

Lab #5

SERVLET & MVC PATTERN

EXERCISE 5: SEARCH FUNCTIONALITY

5.1: Update StudentDAO

Think about:

- How will you reuse the code from `getAllStudents()` to build Student objects?
 - I reuse exact same block of code to populate a Student object:

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

    pstmt.setString(1, searchPattern);
    pstmt.setString(2, searchPattern);
    pstmt.setString(3, searchPattern);

    try (ResultSet rs = pstmt.executeQuery()) {
        while (rs.next()) {
            Student student = new Student();
            student.setId(rs.getInt("id"));
            student.setStudentCode(rs.getString("student_code"));
            student.setFullName(rs.getString("full_name"));
            student.setEmail(rs.getString("email"));
            student.setMajor(rs.getString("major"));
            student.setCreatedAt(rs.getTimestamp("created_at"));
            students.add(student);
        }
    }

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

return students;
}
```

- What happens if the keyword is null or empty?
 - If the keyword is null: The database searches for the literal text "null". Unless a student named "null", it will return 0 results.

```
Found 0 students:
```

- If the keyword is empty "": The % wild card matches every single record in the database. Therefore, the result usually shows all records.

```
Found 5 students:  
Miemie Love - tmy00037@gmail.com  
Sarah Davis - sarah.d@email.com  
Michael Brown - michael.b@email.com  
Emily Johnson - emily.j@email.com  
John Smith - john.smith@email.com
```

- Should you validate the keyword before building the SQL?

Yes, absolutely.

Testing:

The screenshot shows an IDE interface with a code editor and a terminal window. The code in the editor is:

```
public static void main(String[] args) {  
    StudentDAO dao = new StudentDAO();  
    List<Student> results = dao.searchStudents("john");  
  
    System.out.println("Found " + results.size() + " students:");  
    for (Student s : results) {  
        System.out.println(s.getFullName() + " - " + s.getEmail());  
    }  
}
```

The terminal window shows the output of the code execution:

```
om.student.dao.StudentDAO > searchStudents >  
t - Run (StudentDAO) x  
  
Found 2 students:  
Emily Johnson - emily.j@email.com  
John Smith - john.smith@email.com  
  
BUILD SUCCESS
```

1. Home page with student list

The screenshot shows a web browser window titled "Student List - MVC". The URL is "localhost:8080/student-management-mvc/student". The page is titled "Student Management System" and includes a subtitle "MVC Pattern with Jakarta EE & JSTL". At the top, there is a purple button labeled "Add New Student" and a search bar with placeholder text "Search by Name, Email, or Code...". A teal "Search" button is located to the right of the search bar. Below the header is a table with the following data:

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
8	SV009	Miemiemim	tmy@gmail.com	Software Engineering	<button>Edit</button> <button>Delete</button>
6	SV006	Miemie Love	tmy00037@gmail.com	Computer Science	<button>Edit</button> <button>Delete</button>
4	SV004	Sarah Davis	sarah.d@email.com	Data Science	<button>Edit</button> <button>Delete</button>
3	SV003	Michael Brown	michael.b@email.com	Software Engineering	<button>Edit</button> <button>Delete</button>
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	SV001	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>

2. Add student form

The screenshot shows a web browser window titled "Add New Student". The URL is "localhost:8080/student-management-mvc/student?action=new". The page has a purple header with the title "Add New Student". The main content area contains a form with the following fields:

- Student Code *: Input field containing "e.g., SV001, IT123". Below it is a note: "Format: 2 letters + 3+ digits".
- Full Name *: Input field containing "Enter full name".
- Email *: Input field containing "student@example.com".
- Major *: A dropdown menu with the placeholder "Select Major".

At the bottom of the form are two buttons: a purple "Add Student" button and a grey "Cancel" button.

3. Edit student form (pre-filled)

Add New Student

Student Code *

Format: 2 letters + 3+ digits

Full Name *

Email *

Major *

4. Search results

Add New Student

John

Search

Student Management System

MVC Pattern with Jakarta EE & JSTL

ID	STUDENT CODE	FULL NAME	EMAIL	MAJOR	ACTIONS
2	SV002	Emily Johnson	emily.j@email.com	Information Technology	<button>Edit</button> <button>Delete</button>
1	SV001	John Smith	john.smith@email.com	Computer Science	<button>Edit</button> <button>Delete</button>

5. Validation errors

The screenshot shows a web browser window titled "Edit Student" with the URL "localhost:8080/student-management-mvc/student". A modal dialog box is open, titled "+ Add New Student". The form contains four fields: "Student Code *", "Full Name *", "Email *", and "Major *". Each field has an error message below it:

- "Student Code *": "Format: 2 letters + 3+ digits" and "Invalid format. Use 2 Uppercase + 3 Digits (e.g., SV001)"
- "Full Name *": "Name must be at least 2 characters"
- "Email *": "singer@gmail.com"
- "Major *": "-- Select Major --" and "Major is required"

At the bottom of the modal are two buttons: "Add Student" and "Cancel".