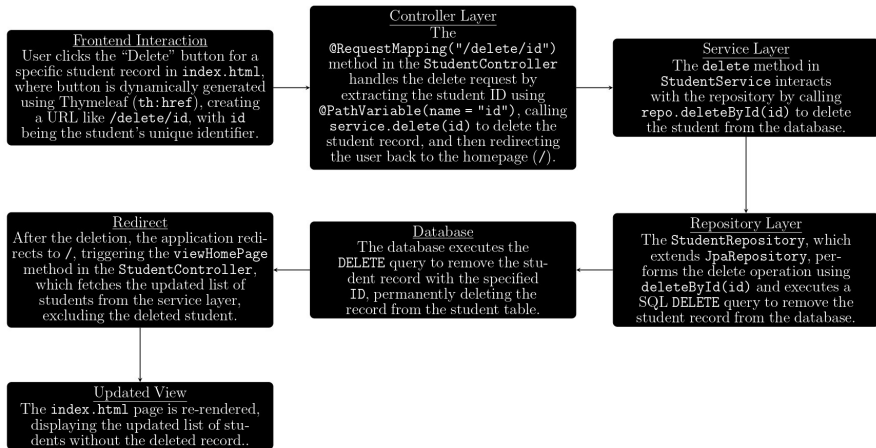


Spring Boot CRUD System - Student Registration

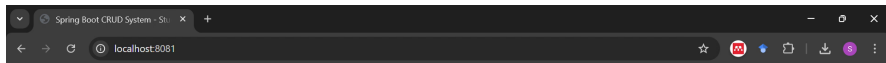
Student ID	Student Name	Course	Fee	Edit	Delete
1	Doraemon	Python	1200	Edit	Delete
2	John Wick	Psychology	1500	Edit	Delete
3	Harry Potter	Database Management Systems	2500	Edit	Delete

Add New Student

Deleting Student Record



Deleting Student Record (continued...)



Spring Boot CRUD System - Student Registration

Student ID	Student Name	Course	Fee	Edit	Delete
1	Doraemon	Python	1200	Edit	Delete
2	John Wick	Psychology	1500	Edit	Delete

[Add New Student](#)

Future Improvements

- **Role-Based Access Control (RBAC):**

- Define roles such as Admin, Teacher, and Student.
- Admins can delete student records, teachers can manage grades, students can view their own details

- **Advanced Search:**

- Search by name, course, or multiple parameters.
- Autocomplete suggestions and intelligent search results to improve user experience

- **Integration with External Systems:**

- Integrate with external tools (LMS, Attendance Management Systems)
- Enable data export/import in formats like CSV or JSON
- Support bulk operations like registering multiple students or batch updates and deletions

- **Notifications:**

- Email or SMS alerts for critical updates, such as new registrations.

- **Analytics Dashboard:**

- Visualize data trends such as course popularity or registration counts.

Conclusion

Summary Of The Project:

- The Spring Boot Student Registration System successfully demonstrates the implementation of core CRUD operations using the Model-View-Controller (MVC) architecture.
- It showcases seamless integration between:
 - Backend: Spring Boot and JPA for business logic and database operations.
 - Frontend: Thymeleaf and Bootstrap for a dynamic, responsive user interface.
 - Database: MySQL for reliable and persistent data storage

Key Achievements:

- Developed a functional, scalable, and modular web application.
- Implemented essential features such as:
 - Adding, editing, and deleting student records.
 - Displaying records in an intuitive tabular format.
- Ensured user-friendly navigation and data validation through Thymeleaf and Bootstrap.