Control práctico de DP1 – Grupo 2

Enunciado

En este ejercicio, añadiremos la funcionalidad de gestión de planes de alimentación para las mascotas. Para ello realizaremos una serie de ejercicios basados en funcionalidades que implementaremos en el sistema, y validaremos mediante pruebas unitarias. Si desea ver el resultado que arrojarían las pruebas, puedes ejecutarlas (bien mediante su entorno de desarrollo favorito, bien mediante el comando "*.\mvnw test*" en la carpeta raíz del proyecto). Cada prueba correctamente pasada valdrá un punto.

Para comenzar el control debe aceptar la tarea de este control práctico a través del siguiente enlace: https://classroom.github.com/a/DqDJnFDC

Al aceptar dicha tarea, se creará un repositorio único individual para usted, debe usar dicho repositorio para realizar el control práctico. Debe entregar la actividad en EV asociada al control check proporcionando como texto la dirección url de su repositorio personal. Recuerde que además debe entregar su solución del control.

La entrega de su solución al control se realizará mediante un único comando "*git push*" a su repositorio individual. Recuerde que debe hacer push antes de cerrar sesión en la computadora y abandonar el aula, de lo contrario, su intento se evaluará como no presentado.

Su primera tarea en este control será clonar (use el comando "*git clone XXXX*" para ello, donde XXXX es la url de su repositorio). A continuación, deberá importar el proyecto en su entorno de desarrollo favorito y comenzar los ejercicios abajo listados. Al importar el proyecto, el mismo puede presentar errores de compilación. No se preocupe, si existen, dichos errores irán despareciendo conforme usted vaya implementando los distintos ejercicios del control.

Nota importante 1: No modifique los nombres de las clases ni la signatura (nombre, tipo de respuesta y parámetros) de los métodos proporcionados como material de base para el control. Las pruebas que se usan para la evaluación dependen de que las clases y los métodos tengan dicha la estructura y nombres proporcionados. Si los modifica probablemente no pueda hacer que pasen las pruebas, y obtendrá una mala calificación.

Nota importante 2: No modifique las pruebas unitarias proporcionadas como parte del proyecto bajo ningún concepto. Aunque modifique las pruebas en su copia local del Proyecto, éstas serán restituidas mediante un comando git previamente a la ejecución de las pruebas para la emisión de la nota final, por lo que sus modificaciones en las pruebas no serán tenidas en cuenta en ningún momento.

Test 1 – Creación de la Entidad Feeding y su repositorio asociado

Modificar la clase "Feeding" para que sea una entidad. Esta clase está alojada en el paquete "org.springframework.samples.petclinic.feeding", y debe tener los siguientes atributos y restricciones:

- El atributo de tipo entero (Integer) llamado "id" actuará como clave primaria en la tabla de la base de datos relacional asociada a la entidad.
- El atributo de tipo fecha (LocalDate) llamado "startDate", que representa el momento en el que se inicia el plan de alimentación, seguirá el formato "yyyy/MM/dd" (puede usar como ejemplo la clase Pet y su fecha de nacimiento para especificar dicho formato) y se almacenará en la base de datos con el nombre de columna "start_date". Este atributo debe ser obligatorio.
- El atributo de tipo numérico de coma flotante de precisión doble (double) llamado "weeksDuration", que indica la duración en semanas del plan de alimentación, debe ser obligatorio y permitir únicamente valores mayores o iguales a 1. En la base de datos se almacenará con el nombre de columna "weeks_duration".
- Debe tener una relación N-1 con la clase Pet utilizando un atributo llamado "pet". Esta relación indicará la mascota a la que se le asigna un plan de alimentación. Todo plan de alimentación debe estar asociado a una mascota (el campo es obligatorio y no puede ser nulo).

Modificar el interfaz "FeedingRepository" alojado en el mismo paquete para que extienda a CrudRepository.

Test 2 – Creación de la Entidad FeedingType

Modificar la clase "FeedingType" alojada en el paquete "org.springframework.samples.petclinic.feeding" para que sea una entidad. Esta entidad debe tener los siguientes atributos y restricciones:

- Un atributo de tipo entero (Integer) llamado "id" que actúe como clave primaria en la tabla de la base de datos relacional asociada a la entidad.
- Un atributo de tipo cadena (String) llamado "name" que no puede ser vacío, de valor único (es decir, que no puede haber en la base de datos otro FeedingType con el mismo nombre), y cuya longitud mínima son 5 caracteres y la máxima 30.
- Un atributo de tipo cadena (String) llamado "description". No se permiten descripciones vacías.
- Una relación N a 1 con la clase PetType (tipo de mascota) que describa el tipo de mascota al que se debe aplicar ese tipo de alimentación. Este valor no puede ser nulo. La columna que representa este atributo en la base de datos debe llamarse "pet_type_id".

Además, se debe descomentar el método "List<FeedingType> findAllFeedingTypes()" en el repositorio de planes de alimentación (FeedingRepository) y anotarlo para que ejecute una consulta que permita obtener todos los tipos de productos existentes.

