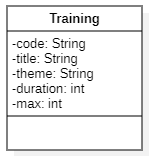
Exercises JPA

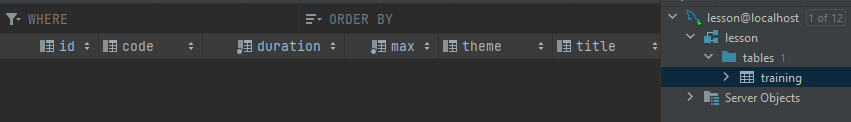
**Exercise 1: Trainings**

1. Download startproject.zip, unzip this project, rename it and put it in your corresponding GitHub-folder and add the following class (in a package called model) according to the UML-diagram below

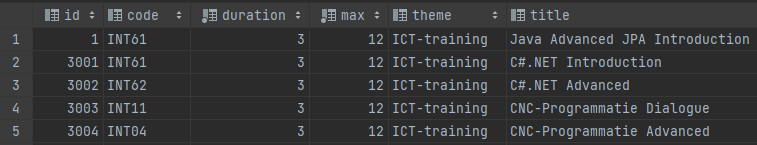


Make sure you create the no-arg constructor and all getters and setters

1. If you programmed exactly what is in the class diagram and you run your application, the corresponding table should exist in your database (check this in your Database tab).



1. Now run the SQL script, *script trainings.sql,* to fill the corresponding table. Check this in your database tab. The table should be filled with 50 records:



1. The statement "spring.jpa.hibernate.ddl-auto=create-drop" in your properties file will cause the database to be deleted after closing your program. So you will have to refill the database with the script “*script trainings.sql”* every time you run your program.
2. In the new project, create a welcome screen (index.html) with 2 buttons and 1 link:

Afbeelding met tekst

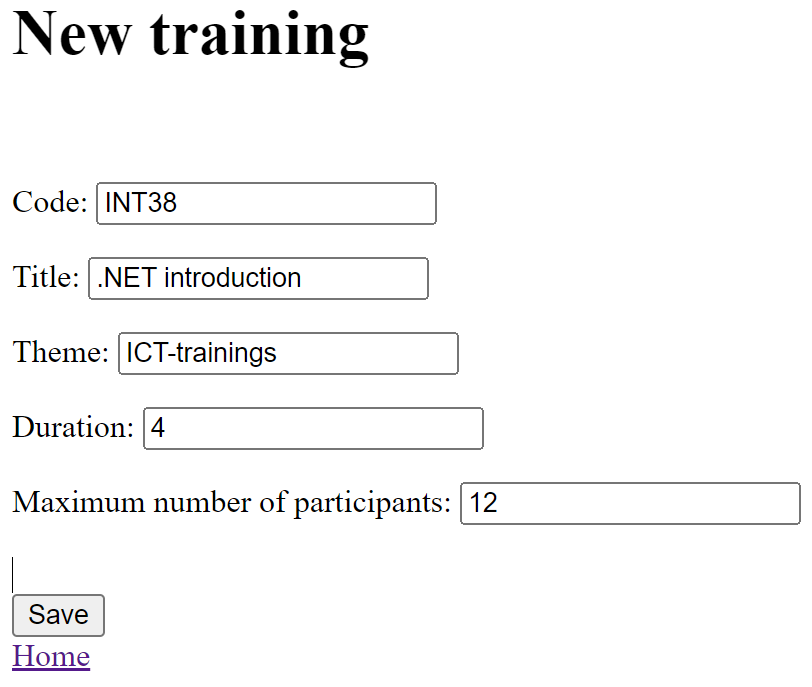
Automatisch gegenereerde beschrijving

1. Then create the corresponding controller class and repository class with the necessary methods to offer the following functionalities:
   1. The Admin link takes you to a screen which shows you all the trainings that are available (Code, Title and Theme) and gives you the possibility to change, remove or add one. See screen below for a preview...

Afbeelding met tekst

Automatisch gegenereerde beschrijving

* 1. The link “new” takes you to a page on which you can add a training with the following values
     + INT38
     + .NET introduction
     + ICT-trainings
     + 4
     + 12



The result should then look something like this:



