

Nguyễn Ngọc Đình Trung

ITITU20331

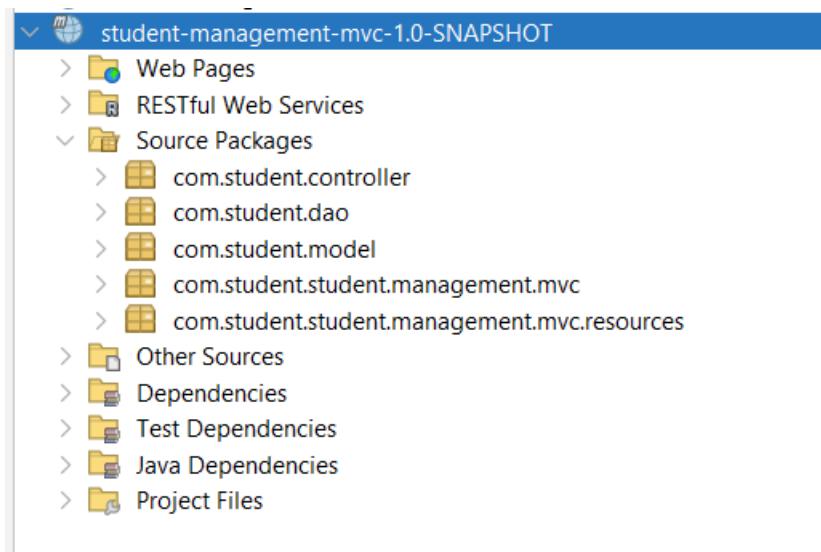
WEB APP REPORT

LAB 05

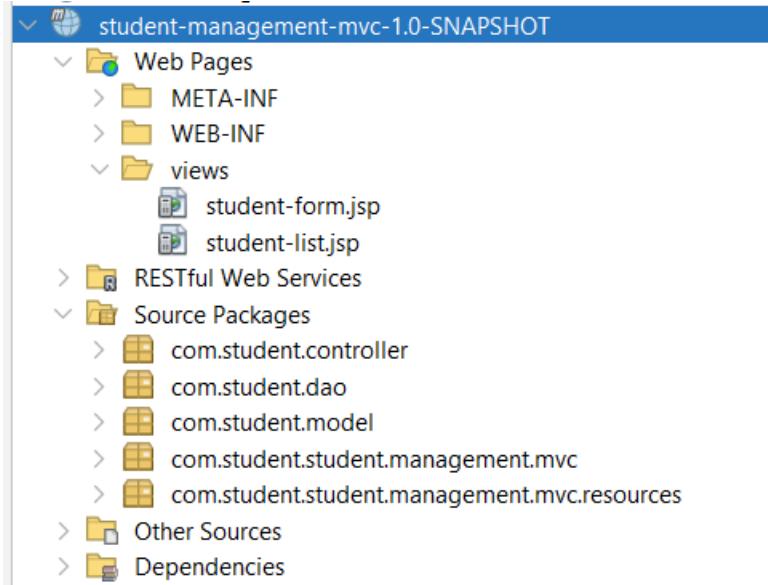
EX1:

TASK 1.1

CREATE PROJECT AND CREATE JAVA PACKAGE



CREATE VIEWS:



CONNECT MYSQL

```

CREATE DATABASE IF NOT EXISTS student_management;
USE student_management;

CREATE TABLE students (
    id INT PRIMARY KEY AUTO_INCREMENT,
    student_code VARCHAR(20) UNIQUE NOT NULL,
    full_name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    major VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert sample data
INSERT INTO students (student_code, full_name, email, major) VALUES
('SV001', 'Nguyen Van A', 'a.nguyen@example.com', 'Computer Science'),
('SV002', 'Tran Thi B', 'b.tran@example.com', 'Information Technology'),
('SV003', 'Le Van C', 'c.le@example.com', 'Software Engineering');

