Control práctico de DP1 – Grupo 2

To start the control check, you must accept the task of this practical control through the following link: https://classroom.github.com/a/DqDJnFDC

By accepting this task, an individual unique repository will be created for you, you must use this repository to perform the control check and solve the exercises.

You must submit the activity in EV associated to the control check by providing as text the URL address of your personal repository. Recall that you must also submit your solution to the control check.

Submission of your solution will be done by a single "git push" command to your individual repository. Remember to push before logging out of the computer and leaving the classroom, otherwise your attempt will be evaluated as unsubmitted.

Your first task in this control check will be to clone the repository (use the command "git clone XXXX" for this, where XXXX is the URL of your repository). Next, you will need to import the project into your favorite development environment and start the exercises listed in the document provided through the repository. When importing the project, it is possible that the project has compilation errors. Don't worry, these errors, if present, will disappear as you implement the exercises of the control check.

In this exercise, we will add the functionality to manage the products that will be sold in our pet clinic. To do this, we will perform a series of exercises based on features that we will implement in the system, and we will validate them through unit tests. If you want to see the results of the tests, you can execute the command ".\mvnw test" in the root folder of the project. Each successfully passed test will be worth one point.

Note 1: Do not modify the class names or signature (name, response type, and parameters) of the methods provided as source material for the control check. The tests used for evaluation depend on the classes and methods having the provided structure and names. If you modify them, you will probably not be able to make the tests pass, and you will get a bad grade.

Note 2: Do not modify the unit tests provided as part of the project under any circumstances. Even if you modify the tests in your local copy of the project, they will be restored via a git command prior to running the tests for the final grade, so your modifications to the tests will not be taken into account.

Test 1 – Creation of the Feeding Entity and its associated repository

Modify the "Feeding" class to be an entity. This class is placed in the "org.springframework.samples.petclinic.feeding" package and must have the following attributes and restrictions:

1. The integer attribute called "id" will act as the primary key in the relational database table associated with the entity.
2. The attribute of type date (LocalDate) called "startDate", which represents the moment in which the meal plan is started, will follow the format "yyyy/MM/dd" (you can use as an example the Pet class and your date of birth to specify this format) and will be stored in the database with the column name "start_date". This attribute must be required.
3. The double-precision floating-point numeric type attribute called "weeksDuration," which indicates the week-long duration of the meal plan, should be mandatory and allow only values greater than or equal to 1. In the database, it will be stored with the column name "weeks_duration".
4. It must have an N-1 relationship with the Pet class using an attribute called "pet". This relationship will indicate the pet to which an eating plan is assigned. Every feeding plan must be associated with a pet (the field is mandatory and cannot be null).

Modify the "FeedingRepository" interface hosted in the same package to extend to CrudRepository.

Test 2 – Creating the FeedingType Entity

Modify the "FeedingType" class hosted in the "org.springframework.samples.petclinic.feeding" package to be an entity. This entity must have the following attributes and constraints:

1. An integer attribute named "id" that acts as the primary key in the relational database table associated with the entity.
2. A String attribute named "name" that cannot be empty, of unique value (that is, there cannot be another FeedingType with the same name in the database), and whose minimum length is 5 characters and a maximum of 30.
3. A String attribute named "description". Empty descriptions are not allowed.
4. A N to 1 relationship with the PetType class (type of pet) that describes the type of pet to which that type of feeding should be applied. This value cannot be null. The column that represents this attribute in the database should be called "pet_type_id".

In addition, you must uncomment the "List<FeedingType> findAllFeedingTypes()" method in the FeedRepository repository and annotate it to run a query to get all existing product types.

