Control práctico de DP1 Grupo 1

Enunciado

En este ejercicio, añadiremos la funcionalidad de gestión de la vacunación de 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 tu entorno de desarrollo favorito, o con 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/Ol8rfJv1

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 el repositorio. A continuación, deberá importar el proyecto en su entorno de desarrollo favorito y comenzar los ejercicios abajo listados. Al importar el proyecto, es posible que el mismo presente 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 Vaccination y su repositorio asociado

Se propone modificar la clase "Vaccination" para que sea una entidad. Esta entidad estará alojada en el paquete "org.springframework.samples.petclinic.vaccination", y debe tener los siguientes atributos y restricciones:

- Un atributo de tipo Integer llamado id que actúe como clave primaria en la tabla de la base de datos relacional asociada a la entidad.
- Un atributo "date" de tipo fecha (LocalDate) obligatorio. Este atributo representará la fecha en la que realizó la vacunación. El atributo debe seguir el formato de fecha "yyyy/MM/dd". Puedes usar como ejemplo la propiedad birthDate de la clase Pet.
- Una relación N a 1 con la clase Pet, que representará la mascota que se vacuna. El nombre de la columna que representa la relación en la BD (joincolumn) debe ser "vaccinated_pet_id". Toda vacunación debe tener asociada una mascota (el campo es obligatorio).

Se propone modificar el interfaz "VaccinationRepository" alojado en el mismo paquete, para que extienda a CrudRepository. (Si desea ejecutar el Test1, deberá comentar temporalmente el método findAllVaccines, si no lo estuviera ya, y una vez resuelto el test 2, todo debería funcionar sin problema con el método descomentado).

Test 2 – Creación de la Entidad Vaccine

Modificar la clase denominada "Vaccine" alojarse en el paquete "org.springframework.samples.petclinic.vaccination" para que sea una Entidad. Esta entidad debe tener los siguientes atributos y restricciones:

- Un atributo de tipo Integer llamado "Id" que actúe como clave primaria en la tabla de la base de datos relacional asociada a la entidad.
- Un atributo "name" de tipo cadena obligatorio, y cuya longitud minima son 3 carácteres y la máxima 50. Además, **el nombre de la vacuna debe ser único (use la anotación a nivel de columna, no la de clase)**.
- Una relación N a 1 con la clase Tipo de Mascota obligatoria (no opcional), que describa el tipo de mascota al que se debe aplicar la vacuna. La columna que representa el id del tipo de mascota a nivel de base de datos (joincolumn) debe llamarse "pet_type_id".
- Un atributo "price" de tipo Double, cuyo valor debe ser positivo (cero incluido).

Además, se debe anotar el método findAllVaccines, del repositorio de vacunaciones, para que ejecute una consulta que permitan obtener todas las vacunas existentes.

Test 3 – Modificación del script de inicialización de la base de datos para incluir dos vacunaciones y dos vacunas

Modificar el script de inicialización de la base de datos, para que se creen las siguientes vacunaciones y vacunas:

Vacunación 1:

- id: 1
- date: 2021-12-08
- vaccinated_pet_id: 1

Vacunación 2:

- id: 2
- date: 2021-10-08
- vaccinated_pet_id: 3

Vacuna 1:

- Id: 1
- name: “Anti-rabid”
- pet_type_id: 2
- price: 50.0

Vacuna 2:

- Id: 2
- name: “Covid19 for Cats”
- pet_type_id: 1
- Price: 1200.5

Test 4 - Crear una relación N a 1 unidireccional desde Vaccination hacia Vaccine

Además de crear una relación de N a 1 unidireccional desde Vaccination hacia Vaccine, se propone modificar el script de inicialización de la base de datos para que la Vacunación 1 corresponda con la Vacuna 2 y la Vacunación 2 con la Vacuna 1.

Test 5 – Creación de un Servicio de gestion de la vacunación

Modificar la clase VaccinationService, para que sea un servicio Spring de lógica de negocio, que permita obtener todas las Vacunaciones realizadas (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 las Vacunas por nombre, e implementar el método asociado en el servicio de gestión de Vacunaciones

Crear una consulta personalizada que pueda invocarse a través del repositorio de vacunaciones que obtenga una vacuna por nombre. Exponerlo a través del servicio de gestión de vacunaciones mediante el método “getVaccine(String name)”.

Test 7– Creación de un Formatter de Vacunas

Implementar los métodos del formatter para la clase Vaccine (usando la clase llamada VaccineFormatter que debe está ya alojada en el mismo paquete con el que estamos trabajando). El formatter debe mostrar las vacunas usando la cadena de su nombre, y debe obtener una Vacuna dado su nombre buscándolo en la BD (a través del servicio de gestión de vacunaciones, no del repositorio). Recuerde que, si el formatter no puede parsear un valor apropiado a partir del texto proporcionado, debe lanzar una excepción de tipo ParseException.

Test 8– Grabación de las vacunaciones y comprobación de la coherencia entre la mascota vacunada y el tipo de mascota de la vacuna

Implementar el método save del servicio de gestión de vacunaciones. Si la mascota especificada en la vacunación no es del tipo asociado a la vacuna correspondiente, se debe lanzar una excepción de tipo UnfeasibleVaccinationException (esta clase está ya creada en el paquete vaccination). Además, en caso de lanzarse la excepción, la transacción asociada a la operación de guardado de la vacunación debe echarse atrás (hacer rollback).

Test 9 – Creación de un controlador para la creación de vacunaciones

Se proporciona un formulario en el proyecto para la creación de vacunaciones. Permite especificar La vacuna, la mascota, y la fecha de la vacunación. La página jsp del formulario se ha creado dentro de la carpeta de jsps (webapp/WEB-INF/jsp), en una carpeta llamada "vaccination", y el fichero se llama "createOrUpdateVaccinationForm.jsp".

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

http://localhost:8080/vaccination/create

El formulario debe aparecer por defecto vacío, y contiene dos selectores para escoger la mascota y la vacuna, un campo de texto para la fecha, y un botón para enviar los datos. Los datos del formulario se envían a la misma dirección usando el método post. Quizás necesite añadir algún método al servicio de gestión de vacunaciones.

El controlador asociado debe crearse como método de la clase "VaccinationController" en el mismo paquete, y pasar al formulario un objeto de tipo Vaccination con el nombre "vaccination" a través del modelo. Además, se deben pasar desde el controlador a la vista, la colección completa de vacunas, con el nombre "vaccines", y la colección completa de mascotas, con el nombre "pets". Para obtener esto último 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 nuevas Vacunaciones

Se propone 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 de la vacunación, mostrar los errores encontrados si existieran a través del formulario, y si no existen errores, almacenar la vacunación a través del servicio de gestión de vacunas. Tenga en cuenta que, si la mascota seleccionada no es de un tipo coherente con la vacuna a inocular, debe capturarse la excepción correspondiente y mostrar el mensaje error "La mascota seleccionada no puede recibir la vacuna especificada." en el campo de la vacuna seleccionada del formulario. Tras grabar la vacunación, en caso de éxito, se mostrará la página de inicio de la aplicación (welcome), sin usar 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).