Ejercicios de Relaciones entre objetos

Índice

Se proporciona el fichero **hib-sesion3.doc** como plantilla para que contestéis a las preguntas formuladas en los ejercicios.

Vamos a utilizar el proyecto `hib-sesion3`, que se encuentra en el fichero **sesion03-ejercicios.zip**. En dicho proyecto realizaremos las modificaciones que se piden en los ejercicios propuestos.

1. Ejercicio 1: Asociación unidireccional

Utilizando la clase *Person* de la sesión anterior, y la nueva clase *Bill*, que representa las facturas asociadas de una persona, tenéis que realizar los cambios necesarios para mapear una relación de asociación de uno a muchos unidireccional entre dichas clases desde *Person* a *Bill* (es decir, no será posible navegar desde *Bill* hacia *Person*). Añadiremos la restricción de que una factura no puede existir si no está asociada a una persona. Una vez implementada la asociación, tienes que:

- Implementar el método *public void createInitialPersons()* (fichero *PersonManager.java*) que inicializa los valores de la tabla *PERSON* con los mismos datos de la sesión anterior:(10,"Pepa","Flores", "1111","2222"),(20,"Maria","Flores", "1111","2222"), (45,"Juan","Flores","1111","2222"), (30,"Pepe","Flores", "1111","2222"), (56,"Marta","Flores", "1111","2222")
- Implementar el método *public void createAndStoreBill(Long theNumber, Date theDate, Long cantidad)* (fichero *BillManager.java*) que crea una nueva fila en la tabla *BILLS*.
- Implementar el método *public void addBillToPerson(Long idPerson, Long numberBill, Date dateBill, Long amountBill)* en *PersonManager.java*. Dicho método añade una factura a una persona cuyo identificador es *idPerson*. Prueba a habilitar la popagación en cascada de las operaciones *"save"* y *"update"* con *cascada="save-update"* y justifica las diferencias en la implementación de *addBillToPerson* con y sin habilitar la propagación en cascada. Para probar el método puedes hacerlo con: *addBillToPerson(2,1000,10/10/05,500))*
- Implementar el método *public void printBillsFromPerson(Long personID)* en *PersonManager.java* que, dado el identificador de una persona, imprima por pantalla los datos de las facturas de dicha persona. Si, por ejemplo, hemos añadido a la persona con identificador 2 una factura con número 1000, fecha 10/10/2005 y cantidad 500, nos aparecerá por pantalla:

  Facturas de la persona 2

  ID: 1 Número Factura: 1000 Fecha: 10/oct/2005 Cantidad: 500
- **OPCIONAL:** Implementar el método *public void changeAmountFromPerson(Long personID, Long billNumber, Long newAmount)* en *PersonManager.java* que, dado el

identificador de una persona (*personID*) y el número de factura (*billNumber*), cambia el importe a *newAmount*. Nota: Para comparar dos elementos utiliza el método `equals` en vez de ==. Explica por qué tenemos que cargar en memoria de forma explícita las instacias de factura para modificar su importe. Indica qué modificaciones tendríamos que hacer en el código para que las instancias de la colección de facturas se cargaran automáticamente por Hibernate al acceder a una persona.

2. Ejercicio 2: Asociación Bidireccional

En este ejercicio vamos a seguir añadiendo código a la aplicación anterior. En este caso se trata de:

- Añadir una clase *Event*, con los campos: *String name*, *Date date*, y *String site* (podéis utilizar la clase *Event* de los ejercicios de la sesión anterior).
- La nueva clase *Event* tiene una relación **muchos a muchos bidireccional** con la clase *Person*. Implementa los cambios necesarios en los POJOS y ficheros de mapeado de ambas clases para dicha asociación bidireccional.
- Puedes reutilizar el fichero *EventManager.java* de la sesión anterior, concretamente los métodos *createAndStoreEvent*, *createInitialEvents*, *listEvents*, *printEvents* y *obtenerFecha*. Utiliza los métodos *createInitialEvents()* y *createInitialPersons()* para crear eventos y personas inicialmente, **cada vez** que ejecutemos el programa. Utiliza los mismos datos que ya hemos introducido en ejercicios anteriores.
- Implementa el método *public void addPersonToEvent (Long personId, Long eventId)* que, añade una persona a un determinado evento (en el fichero *EventManager.java*. Modifica convenientemente las clases *Person.java* y *Event.java* con los métodos *addEvent()* y *addPerson()*, respectivamente. Prueba a añadir los siguientes datos: (persona 1, evento 3), (persona 2, evento 3), (persona 4, evento 3).
- Implementa el método *public void removePersonFromEvent (Long personId, Long eventId)* que, "borra" a una persona de un determinado evento. También en el fichero *EventManager*. Prueba a deshacer los cambios del punto anterior.