CA2_JPAMappingAndREST - Marek Furak, Smaranda Dungeanu, Cristian Nita

GITHUB LINK

You can fork our project from here.

DESIGN

For this project we used the DB Façade design. We created an interface for the Person interface so that it could be mocked for the testing and then we just implemented it in the PersonFacade. We use this class to communicate with the database through the EntityManager. In the beginning and for testing we have selected the Drop and Create method of the DB but after deployment we switched to None so the program would not create new fields or delete our tables. We created our own exception for NotFound which extends Exception class and it is called when we try to work with a person that is not in the DB.

TESTING

For the testing we used classes to test in isolation, mock objects and integration test on a test DB. For the isolation test we just tested the Person class to see if the methods we created worked. We created a mock object for the PersonFacade which allowed us to test the methods from the façade in isolation. Next we tested the PersonFacade itself on a local testing DB with the Drop and Create generation option. Our test results provided us with feedback and we were able to get the tests done.

Contribution to the CA

During this CA, we split the tasks equally between the team members. Therefore, Smaranda was responsible with the JPA part, the façade and exposing the façade via the Rest API, Cristi created the web interface, implemented the project web-site on our server, set up the server on Azure, implemented editing a person and remove a roleschool and Marek made the testing, while working with Smaranda on the façade.

Our strategy used to implement inheritance

For our project, we chose to use the JOINED inheritance strategy for our tables, especially because it is intuitive. Also, it is known that querying can affect the performance, but in our case we don't need to reassemble any instance of the subclasses, the only query that we use being for retrieving all the persons. It was also easy for us to use this strategy as it is close to what we know from OO-inheritance.

Extra

Since we made changes to the person façade interface by adding methods to edit a person and remove a role from a person we made the tests for that as well. Also we added tests for editing, getting, deleting a non existing person and removing a role from a non existing person.