Exercise: Enhancing the Student Grades Application

Duration: 50min| Google searches allowed on small unkown topics | AI code not allowed

Objective: In this exercise, you will extend the functionality of the existing Spring Boot application for managing student grades. The goal is to practice adding new features, server-side form validation, and modularizing the application further by introducing a new model and controller.

1. Implement Deletion of Student Grades

- Add a "Delete" button next to the "Update" button in the grades list view.
- This button should trigger a deletion operation when pressed.
- Update the controller to handle the deletion of a grade from the ArrayList datastore.
- Ensure that after deletion, the user is redirected back to the list view with the updated list of grades.

Hint:

- Consider using path variables or request parameters to identify which grade to delete.

2. Add Form Validation

- Implement server-side form validation using eg. Jakarta Validation (jakarta.validation).
- Add annotations to the Grade model to enforce validation rules:
  - studentName: Must not be empty and should have a minimum length of 3 characters.
  - subject: Must not be empty.
  - score: Must be an integer between 0 and 20 (inclusive).

- Modify the controller and HTML templates to handle and display validation errors.

```
public String submitGrade(@Valid Grade grade, BindingResult result...
    if (result.hasErrors()) {
        ...
        return "form";
```

Hint: Use annotations such as @Valid, @NotBlank and @Min/@Max to validate these fields. Review how these annotations enforce rules and how to display errors in Thymeleaf templates.

Look at the example from the "Global Store" challenge

@NotBlank(message = "Name cannot be blank")

3. Modularize the Application by Adding a New Model and Controller

- **Add a Student Model:**
  - Create a new model class called Student with fields such as studentId, firstName, and lastName.
  - Update the Grade model to include a reference to Student by using a studentId to link the two.
  - Create a dropdown for selecting students when creating or updating a grade. Populate this dropdown dynamically based on data stored in the ArrayList (datastore).

Hints:

- Think about how to store a studentId in the Grade class. Consider how you might retrieve student details using this ID without making the models tightly coupled.
- Create a StudentController:
  - Add a controller to manage student-related operations (e.g., adding and listing students).

- Refactor the Application:
  - Update the existing controller to accommodate the new association and update the HTML templates to reflect these changes.
  - GradeController.java adding students:

```
// access students
StudentService studentService = new StudentService();

// Add students to the model for the dropdown
model.addAttribute("students", studentService.getStudents());
...
```

- form.java adding students Dropdown:

```
<select id="student" name="studentId" th:field="*{studentId}">
    <option value="" disabled selected>Select a student</option>
    <option th:each="student : ${students}"
th:value="${student.studentId}"
            th:text...
```