

# **Web Application Development**

## **Lab 5: SERVLET & MVC PATTERN**

Name: Dang Thanh Lam – ITITDK23039

### **Part A: In class exercises**

Demonstrations:

#### **Model Layer**

1. Student JavaBean follows conventions
  - o The Student class encapsulates all student properties with private fields (id, studentCode, fullName, email, major, createdAt).
  - o It provides public no-arg and parameterized constructors.
  - o Getters and setters allow controlled access.
  - o `toString()` is overridden for meaningful object representation.
  - o This ensures encapsulation and JavaBean compliance.
2. StudentDAO has all CRUD methods
  - o DAO implements database operations:
    - `getAllStudents()` – fetches all students.
    - `getStudentById(int id)` – retrieves a single student.
    - `insertStudent(Student s)` – adds a new student.
    - `updateStudent(Student s)` – modifies an existing student.
    - `deleteStudent(int id)` – removes a student.
  - o Uses try-with-resources for safe database connection handling.
  - o Exceptions are caught and logged, ensuring robust error handling.

3. Database operations work correctly
  - All CRUD operations interact with MySQL database successfully.
  - Test sequence demonstrates creation, retrieval, update, and deletion of student records.
  - Timestamps (createdAt) are automatically recorded for new students.

## Controller Layer

1. Servlet properly annotated
  - StudentController is annotated with @WebServlet("/student"), making it accessible at the specified URL.
2. Routes requests correctly
  - doGet() routes based on the action parameter (list, new, edit, delete).
  - doPost() handles insert and update actions.
3. Calls DAO methods
  - Controller uses StudentDAO to perform all data operations, keeping business logic separate from presentation.
4. Sets request attributes
  - Attributes like studentList and student are set before forwarding to JSP.
  - This allows JSP to access data through Expression Language (EL).
5. Forwards/redirects appropriately
  - GET actions like list, new, and edit are forwarded to JSP views.
  - POST actions like insert and update are redirected to /student?action=list to avoid form resubmission.

## **View Layer**

1. No scriptlets in JSP
  - o All JSP pages are scriptlet-free, using JSTL and EL to render dynamic content.
2. JSTL tags used correctly
  - o Tags like <c:forEach> iterate over student lists.
  - o Conditional rendering uses <c:if> to display messages or empty states.
3. EL for all data access
  - o All dynamic content is accessed via \${studentList} or \${student.field}.
  - o This separates presentation from logic, adhering to MVC.
4. Single form for add/edit works
  - o student-form.jsp is used for both adding and editing students.
  - o Form fields are pre-filled when editing.
  - o Action changes dynamically (insert or update) based on context.

## **Functionality**

1. **List students**
  - o Navigating to /student displays all students in a table.
  - o If no students exist, shows "No students found."

Student Management System					
MVC Pattern with Jakarta EE & JSTL					
<a href="#" style="color: inherit; text-decoration: none;">+ Add New Student</a>					
ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
6	<b>ITIT1234</b>	Nguyen Van A	student1234@gmail.com	Computer Science	<a href="#"> Edit</a> <a href="#"> Delete</a>
5	<b>SV005</b>	David Wilson	david.w@email.com	Computer Science	<a href="#"> Edit</a> <a href="#"> Delete</a>
4	<b>SV004</b>	Sarah Davis	sarah.d@email.com	Data Science	<a href="#"> Edit</a> <a href="#"> Delete</a>
3	<b>SV003</b>	Michael Brown	michael.b@email.com	Software Engineering	<a href="#"> Edit</a> <a href="#"> Delete</a>
2	<b>SV002</b>	Emily Johnson	emily.j@email.com	Information Technology	<a href="#"> Edit</a> <a href="#"> Delete</a>
1	<b>SV001</b>	John Smith	john.smith@email.com	Computer Science	<a href="#"> Edit</a> <a href="#"> Delete</a>

## 2. Add student

- Clicking "Add New Student" opens the form.
- Submitting inserts a new record and redirects to the list.
- Success message is displayed.

+ **Add New Student**

Student Code \*
   

Format: 2 letters + 3+ digits

Full Name \*

Email \*

Major \*

[Add Student](#)

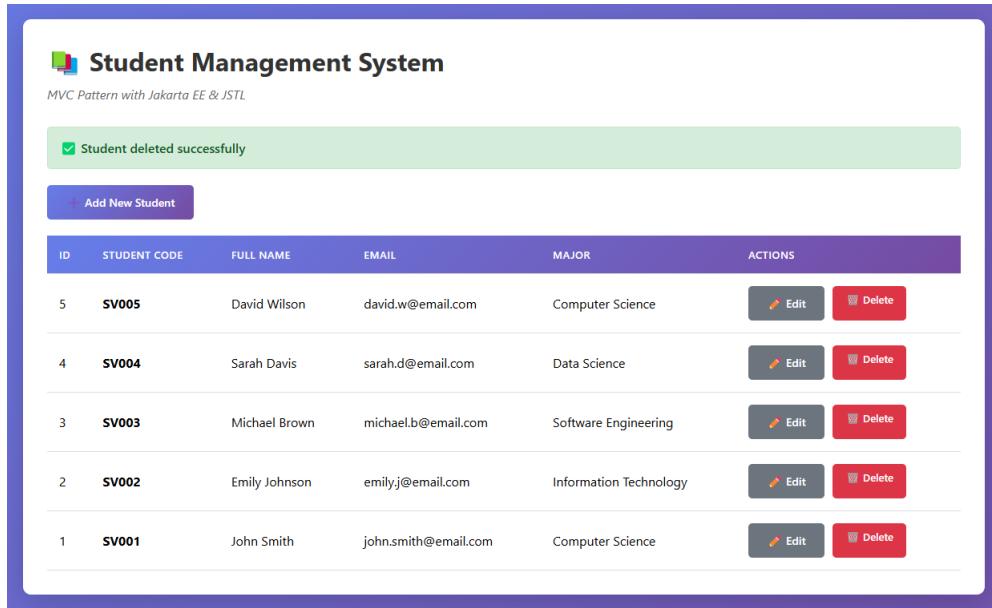
[Cancel](#)

## 3. Edit student

- Clicking "Edit" on a student pre-fills the form.
- Modifying and submitting updates the record.
- Update message is displayed.

#### 4. Delete student

- Clicking "Delete" removes the record.
- Redirects to the list with a deletion message.
- Handles empty state gracefully when all students are removed.



The screenshot shows a web application titled "Student Management System" using the "MVC Pattern with Jakarta EE & JSTL". The main content area displays a table of student records with columns: ID, STUDENT CODE, FULL NAME, EMAIL, and MAJOR. Each row has "Edit" and "Delete" buttons in the ACTIONS column. A green success message at the top states "✓ Student deleted successfully". A purple button labeled "+ Add New Student" is visible. The table data is as follows:

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
5	<b>SV005</b>	David Wilson	david.w@email.com	Computer Science	<button>Edit</button> <button>Delete</button>
4	<b>SV004</b>	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
3	<b>SV003</b>	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	<b>SV002</b>	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	<b>SV001</b>	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>

#### 5. Messages display

- Confirmation, success, and empty state messages are displayed using JSTL and EL.
- Ensures clear user feedback for all operations.