```

Action	Time	Message	Dur
CREATE TABLE students (id INT PRIMARY KEY AUTO_INCREMENT, student_code VARCHAR(20) UN...)	09:23:16	Error Code: 1050. Table 'students' already exists	0.0x
DROP DATABASE 'student_management'	09:23:27	1 row(s) affected	0.0'
CREATE DATABASE IF NOT EXISTS student_management	09:23:29	1 row(s) affected	0.0x
USE student_management	09:23:29	0 row(s) affected	0.0x
CREATE TABLE students (id INT PRIMARY KEY AUTO_INCREMENT, student_code VARCHAR(20) UN...)	09:23:29	0 row(s) affected	0.0x
INSERT INTO students (student_code, full_name, email, major) VALUES ('SV001', 'Nguyen Van A', 'a.nguyen@example.com', 'Computer Science')	09:23:29	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.0'

TASK 1.2

Create Student JavaBean

The screenshot shows a Java development environment with the following details:

- Project Structure:** The project is named "student-management-mvc-1.0-SNAPSHOT". It contains "Web Pages", "META-INF", "WEB-INF", "views" (with "student-form.jsp" and "student-list.jsp"), "RESTful Web Services", "Source Packages" (including "com.student.controller", "com.student.dao", and "com.student.model" which contains "Student.java"), "Other Sources", "Dependencies", "Test Dependencies", "Java Dependencies", and "Project Files".
- Code Editor:** The code editor displays the "Student.java" file. The code is as follows:

```

4  /*
5   * package com.student.model;
6
7  import java.sql.Timestamp;
8
9  public class Student {
10    private int id;
11    private String studentCode;
12    private String fullName;
13    private String email;
14    private String major;
15    private Timestamp createdAt;
16
17    // No-arg constructor (required for JavaBean)
18    public Student() {
19    }
20
21    // Constructor for creating new student (without ID)
22    public Student(String studentCode, String fullName, String email, String major) {
23        this.studentCode = studentCode;
24        this.fullName = fullName;
25        this.email = email;
26        this.major = major;
27    }
28
29    // Getters and Setters
30    public int getId() {
31        return id;
32    }
33
34    public void setId(int id) {
35        this.id = id;
36    }
37
38    public String getStudentCode() {
39        return studentCode;
40    }
41
42    public String getFullName() {
43        return fullName;
44    }
45
46    public String getEmail() {
47        return email;
48    }
49
50    public String getMajor() {
51        return major;
52    }
53
54    public String getCreatedAt(Timestamp createdAt) {
55        return createdAt.toString();
56    }
57}

```

- Navigator:** The Navigator panel shows the "Members" section for the "Student" class, listing all its methods and fields.
- Bottom Bar:** The bottom bar includes icons for file operations like Open, Save, Find, and others, along with a message: "Activate Windows Go to Settings to activate Wi-Fi".

- **CONSTRUCTORS:** Many frameworks require a no-arg constructor so they can instantiate the object and then set properties using setters.
This.studentCode = studentCode;
- This constructor helps you create an object quickly with its main properties.
- **GETTER AND SETTER:** This pattern is part of the JavaBeans convention: private fields + public getters/setters + no-arg constructor. It makes the object compatible with tools and frameworks that use reflection.
- **@Override:** means this method replaces the default

TASK 1.3

Create StudentDAO

The screenshot shows the NetBeans IDE interface. On the left is the Project Explorer with a tree view of the 'student-management-mvc-1.0-SNAPSHOT' project, including Web Pages, META-INF, WEB-INF, views, RESTful Web Services, Source Packages, com.student.controller, com.student.dao, and com.student.model. The 'StudentDAO.java' file is selected in the Source Packages section. On the right is the code editor window displaying the Java code for StudentDAO. Below the code editor is the Navigator pane, which lists the members of the StudentDAO class, such as addStudent(Student student), deleteStudent(int id), getAllStudents(), getConnection(), getStudentById(int id), updateStudent(Student student), and constants DB_PASSWORD, DB_URL, and DB_USER. At the bottom right of the screen, there is a message: 'Activate Windows Go to Settings to activate Windows'.

```

2 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4 */
5 package com.student.dao;
6
7 import com.student.model.Student;
8 import java.sql.*;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 public class StudentDAO {
13
14     // Database configuration
15     private static final String DB_URL = "jdbc:mysql://localhost:3306/student_management";
16     private static final String DB_USER = "root";
17     private static final String DB_PASSWORD = "123456";
18
19     // Get database connection
20     private Connection getConnection() throws SQLException {
21         try {
22             Class.forName("com.mysql.cj.jdbc.Driver");
23             return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
24         } catch (ClassNotFoundException e) {
25             throw new SQLException("MySQL Driver not found", e);
26         }
27     }
28
29     // Get all students
30     public List<Student> getAllStudents() {
31         List<Student> students = new ArrayList<>();
32         String sql = "SELECT * FROM students ORDER BY id DESC";
33
34         try (Connection conn = getConnection();
35             Statement stmt = conn.createStatement();
36             ResultSet rs = stmt.executeQuery(sql)) {