Test 3 – Modify the database initialization script to include two feeding plans and two feeding types

Modify the database initialization script to create the following Feeding plans and FeedingType:

Feeding 1:

- id: 1
- start_date: 2022-01-05
- weeks_duration: 7.5
- pet_id: 7

Feeding 2:

- id: 2
- start_date: 2022-01-04
- weeks_duration: 6
- pet_id: 4

FeedingType 1:

- andd: 1
- name: High Protein Puppy Food
- description: Using a standard 8 oz/250 ml measuring cup which contains approximately 112 g of food: For a body weight of 3 - 12, feed with 1/2 to 2/3 cups until 3 months.
- pet_type_id: 2

FeedingType 2:

- id: 2
- name: Adult Weight Management
- description: Chicken and Rice Formula Dry Cat Food. Feed 1 can per 2-1/2 lbs of body weight daily. Adjust as needed. Divide into 2 or more meals.
- pet_type_id: 1

Test 4 - Create a one-way N-to-1 relationship from Feeding to FeedingType

Create a one-way N-to-1 relationship from "Feeding" to "FeedingType", using "feedingType" as the name of the corresponding attribute in the class. Modify the database initialization script so that *Feeding 1* is associated with *FeedingType 2* and *Feeding 2* is associated with *FeedingType 1.*

Test 5 – Creation of a feeding plan management service

Modify the "FeedingService" class, to be a Spring business logic service, and to get all registered Feeding plans (as a list) using the repository. Do not modify the implementation of the other methods of the service yet.

Test 6 – Annotate the repository to get the types of feeding plans (FeedingType) by name, and implement the associated method in the feeding plan management service.

Create a custom query that can be invoked through the feeding plan repository that gets one type of feeding plan by name. Use it to implement the "getFeedingType(String name)" method of the power plan management service.

Test 7– Creating a Formatter of product types

Implement the *formatter* methods for the FeedingType class (using the class called "FeedingTypeFormatter" that is already created in the same "feeding" package we are working with). The formatter must display the types of feeding plans using the string of his name, and must obtain a type of feeding plan given his name, looking for it in the database (for this it must use of the feeding plan management service and not the repository). Remember that if the formatter cannot get an appropriate value from the provided text, it must throw an exception of type "ParseException".

Test 8 – Registration of feeding plans and verification of consistency between the pet to which the plan is assigned and the type of pet associated with a type of feeding.

Implement the "save" method of the power plan management service. If the pet specified in the feeding plan is not of the type of pet associated with the corresponding type of feeding, an exception of type "UnfeasibleFeedingException" must be thrown (this class is already created in the feeding package). In addition, if the exception is thrown, the database should rollback the transaction associated with the save operation.

Test 9 – A form for creating/editing meal plans

A form is provided in the project for the creation of new feeding plans. This form allows you to specify the start date, the duration in weeks, the pet to which the feeding plan is assigned, and the type of associated feeding plan. The jsp page of the form has been created inside the jsps folder (webapp/WEB-INF/jsp), in a folder named "feedings", and the filename is "createOrUpdateFeedingForm.jps".

You must create a handler method that displays the form. It must be available through the url:

http://localhost:8080/feeding/create

The form must appear empty by default, and must contain a text field for the date, a field to specify the duration in weeks, two selectors to select the pet and the type of feeding plan and a button to send the data. The form data must be sent to the same address using the post method. You may need to add some method to the service of managing feeding plans and feeding types.

The associated controller must be created as a method of the "FeedingController" class in the same package, and it should pass an object of type Feeding with the name "feeding" through the model to the form. In addition, you must provide the following: the complete collection of feed type with the name "feedingTypes", and the complete collection of pets with the name "pets". To do the latter, you will need to use the PetService pet management service (an appropriate method has been created for this in that class). Recall that you must modify the security settings of the application allow access only administrator users (authority "admin").

Test 10 – Creation of the Controller for the creation of new power plans.

Create a controller method in the previous class that responds to POST requests at the url specified above. The method responsible for validating the data of the feeding plan, showing the errors found through the form -if present. If there are no errors, the method should save the new feeding plan using the feeding plan management service. Note that if the selected pet is not of a type consistent with the type of feeding plan chosen, the corresponding exception should be captured, and the form should show the following error message in the feeding plan field: "The selected pet cannot be assigned the specified feeding plan". After saving the feeding plan, if successful, a redirect will be used to return to the application home page (welcome). Remember that the input format of the start date of the meal plan must be "yyyy/MM/dd", otherwise the tests will not pass.