Test 3 – Modificación del script de inicialización de la base de datos para incluir dos planes de alimentación y dos tipos de alimentación

Modificar el script de inicialización de la base de datos, para que se creen los siguientes planes de alimentación (Feeding) y tipos de alimentación (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:

- id: 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 - Crear una relación N a 1 unidireccional desde Feeding hacia FeedingType

Crear una relación de N a 1 unidireccional desde “Feeding” hacia “FeedingType”, usando como nombre del atributo en la clase “feedingType”. Modificar el script de inicialización de la base de datos para que *Feeding 1* se asocie con *FeedingType 2* y *Feeding 2* se asocie con *FeedingType 1*.

Test 5 – Creación de un Servicio de gestion de los planes de alimentación

Modificar la clase “FeedingService”, para que sea un servicio Spring de lógica de negocio, que permita obtener todos los planes de alimentación (Feeding) registrados (como una lista) usando el repositorio. No modifique por ahora la implementación de los demás métodos del servicio.

Test 6 – Anotar el repositorio para obtener los tipos de planes de alimentación (FeedingType) por nombre, e implementar el método asociado en el servicio de gestión de planes de alimentación.

Crear una consulta personalizada que pueda invocarse a través del repositorio de planes de alimentación que obtenga un tipo de plan de alimentación por nombre. Usarlo para implementar el método "getFeedingType(String name)" del servicio de gestión de planes de alimentación.

Test 7– Creación de un Formatter de tipos de producto

Implementar los métodos del *formatter* para la clase FeedingType (usando la clase llamada "FeedingTypeFormatter" que está ya alojada en el mismo paquete "feeding" con el que estamos trabajando). El formatter debe mostrar los tipos de planes de alimentación usando la cadena de su nombre, y debe obtener un tipo de plan de alimentación dado su nombre, buscándolo en la BD (para esto debe hacer uso del servicio de gestión de planes de alimentación y no del repositorio). Recuerde que, si el formatter no puede obtener un valor apropiado a partir del texto proporcionado, debe lanzar una excepción de tipo "ParseException".

Test 8– Registro de los planes de alimentación y comprobación de la coherencia entre la mascota a la que se le asigna el plan y el tipo de mascota asociado a un tipo de alimentación.

Implementar el método "save" del servicio de gestión de planes de alimentación. Si la mascota especificada en el plan de alimentación no es del tipo de mascota asociado al del tipo de alimentación correspondiente, se debe lanzar una excepción de tipo "UnfeasibleFeedingException" (esta clase está ya creada en el paquete feeding). Además, en caso de lanzarse la excepción, la transacción asociada a la operación de guardado del plan de alimentación debe echarse atrás (hacer rollback).

Test 9 – Creación de un formulario para la creación/edición de planes de alimentación

Se proporciona un formulario en el proyecto para la creación de nuevos planes de alimentación (Feeding). Este formulario permite especificar la fecha de inicio, la duración en semanas, la mascota a la que se le asigna el plan de alimentación y el tipo de plan de alimentación asociado. La página jsp del formulario se ha creado dentro de la carpeta de jsps (webapp/WEB-INF/jsp), en una carpeta llamada "feedings", y el fichero se llama "createOrUpdateFeedingForm.jps".

Se pide crear un método de controlador que permita mostrar el formulario. Debe quedar disponible a través de la url:

http://localhost:8080/feeding/create

El formulario debe aparecer por defecto vacío, y debe contener un campo de texto para la fecha, un campo para especificar la duración en semanas, dos selectores para seleccionar la mascota y el tipo de plan de alimentación y un botón para enviar los datos. Los datos del formulario deben enviarse a la misma dirección usando el método post. Quizás necesite añadir algún método al servicio de gestión de planes de alimentación y tipos de alimentación.

El controlador asociado debe crearse como método de la clase "FeedingController" en el mismo paquete, y pasar al formulario un objeto de tipo Feeding con el nombre "feeding" a través del modelo. Además, se deben pasar desde el controlador a la vista, la colección completa de tipo de alimentación con el nombre

“feedingTypes”, y la colección completa de mascotas con el nombre “pets”. Para hacer esto último deberá necesitará usar el servicio de gestión de mascotas PetService (se ha creado un método apropiado para ello en dicha clase). Recuerde que como parte de la implementación debe modificar la configuración de seguridad de la aplicación para permitir que solo los usuarios administradores (authority “admin”) accedan al formulario.

Test 10 – Creación del Controlador para la creación de nuevos planes de alimentación.

Crear un método de controlador en la clase anterior que responda a peticiones tipo post en la url y se encargue de validar los datos del nuevo plan de alimentación, mostrar los errores encontrados si existieran a través del formulario, y si no existen errores, almacenar el nuevo plan de alimentación a través del servicio de gestión de planes de alimentación. Tenga en cuenta que, si la mascota seleccionada no es de un tipo coherente con el tipo de plan de alimentación elegido, debe capturarse la excepción correspondiente y mostrar el mensaje error “La mascota seleccionada no se le puede asignar el plan de alimentación especificado.” en el campo del plan de alimentación seleccionado del formulario. Tras grabar el plan de alimentación, en caso de éxito, se usará redirección para volver a la página de inicio de la aplicación (welcome). Recuerde que el formato de entrada de la fecha de inicio del plan de alimentación debe ser “yyyy/MM/dd” en caso contrario las pruebas no pasarán.