Control práctico de DP1 (Noviembre 2022)

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/wXt9QAEF

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 (recuerde que si va a usar los equipos del aula para realizar el control necesitará usar un token de autenticación de github como clave, tiene un documento de ayuda a la configuración en el propio repositorio del control). 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.

Nota importante 3: Mientras haya ejercicios no resueltos habrá tests que no funcionan y, por tanto, el comando "*mvnw install*" finalizará con error. Esto es normal debido a la forma en la que está planteado el control y no hay que preocuparse por ello. Si se quiere probar la aplicación se puede ejecutar de la forma habitual pese a que "mvn install" finalice con error.

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). Este atributo debe ser obligatorio. En la base de datos se almacenará con el nombre de columna "start_date".
- 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".

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 y cuya longitud mínima son 5 caracteres y la máxima 30.
- Un atributo de tipo cadena (String) llamado "description". No se permiten descripciones vacías.

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 alimentación 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

Feeding 2:

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

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.

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.

Test 4 – 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 y proporcione una implementación a los métodos que permitan:

1. Obtener todos los planes de alimentación (*Feeding*) almacenados en la base de datos como una lista usando el repositorio (método *getAll*).
2. Obtener todo el conjunto de tipos de planes de alimentación (*FeedingType*) registrados (método *getAllFeedingTypes*).
3. Grabar un plan de alimentación (*Feeding*) en la base de datos (método *save*).

Todos estos métodos **deben ser transaccionales.**

No modifique por ahora la implementación del resto de métodos del servicio.

Test 5 – Creación del controlador para mostrar un formulario de creación de planes de alimentación

Crear un método en el controlador "*FeedingController"* (hay que anotar la clase previamente para que sea un controlador) que permita mostrar el formulario implementado en el fichero JSP llamado "*createOrUpdateFeedingForm.jsp*" en la carpeta "*webapp/WEB-INF/jsp/feedings*". Este formulario permite crear y editar planes de alimentación (*Feeding*) especificando la fecha y duración del plan de alimentación. El formulario debe mostrarse en la URL:

http://localhost:8080/feeding/create

El controlador debe pasar al formulario un objeto de tipo *Feeding* con el nombre *feeding* a través del modelo de la vista. Por defecto, el formulario debe aparecer vacío ya que estamos creando un nuevo plan de alimentación. Los datos del formulario deben enviarse a la misma dirección usando el método post.

Test 6 – Creación del controlador para la grabación de los nuevos planes de alimentación

Crear un método de controlador en la clase anterior que responda a peticiones tipo post en la url http://localhost:8080/feeding/create y se encargue de validar los datos del plan de alimentación, mostrar los errores encontrados si existieran a través del formulario, y si no existen errores, almacenar el plan de alimentación a través del servicio de gestión de planes de alimentación.

Tras grabar la vacunación, en caso de éxito, se mostrará la página de inicio de la aplicación (welcome) **usando redirección**. Recuerde que el formato de entrada de la fecha de la vacunación debe ser "yyyy/MM/dd" en caso contrario las pruebas no pasarán (puede usar como ejemplo la clase Pet y su fecha de nacimiento para especificar dicho formato).

Test 7 - Crear relaciones N a 1 unidireccionales desde Feeding hacia FeedingType y Pet, y desde FeedingType a PetType

Crear una relación de N a 1 unidireccional desde "Feeding" hacia "FeedingType", usando como nombre del atributo en la clase "feedingType". Este atributo no puede ser nulo.

Crear una relación N a 1 unidireccional desde "Feeding" hacia "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 (no puede ser nulo).

Crear una relación N a 1 unidireccional desde "FeedingType" hacia "PetType" (tipo de mascota), usando como nombre del atributo "*petType*". Este atributo describirá el tipo de mascota al que se debe aplicar ese tipo de alimentación. Este valor no puede ser nulo.

Modificar el script de inicialización de la base de datos para que:

- El *FeedingType* cuyo *id* es 1 se asocie con el *PetType* con id=2
- El *FeedingType* cuyo *id* es 2 se asocie con PetType con id=1
- El *Feeding* cuyo id es 1 se asocie con el *Pet* con id=7 y *FeedingType* con id=*2*
- El *Feeding cuyo id es 2* se asocie con el *Pet* con id=4 y *FeedingType* con id=*1*

Tenga en cuenta que el orden en que aparecen los INSERT en el script de inicialización de la base de datos es relevante al definir las asociaciones.

Test 8– Creación de un Formatter de tipos de planes de alimentación

Crear un método con una consulta personalizada en el repositorio de planes de alimentación para que obtenga un tipo de plan de alimentación por nombre.

Usar el método anterior para implementar el método "getFeedingType(String name)" del servicio de gestión de planes de alimentación.

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 método 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 9– 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.

Modificar el método "*save*" del servicio de gestión de planes de alimentación para que, si la mascota especificada en el plan de alimentación no es del tipo de mascota asociado al del tipo de alimentación correspondiente, se lance 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 10 - Modificación de los controladores de visualización de formulario y creación de planes de alimentación

Descomentar el código que aparece comentado en el formulario de creación de planes de alimentación utilizado en el Test 5 para que permita especificar además de los campos especificados anteriormente, el tipo de plan de alimentación concreto fijado (el formulario debe mostrar el nombre del tipo de plan de alimentación, enviando su id como valor), y la mascota a la que se le define el plan de alimentación (el formulario debe mostrar el nombre de la mascota, enviando su id como valor).

Modificar el controlador para mostrar el formulario de creación de planes de alimentación de manera que, además del "*feeding*" se pase al modelo de la vista:

1. La colección completa de tipo de alimentación con el nombre "feedingTypes".
2. 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).

Modificar el controlador de grabación de planes de alimentación de manera que si la mascota seleccionada no es de un tipo coherente con el tipo de plan de alimentación fijado (conforme a lo especificado en el ejercicio anterior), debe capturarse la excepción correspondiente y mostrar el mensaje error "La mascota seleccionada no puede realizar el plan de alimentación especificado" en el campo del tipo de plan de alimentación seleccionado del formulario.

Modificar la configuración de seguridad de la aplicación para permitir que solo los usuarios administradores (authority "admin") accedan al formulario y a la grabación de planes de alimentación.