Hibernate_IV

Sesiones y objetos Hibernate

- Como ya hemos visto, para poder utilizar los mecanismos de persistencia de Hibernate se debe inicializar el entorno Hibernate y obtener un objeto **Session** utilizando la clase **SessionFactory**.

```
//Inicializa el entorno Hibernate: carga el fichero hibernate.cfg.xml
Configuration cfg = new Configuration().configure();

//Crea el ejemplar de SessionFactory. Se necesita crear un objeto
//StandardServiceRegistry que contiene la lista de servicios de Hibernate
//El ejemplar de SessionFactory se crea normalmente solo una vez y se utiliza para
//crear todas las sesiones relacionadas con el contexto dado (singleton)
SessionFactory sessionFactory = cfg.buildSessionFactory (
        new StandardServiceRegistryBuilder().configure().build() ) ;

//Obtiene un objeto Session
Session session = sessionFactory.openSession();
```

Transacciones

- Un objeto **Session** de Hibernate representa una única unidad de trabajo. Al crear la sesión se crea la transacción para dicha sesión. Se deben cerrar las sesiones.

```
Session session = session.openSession();
Transaction tx = session.beginTransaction();
// …. código de persistencia ...//
tx.commit();
session.close();
```

- El método **beginTransaction()** marca el inicio de una transacción. El método **commit()** la valida; y el método **rollback()** deshace los cambios.

Estados de un objeto Hibernate

- **Transitorio** (Transient) : Un objeto es transitorio si ha sido recién instanciado y no está asociado a una Session de Hibernate. No tiene una representación persistente en la bbdd y no se le ha asignado un identificador.
- **Persistente** (Persistent) : Un objeto estará en este estado cuando ya está almacenado en la bbdd. Puede haber sido guardado o cargado pero se encuentra en el ámbito de una **Session.**
- **Separado** (Detached) : Una instancia separada es un objeto que se ha hecho persistente pero su sesión ha sido cerrada. La referencia al objeto todavía es válida, y podría ser modificado. Para hacer persistentes dichas modificaciones debemos asociar el objeto a una nueva **Session**

Carga de objetos

Para la carga de objetos usaremos los siguientes métodos de **Session:**

- **<T> T load (Class<T> Clase, Serializable id)** → Devuelve la instancia persistente de la clase indicada con el identificador dado. Si la instancia no existe el método lanza una excepción.
- **Object load (String nombreClase, Serializable id)** → Lo mismo que antes pero indicando nombre de la clase
- **<T> T get (Class<T> Clase, Serializable id)** → Lo mismo pero si la instancia no existe devuelve *null*
- **Object get (String nombreClase, Serializable id)** → Lo mismo que en el caso anterior pero indicando el nombre de la clase.

Primer ejemplo: visualiza los datos del departamento 20

```
Depart dep = new Depart();
try  {
        dep = (Depart) session.load(Depart.class,  20);
        System.out.printf("Nombre Dep: %s%n", dep.getDnombre());
} catch (ObjectNotFoundException o) {
        System.out.println ("No existe el departamento");
}
```

Segungo ejemplo: lo mismo pero utilizando otro formato de load()

```
Depart dep = new Depart();
try  {
        dep = (Depart) session.load(primero.Depart,  20);
        System.out.printf("Nombre Dep: %s%n", dep.getDnombre());
} catch (ObjectNotFoundException o) {
        System.out.println ("No existe el departamento");
}

//El método load lanza la excepción ObjectNotFound si la fila no existe
```

Tercer ejemplo: uso del método get. Será conveniente cuando no tengamos la certeza de la existencia de la fila

```
Depart dep =  (Depart) session.get (Depart.class,  11);

if (dep==null) {
        System.out.println (“No existe el departamento”);
}
else {
        System.out.print(“Nombre Dep: %s%n”, dep.getDnombre());
}
```

Problemas HibernateIV_1

1. Visualiza el apellido y el salario del empleado con número: 7369
2. Obtén los datos del departamento 10 y el APELLIDO y SALARIO de sus empleados. Ayuda: para obtener los empleados usamos el método **getEmples()** de la clase Depart y usamos un **Iterator** (java.util.Iterator) para recorrer la lista de empleados. Algo así:

```
import java.util.Set;
import java.util.Iterator;
(....)
        //dep es una instancia permanente de Depart

        Set<Emple> listaemp = dep.getEmples();
        Iterator<Emple> it = listaemp.iterator();
(...)
```

Almacenamiento, modificación y borrado de objetos

Disponemos de los siguientes métodos:

- **Serializable save (Object obj)** → Guarda el objeto que se pasa como argumento en la base de datos. Hace que la instancia transitoria del objeto sea persistente
- **void update (Object obj)** → Actualiza en la bbdd el objeto que se pasa como argumento. El objeto a modificar debe ser cargado con el método **load()** o **get()**
- **void delete (Object obj)** → Elimina de la bbdd el objeto que se pasa como argumento. El objeto a eliminar debe ser cargado con el método **load()** o **get().**

Primer ejemplo: generar y guardar un nuevo departamento

```
Depart dep = new Depart();
dep. setDeptNo (70);
dep.setDnombre (“INFORMATICA”);
dep.setLoc(“TOLEDO”);
session.save(dep);
```

Segundo ejemplo: eliminación del empleado 7369

```
Emple emp = new Emple();
emp = (Emple) session.load ( Emple.class, 7369);
session.delete (emp);
```

Tercer ejemplo: eliminación de un departamento controlando las distintas excepciones: que el departamento no exista y que tenga empleados

```
import org.hibernate.ObjectNotFoundException;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.exception.ConstraintViolationException;
public class BorradoDep {
	public static void main(String[] args) {
		SessionFactory sesion = HibernateUtil.getSessionFactory();
		Session session = sesion.openSession();
		Transaction tx = session.beginTransaction();
		Depart dep = (Depart) session.load(Depart.class, 10);
		try {
			session.delete(dep);
			tx.commit();
			System.out.println ("Departamento eliminado");
		} catch (ObjectNotFoundException c) { System.out.println ("No existe el departamento");}
		catch (ConstraintViolationException c) { System.out.println ("No se puede eliminar. Tiene
empleados");}
		catch (Exception e) { e.printStackTrace();}
	session.close(); System.exit(0); } }
```

Problemas HibernateIV_2

1. Para modificar un objeto, igual que para borrarlo, primero hemos de cargarlo, a continuación realizamos la modificaciones y, por último, utilizamos el método **update()**. Realiza un programa que modifique el salario y el departamento del empleado 7369, sumando 1000 al salario y asignándole el departamento 30.