Assignment: RESTful Service in Spring boot

In this assignment you will learn how to implement REST CRUD endpoints in a Spring Boot web app.

Difficulty: ☆☆☆

Learning objectives:

- I can implement CRUD endpoints for RESTful operations in Spring Boot
- I can apply a fake data-store to retrieve data
- I can validate the input of a REST endpoint

Estimated time required: 100 minutes

Step 1: Download the startup project

Download the startup Java project and open it in IntelliJ (or another IDE). This project contains one Gradle module: school. This is a Spring boot project partially implemented.

Field Code Changed

It has all the CRUD operations for resource "students", but not for "countries". Your job will be to implement all the remaining operations for it.

The project uses fake persistence classes to save/retrieve data. The intention is to move to a real database very soon.

Step 2: Implementation of *StudentsController*

Take a brief look at the source code of the project, starting from *StudentsController*.

The layering approach is very similar to the previous Dependency Injection assignment one. A difference is that the persistence layer has an internal representation of data called "entity" (from JPA entity) and it doesn't use the domain layer at all. It's a design decision for avoiding that the data persistence format gets mixed-up with the domain model. As a result, changes in the database tables shouldn't impact directly (and unexpectedly) the domain classes. This will make more sense when these classes are mapped into database tables as JPA entities, but anyways the separation is already there.

Field Code Changed

Note: Layering like any architectural decision is based on trade-offs and there's always pros and cons attached to each decision. We don't want to oversimplify or overengineer our solutions. With time it gets easier to find the right balance and adapt to changes when needed.

Field Code Changed

The students API is designed as shown in the table below. These CRUD operations are implemented in StudentsController class and are at the maturity level 2. Take a good look at this class, for each of the operations in the table below there is one method in the StudentsController class.

Table 1. Students service API: "students" endpoint.

URL	resource	operation	
/students/123	Students	GET	read the student with id 123
/students	Students	GET	read a list with all students
/students?country=BG	Students	GET	read a list with all students from a given country ps: here we are doing the filtering
/students/456	Students	DELETE	delete the student with id 456

/students	Students	POST	crate a new student
/students	Students	PUT	update or create the student (student as JSON)
/students/789	Students	PUT	update or create the student with id 789 (form parameters)

Make sure you run the app and test *StudentsController* via a http client (like Postman).

Field Code Changed

Step 3: Completing *CountriesController*

There's also a controller for countries but it's not fully implemented yet: *CountriesController*.

Follow the code conventions in the project in order to implement the following services:

1. Create Country
    - country code should be unique
2. Get Country
3. Update Country
4. Delete Country

Test each new service implemented with your favorite http client.

General tips when making a service

- Make sure that CRUD operations are properly mapped to HTTP operations with the expected URLS (i.e., Richardson Maturity level 2):
    - Create = POST
    - Read = GET
    - Update = PUT
    - Delete = DELETE
- Always return good response code/type which is compliant with the http Status Code Definitions (see https://www.w3.org/Protocols/rfc2616/rfc2616.html).
- JSON should be the preferred data format
- When the operation is executed successfully:
    - GET returns 200 OK
    - DELETE and PUT return 204 NO_CONTENT, and
    - POST returns 201 CREATED, with the id of the just created resource
- If the service cannot process client's request due to an error in the client's request, return 400 BAD_REQUEST.
- If client's request is good, but there has been an internal service error (e.g., an Exception), return 500 INTERNAL_SERVER_ERROR.

Field Code Changed

When completed

You will discuss the results of this assignment plenary with your teacher. If you are done, start with assignment applying the knowledge in the individual track. In this case, creating at least 3 RESTful endpoints for your application. However, don't hold yourself back. In a later stage you will need multiple operations for a successful application, so it never hurts to already create some more.