```

DAO.Java: include connect dtb (MySQL)

Database Configuration: The class defines constants for database connection details (DB_URL, DB_USER, DB_PASSWORD). These values are used to establish connections with the MySQL database.

- Class.forName(): to load driver
- DriverManager.getConnection(): to connect dtb (MySQL)

CRUD Operations: The class provides full CRUD functionality (Create, Read, Update, Delete) for the Student model.

The screenshot shows a web browser window with the URL 'localhost:8080/student-management-mvc/student'. The page title is 'Student Management System' and it displays 'MVC Pattern with Jakarta EE & JSTL'. The main content is a table listing student data:

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
3	SV003	Le Van C	c.le@example.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	<button>Edit</button> <button>Delete</button>

EX2:

TASK 2.1

Create Basic Servlet

The screenshot shows the NetBeans IDE interface with the following details:

- Project Structure:** The left pane shows a tree view of the project "student-management-mvc-1.0-SNAPSHOT". It includes Web Pages, META-INF, WEB-INF, Views (with student-form.jsp and student-list.jsp), RESTful Web Services, Source Packages (com.student.controller, com.student.dao, com.student.model), and com.student.student.management.mvc resources.
- Code Editor:** The right pane displays the code for `StudentController.java`. The code implements a HttpServlet and overrides methods like init(), doGet(), and doPost(). It uses Jakarta Servlet API annotations and imports StudentDAO, Student, RequestDispatcher, ServletException, and HttpServletRequest/HttpServletResponse from jakarta.servlet.http.
- Navigator:** Below the code editor, the Navigator pane shows the members of the `StudentController` class, including various HTTP methods (doGet, doPost, etc.) and generic methods (insert, update, delete).
- Status Bar:** At the bottom right, there are status messages: "Activate Windows" and "Go to Settings to activate Windows".

TASK 2.2

Add More CRUD Methods

- `showNewForm()` forwards to form: method correctly forwards the user to the JSP form used for creating a new student.
- `showEditForm()` loads student and forwards: method loads the specific student by ID and sends it to the form for editing.
- `doPost()` routes correctly: handles form submissions and routes by action:

```
switch (action) {    case "insert": insertStudent(...); break;    case "update": updateStudent(...); break;}
```
- `insertStudent()` creates student: The method accepts form fields (studentCode, fullName, email, major), builds a new Student object, and sends it to `studentDAO.addStudent()`.
- `updateStudent()` modifies student: retrieves updated form data, sets the ID, and forwards it to DAO for update
- `deleteStudent()` removes student: The method deletes a student based on ID

EX3:

TASK 3.1

Create Student List View

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (left):** Shows the project structure:
 - StudentManagement-1.0-SNAPSHOT (main)
 - StudentManagementMVC-1.0-SNAPSHOT
 - student-management-mvc-1.0-SNAPSHOT
 - Web Pages
 - META-INF
 - WEB-INF
 - views
 - student-form.jsp
 - student-list.jsp
 - RESTful Web Services
 - Source Packages
 - com.student.controller
 - StudentController.java
 - com.student.dao
 - StudentDAO.java
 - com.student.model
 - Student.java
 - com.student.student.management.mvc
 - com.student.student.management.mvc.resources
 - Other Sources
 - Dependencies
 - Test Dependencies
- Navigator (bottom-left):** Shows the selected file: student-list.jsp.
- Editor (right):** Displays the content of student-list.jsp:

```
<div class="container">
    <h1>Student Management System</h1>
    <p class="subtitle">MVC Pattern with Jakarta EE & JSTL</p>

    <!-- Success Message -->
    <c:if test="${not empty param.message}">
        <div class="message success">
            ${param.message}
        </div>
    </c:if>

    <!-- Error Message -->
    <c:if test="${not empty param.error}">
        <div class="message error">
            ${param.error}
        </div>
    </c:if>

    <!-- Add New Student Button -->
    <div style="margin-bottom: 20px;">
        <a href="#studentAction=new" class="btn btn-primary">
            + Add New Student
        </a>
    </div>

    <!-- Student Table -->
    <c:choose>
        <c:when test="${not empty students}">
            <table>
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Student Code</th>
                        <th>Full Name</th>
                        <th>Email</th>
                        <th>Major</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>1</td>
                        <td>S001</td>
                        <td>John Doe</td>
                        <td>johndoe@example.com</td>
                        <td>Computer Science</td>
                    </tr>
                </tbody>
            </table>
        <c:otherwise>
            <div>
                <h3>No Students Found</h3>
                <p>Please add new students or search for existing ones.</p>
            </div>
        <c:otherwise>
            <div>
                <h3>No Students Found</h3>
                <p>Please add new students or search for existing ones.</p>
            </div>
        </c:otherwise>
    </c:choose>
</div>
```
- Bottom Status Bar:** Shows "Activate Windows" and "Go to Settings to activate Windows".

- Add JSTL taglib directive
=> This directive enables the use of JSTL Core tags in the JSP page.
 - No scriptlets (no <% %>)
=> which are JSP scriptlets (embedding Java code directly into JSP).
 - Use <c:if> for messages
=> conditionally displays HTML content. Use for showing success message when param.message exists or param.error exists
 - Use <c:forEach> for student list
=> loops through the list of students provided by the controller.
 - Use EL \${ } for all data access
=> is used to print each field. Used to access: request parameters, request attributes and object properties
 - Handle empty list case
=> works like a Java if-else statement.

TASK 3.2

- Single form for both Add and Edit
 - => Instead of having two separate JSP files (student-add.jsp and student-edit.jsp), the page dynamically adjusts based on whether a Student object is present in the request.
 - Use <c:choose> for dynamic title
 - => works like a switch or if-else
 - Use <c:if> to show/hide fields conditionally
 - => conditionally includes HTML elements

- Pre-fill values for edit mode
- => This improves user experience and reduces input errors
- No scriptlets
=> The form avoids any Java code (<% %>)

EX4:

TASK 4.1

```
// Add new student
public boolean addstudent(Student student) {
    String sql = "INSERT INTO students (student_code, full_name, email, major) VALUES (?, ?, ?, ?)";

    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, student.getStudentCode());
        pstmt.setString(2, student.getFullName());
        pstmt.setString(3, student.getEmail());
        pstmt.setString(4, student.getMajor());

        int rowsAffected = pstmt.executeUpdate();
        return rowsAffected > 0;

    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}

// Update student
public boolean updatestudent(Student student) {
    String sql = "UPDATE students SET student_code = ?, full_name = ?, email = ?, major = ? WHERE id = ?";

    try (Connection conn = getConnection();
         PreparedStatement pstmt = conn.prepareStatement(sql)) {

        pstmt.setString(1, student.getStudentCode());
        pstmt.setString(2, student.getFullName());
        pstmt.setString(3, student.getEmail());
        pstmt.setString(4, student.getMajor());
        pstmt.setInt(5, student.getId());
        int rowsAffected = pstmt.executeUpdate();
        return rowsAffected > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

Activate Windows

www.microsoft.com/activation

TASK 4.2

TEST:

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Student List - MVC". The URL in the address bar is "localhost:8080/student-management-mvc/student". Below the address bar, there is a toolbar with various icons and links. The main content area displays a "Student Management System" page. At the top, it says "MVC Pattern with Jakarta EE & JSTL". A button labeled "+ Add New Student" is visible. Below is a table with student data:

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
3	SV003	Le Van C	c.le@example.com	Software Engineering	Edit Delete
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

Add: Click "Add New Student"

Upload Assignment: Lab 05 (Sai... | LAB 5 EXERCISES: SERVLET & M... | Add New Student | Student class explanation

localhost:8080/student-management-mvc/student?action=new

LSD - Genius | Musi... E Library - Google D... Discord ERDPlus Snow White - Listen... Thu - Nguyen Ngoc...

+ Add New Student

Student Code *

e.g., SV001, IT123

Format: 2 letters + 3+ digits

Full Name *

Enter full name

Email *

student@example.com

Major *

-- Select Major --

+ Add Student **✗ Cancel**

Activate Windows
Go to Settings to activate Windows.

Upload Assignment: Lab 05 (Sai... | LAB 5 EXERCISES: SERVLET & M... | Student List - MVC | Student class explanation

localhost:8080/student-management-mvc/student?action=list&message=Student%20added%20successfully

LSD - Genius | Musi... E Library - Google D... Discord ERDPlus Snow White - Listen... Thu - Nguyen Ngoc...

Student Management System

MVC Pattern with Jakarta EE & JSTL

✓ Student added successfully

Add New Student

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
4	SV20331	DINH TRUNG	TRUNG20331@gmail.com	Computer Science	Edit Delete
3	SV003	Le Van C	c.le@example.com	Software Engineering	Edit Delete
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

Activate Windows
Go to Settings to activate Windows.

Edit: Click "Edit" on test student

Upload Assignment: Lab 05 (Sa... | LAB 5 EXERCISES: SERVLET & M... | Edit Student | Student class explanation

localhost:8080/student-management-mvc/student?action=edit&id=4

LSD - Genius | Musi... E Library - Google D... Discord ERDPlus Snow White - Listen... Thu - Nguyen Ngoc...

Edit Student

Student Code *

Format: 2 letters + 3+ digits

Full Name *

Email *

Major *

Update Student

Cancel

Activate Windows
Go to Settings to activate Windows.

Upload Assignment: Lab 05 (Sa... | LAB 5 EXERCISES: SERVLET & M... | Student List - MVC | Student class explanation

localhost:8080/student-management-mvc/student?action=list&message=Student%20updated%20successfully

LSD - Genius | Musi... E Library - Google D... Discord ERDPlus Snow White - Listen... Thu - Nguyen Ngoc...

Student Management System

MVC Pattern with Jakarta EE & JSTL

Student updated successfully

Add New Student

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
4	SV20331	NGUYEN NGOC DINH TRUNG	DINHTRUNG20331@gmail.com	Computer Science	Edit Delete
3	SV003	Le Van C	c.le@example.com	Software Engineering	Edit Delete
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	Edit Delete
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	Edit Delete

Activate Windows
Go to Settings to activate Windows.

Delete: Click "Delete" on test student

A screenshot of a web browser window titled "Student Management". The URL is "localhost:8080/student-management-mvc/student?action=list&message=Student%20updated%20successfully". A dark overlay dialog box is centered, containing the text "localhost:8080 says" and "Are you sure you want to delete this student?". Below the dialog, a green success message box shows "Student updated successfully". The main table has one row removed, showing IDs 3, 2, and 1 with student names Le Van C, Tran Thi B, and Nguyen Van A respectively.

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
4	SV20331	NGUYEN NGOC DINH TRUNG	DINHTRUNG20331@gmail.com	Computer Science	<button>Edit</button> <button>Delete</button>
3	SV003	Le Van C	c.le@example.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	<button>Edit</button> <button>Delete</button>

A screenshot of a web browser window titled "Student Management". The URL is "localhost:8080/student-management-mvc/student?action=list&message=Student%20deleted%20successfully". A dark overlay dialog box is centered, containing the text "localhost:8080 says" and "Are you sure you want to delete this student?". Below the dialog, a green success message box shows "Student deleted successfully". The main table now only contains three rows: ID 3 (Le Van C), ID 2 (Tran Thi B), and ID 1 (Nguyen Van A).

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
3	SV003	Le Van C	c.le@example.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	SV002	Tran Thi B	b.tran@example.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	SV001	Nguyen Van A	a.nguyen@example.com	Computer Science	<button>Edit</button> <button>Delete</button>

Empty State: Delete all students