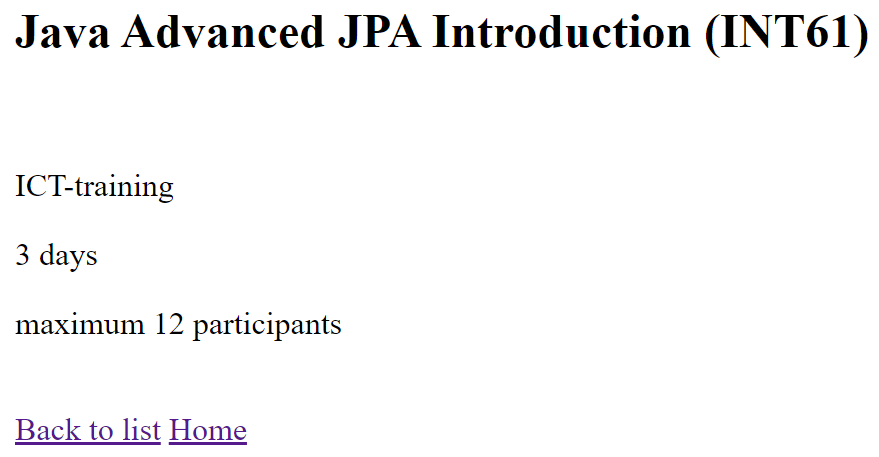
* 1. Also make sure that you can delete the training via the "delete" link and that you can also edit the training via the “edit” link. Attention! After each edit, the user re-enters the list of trainings where the changes must also be visible.

1. If you click on "Training by theme" on the home page , you will again show the list of all trainings but now sorted by theme.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

1. Make sure this page has a hyperlink at the bottom to go back to index.html.
2. If you click on a hyperlink of the code of the training in the list, you will be taken to:

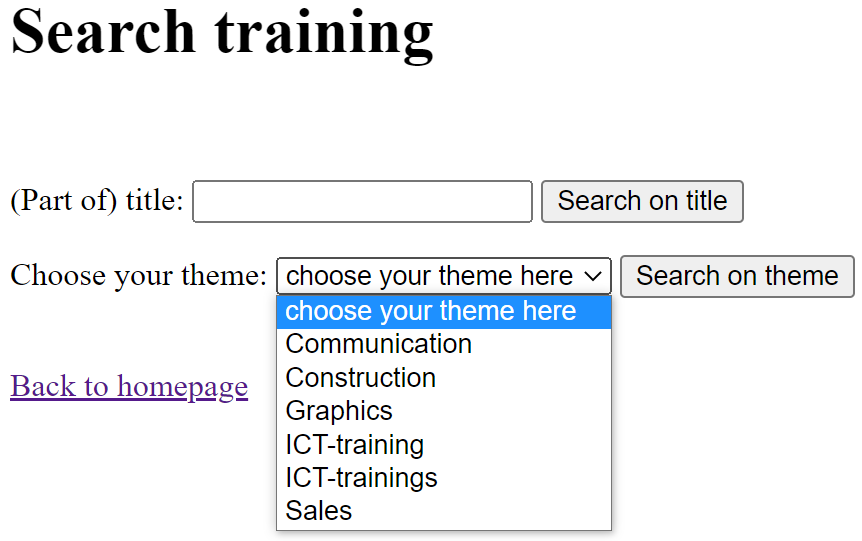


1. If you click on the 'Advanced search' button on the home page, you will be taken to

Afbeelding met tekst

Automatisch gegenereerde beschrijving

1. Each of these search options again shows the list of courses, but then taking into account the search criterion. In the drop-down list “choose your theme” you can see the **unique** themes that appear in the table "training":



For this you will not be able to use a standardized "keyword query" but you have to put together the right query yourself with JPQL...

For example, if you search by theme Communication:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

1. Also add a link at the bottom of the page that will take you back to the search page.

**Rest services**

Now create a TrainingRestController with:

* A service to retrieve all training courses from the database, sorted by theme

Path: /api/training

* A service to request one specific training based on its id

Path: /api /training/{id}

* A service to search for a training based on a (part of) the title. The part of the title is given in the body of the request. **Program this service using streams!**

Path: /api /training/title

* A service to create a new training in the database. This service returns the newly created object.

Path: /api /training

* A service to remove a training course from the database. This service must return the number of trainings that are still in the database (after the training has been removed). The id of the training that needs to be removed is specified in the path:

path: /api/training/{id}

If the id doesn’t exist you return a 404 Not Found-status.

* A service to change a training in the database. The id of the training that needs to be changed is given in the path:
  + Path: /api /training/{id}

In the body of the request, you provide the new values of the attributes. You then save the changes in the database and return the changed object. If the id doesn’t exist you return a 404 Not Found-status.

If your web service works perfectly and you adhered to the above paths, you can test your web service with the postman script that we have prepared for you.

To make this script work, you first make sure that the training table contains exactly the data that we expect in the test script.

When your training table is filled up with data, import the Postman test script "Test training.postman\_collection.json" that you can find on Canvas.

If everything went well, you will now see 36 successful tests (passed(36) failed(0)).

Attention! If you want to test again, you will have to run the sql script again as the test makes changes to your database.

