Department Mathematik/Informatik, Abteilung Informatik
Software & System Engineering Group
Software Technology Group
Lecture Object-Oriented Software Engineering, SS 2024

Alexander Korn, Adrian Bajraktari,
Prof. Dr. Michael Felderer

Project Sheet 4. Fourth Sprint

Hand out: 28.06.2024, 18:00
Hand in: 05.07.2024, 14:00

Exercise 1. This Sprint's Work

This week, we want you to add functionality for creating students.

a) Begin by merging the updated template from `https://github.com/lxndio/oose-24-project` into your project. Review sheet 3 for information on how to do that.

b) Now, functionality should be added that allows creating new students. To accomplish this, the following tasks must be completed:

- Add a GET endpoint at `/student/new` to the student controller. This endpoint should return the `edit_student` view. The corresponding model must include the following attributes:

Attribute name	Description
`student`	A new student object that should be filled with data inside the view.
`page_type`	The view can be used both for creating new students and for editing existing students. As this requires minor changes to the view, a page type (`new` or `edit`) must be specified.
`study_subjects`	A list of strings containing the study subjects available at the university. This can be derived from the institutes that provide different study subjects.

- Add a POST endpoint at `/student/new` to the student controller. The view calls this endpoint when a user filled in data for a new student. Thus, this endpoint must receive the filled student object as a model attribute.

> **Hint:** Use the `@ModelAttribute` annotation for a parameter of the endpoint function to receive a student object from the view calling the endpoint.

- In the endpoint, the given student object should be validated using a `StudentValidator` (see next step). If the student object is valid, it should be saved in the student repository. The endpoint should return the `edit_student` endpoint with a model including all attributes as defined above. Additionally, the following attributes should be added to the view model:

Attribute name	Description
`message_type`	The view can display a message to inform the user about the status of an operation. This can either be `success` or `error`, resulting in a green or red message box.
`message`	The status message that the view should display. The user should be informed about whether their operation was successful or not.

- Create a `StudentValidator` class. This class should be initialized with a reference to the student, and institute repositories. The class should provide a `validateStudent` method that, given a student object, should validate the student's data. Among other things, the validator should check:
  - Are all necessary fields filled?
  - Is the student's e-mail address correct?
  - Does the matriculation number already exist?
  - Does the study subject exist?
- If a validity check fails, the student validator should return a `StudentValidationException`. Create a class for that exception, extending `Exception`. The exception should contain a sensible message describing why the validation has failed.
- Write unit tests to make sure that the validation functions for e-mail addresses etc. are correct.

> **Hint:** Unit tests should be written using JUnit 5. Take a look at the reference to learn how to write tests: https://junit.org/junit5/docs/current/user-guide/.

[6 points]

$\sum$ **6.0 points**